Guia do usuário do Aurora

Amazon Aurora

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Aurora: Guia do usuário do Aurora

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, conectados ou patrocinados pela Amazon.

Table of Contents

O que é o Aurora? 1
Modelo de responsabilidade compartilhada do Amazon RDS 2
Como o Amazon Aurora funciona com o Amazon RDS 2
clusters de bancos de dados Aurora 4
Versões do Aurora 6
Mecanismos de banco de dados compatíveis 6
Especificar uma versão do Aurora 7
Versionamento do Aurora 7
Atualizações de cluster de banco de dados do Aurora 14
Suporte para a versão do Aurora 16
Regiões e zonas de disponibilidade 19
AWSRegiões de 20
Zonas de disponibilidade 29
Fuso horário local para os clusters de banco de dados do 30
Recursos Aurora compatíveis por região e mecanismo 37
Convenções de tabela 38
Implantações azul/verde 38
Configurações de cluster do Aurora 39
Fluxos de atividades do banco de dados 40
Exportar dados do cluster para o Amazon S3 49
Exportar dados de snapshot para o Amazon S3 51
Bancos de dados globais do Aurora 52
Autenticação do banco de dados do IAM 62
Autenticação de Kerberos 63
Machine learning do Aurora 75
Insights de Performance 85
Integrações ETL zero 96
RDS Proxy 104
Integração do Secrets Manager 114
Aurora Serverless v2 115
Aurora Serverless v1 da 121
API de dados do RDS 126
Aplicação de patches com tempo de inatividade zero (ZDP) 137
Base de datos ilimitada de Aurora PostgreSQL 137

Atributos nativos do mecanismo ... 138
Conexões do endpoint ... 138
Tipos de endpoints do Aurora ... 139
Visualizar endpoints ... 141
Endpoints e alta disponibilidade ... 142
Endpoints do cluster ... 143
Endpoints de leitor ... 144
Endpoints da instância ... 145
Endpoints personalizados ... 146
Classes de instâncias de banco de dados ... 166
Tipos de classe de instância de banco de dados ... 166
Mecanismos de banco de dados compatíveis ... 170
Determinar o suporte para classes de instância de bancos de dados nas Regiões da AWS ... 183
Especificações de hardware ... 188
Armazenamento ... 202
Visão geral do armazenamento do Aurora ... 203
Conteúdo do volume de cluster ... 203
Configurações de armazenamento do cluster do Aurora ... 204
Como o armazenamento redimensiona ... 205
Faturamento de dados ... 206
Confiabilidade ... 207
Reparo automático de armazenamento ... 207
Cache de página perdurável ... 207
Recuperação após reinicializações não planejadas ... 208
Segurança do Aurora ... 209
Uso do SSL com clusters de banco de dados Aurora ... 211
Alta disponibilidade ... 211
Alta disponibilidade de dados do Aurora ... 211
Alta disponibilidade de instâncias de banco de dados do Aurora ... 212
Alta disponibilidade entre as regiões da AWS com bancos de dados globais do Aurora ... 213
Tolerância a falhas ... 213
Alta disponibilidade com o Amazon RDS Proxy ... 216
Replicação ... 216
Réplicas do Aurora ... 216
Aurora MySQL ... 219

Aurora PostgreSQL 219
Faturamento de instâncias de banco de dados para o Aurora 220
Instâncias de banco de dados sob demanda 223
Instâncias de bancos de dados reservadas 225
Amazon no nível gratuito da AWS 241
Configuração de seu ambiente 242
Inscrever-se para uma Conta da AWS 242
Criar um usuário com acesso administrativo 243
Conceder acesso programático 244
Determinar requisitos 246
Fornecer acesso ao cluster de banco de dados 248
Conceitos básicos 252
Criar um cluster de banco de dados do Aurora MySQL e se conectar a ele 252
Pré-requisitos 254
Etapa 1: Criar uma instância do EC2 254
Etapa 2: Criar um cluster de banco de dados do Aurora MySQL 260
(Opcional) Criar VPC, instância do EC2 e cluster do Aurora MySQL usando o CloudFormation 265
Etapa 3: Conectar-se a um cluster de bancos de dados do Aurora MySQL 267
Etapa 4: Excluir a instância do EC2 e o cluster de banco de dados 270
(Opcional) Excluir a instância do EC2 e o cluster de banco de dados criado com o CloudFormation 271
(Opcional) Conectar o cluster de banco de dados a uma função do Lambda 272
Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele 272
Criar com a configuração completa 272
Criar com a configuração expressa 292
Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora 317
Iniciar uma instância do EC2 para se conectar ao cluster de banco de dados 319
Criar um cluster de banco de dados 325
Instalar um servidor Web 336
Tutoriais e código de exemplo 349
Tutoriais neste guia 349
Tutoriais em outros guias daAWS 350
Tutoriais e código de exemplo no GitHub 351
Livro de fórmulas para bancos de dados da AWS 352
Workshop da AWS: Amazon Aurora PostgreSQL 353

Workshop da AWS: Amazon Aurora MySQL 354
Como trabalhar com AWS SDKs 356
Acesso programático ao Amazon Aurora 358
Console-to-Code 359
Configurar um cluster de banco de dados do Aurora 360
Criar um cluster de banco de dados 361
Pré-requisitos 362
Criar um cluster de banco de dados 369
Configurações disponíveis 380
Configurações não aplicáveis ao Aurora para clusters de banco de dados 403
Configurações não aplicáveis a instâncias de banco de dados do Aurora 404
Criar recursos com o AWS CloudFormation 407
Aurora e modelos do CloudFormation 407
Saiba mais sobre o CloudFormation 407
Conexão com um cluster de banco de dados 408
Conectar-se aos clusters de banco de dados do Aurora com os drivers da AWS 409
Como conectar-se ao Aurora MySQL 411
Como conectar-se ao Aurora PostgreSQL 418
Solução de problemas de conexões 421
Grupos de parâmetros 422
Visão geral dos grupos de parâmetros 422
Grupos de parâmetros de cluster de banco de dados 427
Grupos de parâmetros do banco de dados 447
Comparação de grupos de parâmetros de banco de dados 464
Especificação de parâmetros de banco de dados 465
Migrar dados para um cluster de banco de dados 471
Aurora MySQL 471
Aurora PostgreSQL 471
Criar um cache do ElastiCache pelo Amazon RDS 472
Visão geral da criação do cache do ElastiCache com as configurações do cluster de banco de dados do Aurora 472
Criar um cache do ElastiCache com as configurações de um cluster de banco de dados do Aurora 473
Migração automática de bancos de dados do EC2 477
Visão geral 477
Pré-requisitos 478

Limitações ... 479
Criar recursos do IAM ... 480
Configurar a migração de dados ... 488
Gerenciamento de migrações ... 490
Monitoramento ... 492
Tutorial: criar um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado ... 494
Pré-requisitos ... 495
Criar um grupo de parâmetros personalizado ... 495
Modificar o valor do parâmetro ... 496
Criar um cluster de banco de dados ... 496
Como gerenciar um cluster de banco de dados Aurora ... 498
Interromper e iniciar um cluster ... 499
Visão geral de como interromper e iniciar um cluster ... 499
Limitações ... 501
Interromper um cluster de banco de dados ... 501
Enquanto um cluster de banco de dados estiver interrompido ... 502
Como iniciar um cluster de banco de dados ... 503
Conectar uma instância do EC2 ... 505
Visão geral ... 506
Conectar uma instância do EC2 ... 511
Visualizar recursos computacionais conectados ... 514
Conectar a uma instância de banco de dados executando um mecanismo de banco de dados específico ... 515
Conectar uma função do Lambda ... 516
Visão geral ... 518
Conectar uma função do Lambda ... 529
Visualizar recursos computacionais conectados ... 531
Modificar um cluster de bancos de dados Aurora ... 533
Modificar o cluster de banco de dados usando o console, a CLI e a API ... 533
Modificar uma instância de banco de dados em um cluster de banco de dados ... 536
Alterar a senha do usuário mestre ... 539
Configurações disponíveis ... 541
Configurações não aplicáveis a clusters de banco de dados Aurora ... 579
Configurações não aplicáveis a instâncias de banco de dados Aurora ... 580
Adicionar réplicas do Aurora ... 582

Auto Scaling com réplicas do Aurora ... 589
Adicionar uma política de ajuste de escala automático ... 596
Editar uma política de ajuste de escala automático ... 608
Excluir uma política de ajuste de escala automático ... 611
Gerenciar a performance e a escalabilidade ... 614
Escalabilidade de armazenamento ... 614
Escalabilidade de instâncias ... 621
Escalabilidade de leitura ... 622
Como gerenciar conexões ... 622
Gerenciar planos de execução de consulta ... 623
Clonar um volume para um cluster de banco de dados do Aurora ... 624
Visão geral da clonagem do Aurora ... 624
Limitações da clonagem do Aurora ... 625
Como a clonagem do Aurora funciona ... 627
Criar um clone do Aurora ... 630
Clonagem entre VPCs ... 641
Clonagem entre contas ... 660
Integrar com serviços da AWS ... 678
Aurora MySQL ... 678
Aurora PostgreSQL ... 678
Manutenção de um cluster de banco de dados do Aurora ... 680
Visão geral das atualizações de manutenção do cluster de banco de dados ... 680
Visualização de manutenção pendente ... 682
Escolher a frequência das atualizações de manutenção do Aurora MySQL ... 684
Maintenance window ... 685
Aplicação de atualizações ... 691
Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora ... 693
Atualizações do sistema operacional ... 698
Implementação de atualizações do AWS Organizations ... 707
Reinicializar um cluster ou instância de banco de dados do Aurora ... 711
Reinicializar uma instância de banco de dados em um cluster do Aurora ... 712
Reinicializar um cluster do Aurora com disponibilidade de leitura ... 713
Reinicializar um cluster do Aurora sem disponibilidade de leitura ... 715
Verificar o tempo de atividade para clusters e instâncias do Aurora ... 716
Exemplos de operações de reinicialização do Aurora ... 719

Fazer failover de um cluster de banco de dados do Aurora ... 736
Descrever clusters e instâncias do Aurora ... 738
Excluir um cluster de banco de dados do Aurora ... 738
Proteção contra exclusão para clusters do Aurora. ... 747
Excluir um cluster parado do Aurora ... 747
Excluir clusters do Aurora MySQL que são réplicas de leitura ... 747
O snapshot final ao excluir um cluster ... 748
Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora ... 748
Marcar recursos do Aurora e do RDS ... 751
Por que usar tags do RDS? ... 752
Como as tags do RDS funcionam ... 753
Práticas recomendadas ... 756
Copiar tags para snapshots de cluster de banco de dados ... 757
Marcar backups automatizados ... 758
Adicionar e excluir tags no Amazon RDS ... 758
Tutorial: Uso de tags para especificar quais clusters de bancos de dados Aurora interromper ... 763
ARNs no Amazon RDS ... 767
Criação de um ARN ... 767
Obter um ARN existente ... 775
Atualizações do Aurora ... 779
Identificar a versão do Amazon Aurora ... 780
Suporte estendido do RDS ... 781
Visão geral do Suporte estendido do RDS ... 782
Cobranças do Suporte estendido do RDS ... 783
Evitar cobranças do Suporte estendido do RDS ... 784
Versões com o Suporte estendido do RDS ... 784
Responsabilidades com o Suporte estendido do RDS ... 785
Responsabilidades do Amazon Aurora ... 785
Suas responsabilidades ... 785
Criação de um cluster de banco de dados do Aurora ou um cluster global ... 786
Comportamento do Suporte estendido do RDS ... 786
Considerações para o Suporte estendido do RDS ... 787
Criar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS ... 788
Visualizar a inscrição no Suporte estendido do RDS ... 789

Visualizar datas de suporte ... 792
Restauração de um cluster de banco de dados do Aurora ou um cluster global ... 794
Comportamento do Suporte estendido do RDS ... 795
Considerações para o Suporte estendido do RDS ... 795
Restaurar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS ... 796
Usar implantações azul/verde para atualizações de banco de dados ... 798
Visão geral das implantações azul/verde ... 799
Disponibilidade de região e versão ... 800
Benefícios ... 800
Fluxo de trabalho ... 800
Autorizar acesso ... 807
Limitações e considerações ... 808
Práticas recomendadas ... 818
Criar uma implantação azul/verde ... 822
Preparação para uma implantação azul/verde ... 823
Especificando alterações ... 825
Criar uma implantação azul/verde ... 826
Configurações disponíveis ... 828
Visualizar uma implantação azul/verde ... 830
Alternar uma implantação azul/verde ... 834
Tempo limite de transição ... 835
Barreiras de proteção de transição ... 835
Ações de transição ... 836
Práticas recomendadas de transição ... 838
Verificar as métricas do CloudWatch antes da transição ... 839
Monitorar o atraso da réplica antes da transição ... 840
Realizar a transição de uma implantação azul/verde ... 840
Após a transição ... 843
Excluir uma implantação azul/verde ... 845
Como fazer o backup e a restauração de um cluster de banco de dados Aurora ... 848
Visão geral do backup e da restauração ... 849
Backups ... 849
Janela de backup ... 850
Restaurar dados ... 854
Clonagem de banco de dados ... 854

Retrocesso ... 855
Reter backups automatizados ... 856
Período de retenção ... 856
Visualização de backups retidos ... 857
Custos de retenção ... 857
Limitações ... 857
Excluir backups automatizados retidos ... 858
Armazenamento de backup ... 860
Armazenamento de backups automatizados ... 860
Armazenamento de snapshots ... 861
Métricas do CloudWatch para armazenamento de backups ... 861
Calcular o uso do armazenamento dos backups ... 862
Perguntas frequentes ... 864
Criar um snapshot de cluster de banco de dados ... 867
Como determinar se o snapshot está disponível ... 869
Restauração de um snapshot de um cluster de banco de dados ... 870
Grupos de parâmetros ... 871
Grupos de segurança ... 871
Considerações sobre o Aurora ... 872
Restauração a partir de um snapshot ... 872
Cópia de snapshot de cluster de banco de dados ... 876
Copiar um snapshot de cluster de banco de dados com o Console de gerenciamento da AWS ... 876
Limitações ... 878
Considerações ... 879
Copiar um snapshot do cluster de banco de dados não criptografado ... 882
Copiar um snapshot do cluster de banco de dados criptografado ... 884
Copiar um snapshot de cluster de banco de dados entre contas ... 888
Compartilhar um snapshot do cluster de banco de dados ... 893
Compartilhar um snapshot ... 894
Compartilhamento de snapshots públicos ... 898
Compartilhamento de snapshots criptografados ... 900
Interromper o compartilhamento de snapshots ... 904
Exportar dados do cluster de banco de dados para o Amazon S3 ... 906
Considerações ... 908
Configurar o acesso a um bucket do S3 ... 910

Criar tarefas de exportação do cluster de banco de dados ... 914
Monitorar exportações do cluster de banco de dados ... 918
Cancelar uma exportação de cluster de banco de dados ... 920
Solução de problemas ... 921
Exportar dados de snapshot de cluster de banco de dados para o Amazon S3 ... 924
Considerações ... 926
Configurar o acesso a um bucket do S3 ... 939
Criar tarefas de exportação de snapshot ... 945
Monitorar exportações de snapshots ... 949
Cancelar uma exportação de snapshot ... 951
Performance de exportação no Aurora MySQL ... 952
Solução de problemas ... 953
Recuperação para um ponto no tempo ... 956
Restaurar um cluster de banco de dados para um ponto no tempo ... 957
Recuperação para um ponto no tempo de um backup automatizado retido ... 960
Recuperação para um ponto no tempo usando o AWS Backup ... 963
Excluir um snapshot de cluster de banco de dados ... 969
Excluir um snapshot de cluster de banco de dados ... 969
Tutorial: Restaurar um cluster de banco de dados de um snapshot ... 971
Restaurar um cluster de banco de dados usando o console ... 972
Restaurar um cluster de banco de dados usando a AWS CLI ... 976
Monitorar métricas em um cluster de banco de dados Aurora ... 983
Plano de monitoramento ... 983
Linha de base de performance ... 984
Orientações de performance ... 984
Ferramentas de monitoramento ... 986
Ferramentas de monitoramento automatizadas ... 986
Ferramentas de monitoramento manual ... 988
Visualizar o status da instância ... 990
Visualizar um cluster de banco de dados ... 991
Visualizar o status do cluster do banco de dados ... 997
Visualizar o status de uma instância de banco de dados em um cluster do Aurora ... 1002
Recomendações do Amazon Aurora ... 1010
Visualização de recomendações ... 1012
Aplicar recomendações ... 1020
Ignorar recomendações ... 1026

Modificar recomendações ignoradas para ativas 1028
Referência de recomendações 1029
Visualizar métricas no console do Amazon RDS 1055
Visualizar o painel do Insights de Performance 1056
Escolher a nova visualização de monitoramento na guia Monitoramento 1057
Escolher a nova visualização de monitoramento na página do Insights de Performance 1059
Criar um painel personalizado 1061
Escolher o painel predefinido 1064
Monitorando Aurora com CloudWatch 1066
Visão geral do Amazon Aurora e do Amazon CloudWatch 1067
Visualizar métricas do CloudWatch 1069
Exportar as métricas do Performance Insights para o CloudWatch 1075
Criar alarmes do CloudWatch 1081
Monitoramento com o Database Insights 1083
Preços 1083
Suporte a mecanismo, região e classe de instância 1083
Ativação do modo Avançado 1087
Ativação do modo Standard 1092
Monitorar consultas lentas 1097
Considerações 1099
Monitoramento de carga de banco de dados com o Performance Insights 1100
Visão geral do Performance Insights 1101
Ativar e desativar o Performance Insights 1112
Performance Schema para o Aurora MySQL 1119
Políticas do Performance Insights 1125
Análise de métricas usando o painel do Performance Insights 1139
Visualizar as recomendações proativas do Performance Insights 1174
Recuperar métricas com a API do Performance Insights 1177
Registrar em log as chamadas do Performance Insights usando o AWS CloudTrail 1202
Endpoints da VPC (AWS PrivateLink) 1205
Analisar a performance com o DevOps Guru para RDS 1209
Benefícios do DevOps Guru para RDS 1210
Como funciona o DevOps Guru para RDS 1211
Configurar o DevOps Guru para RDS 1212
Monitorar o SO com o monitoramento avançado 1221
Visão geral do monitoramento avançado 1221

Configurar e habilitar o monitoramento avançado 1223
Como visualizar métricas do SO no console do RDS 1231
Visualizar métricas do SO usando CloudWatch Logs 1233
Referência de métricas do Aurora 1234
Métricas do CloudWatch para Aurora 1234
Dimensões do CloudWatch para o Aurora 1278
Disponibilidade de métricas Aurora no console Amazon RDS 1279
Métricas do CloudWatch para Performance Insights 1283
Métricas de contadores do Performance Insights 1286
Estatísticas SQL para Performance Insights 1321
Métricas do sistema operacional no monitoramento avançado 1330
Monitorar eventos, logs e transmissões de atividades de bancos de dados 1340
Visualizar logs, eventos e transmissões no console do Amazon RDS 1341
Monitorar eventos do Aurora 1346
Visão geral dos eventos para Aurora 1346
Visualizar eventos do Amazon RDS 1348
Trabalhar com a notificação de eventos do Amazon RDS 1352
Criar uma regra que é acionada em um evento do Amazon Aurora 1378
Categorias de eventos e mensagens de eventos para o Amazon RDS 1383
Monitorar logs do Aurora 1424
Como visualizar e listar arquivos de log do banco de dados 1424
Como baixar um arquivo de log de banco de dados 1426
Como observar um arquivo de log de banco de dados 1427
Publicação no CloudWatch Logs 1429
Leitura do conteúdo de arquivos de log usando REST 1432
Arquivos de log do banco de dados MySQL 1434
Arquivos de log do banco de dados do PostgreSQL 1445
Monitorar chamadas de API do Aurora no CloudTrail 1456
Integração do CloudTrail com o Amazon Aurora 1456
Entradas do arquivo de log do Amazon Aurora 1457
Monitorar o Aurora com o recurso Database Activity Streams 1462
Visão geral 1462
Pré-requisitos de rede do Aurora MySQL 1466
Iniciar um stream de atividade de banco de dados 1469
Obter o status do fluxo de atividades 1472
Interromper um fluxo de atividade de banco de dados 1473

Monitorar os fluxos de atividades ... 1475
Exemplos de políticas do IAM para fluxos de atividades ... 1514
Monitorar ameaças com o GuardDuty RDS Protection ... 1518
Como trabalhar com o Aurora MySQL ... 1520
Visão geral do Aurora MySQL ... 1521
Melhorias de performance do Amazon Aurora MySQL ... 1521
Aurora MySQL e dados espaciais ... 1522
Aurora MySQL versão 3 compatível com o MySQL 8.0 ... 1523
Aurora MySQL versão 2 compatível com o MySQL 5.7 ... 1559
Segurança com o Aurora MySQL ... 1562
Privilégios de usuário mestre com Aurora MySQL. ... 1564
Conexões do TLS ... 1564
Atualizar aplicações para novos certificados TLS ... 1573
Determinar se alguma aplicação está se conectando ao cluster de banco de dados do Aurora MySQL usando TLS ... 1574
Determinar se um cliente requer verificação de certificado para se conectar ... 1574
Atualizar o armazenamento confiável de aplicações ... 1575
Exemplo de código Java para estabelecer conexões TLS ... 1576
Usar a autenticação Kerberos para Aurora MySQL ... 1578
Visão geral da autenticação Kerberos para o Aurora MySQL ... 1579
Limitações ... 1580
Configurar a autenticação Kerberos para Aurora MySQL ... 1582
Conectar-se ao Aurora MySQL com a autenticação Kerberos ... 1593
Gerenciar um cluster de banco de dados em um domínio ... 1596
Migrar dados para o Aurora MySQL ... 1599
Migrar de um banco de dados MySQL externo para o Aurora MySQL ... 1604
Migrar de uma instância de banco de dados MySQL para o Aurora MySQL ... 1634
Como gerenciar o Aurora MySQL ... 1663
Gerenciar a performance e a escalabilidade do Amazon Aurora MySQL ... 1663
Retroceder um cluster de banco de dados ... 1675
Testar o Amazon Aurora MySQL usando consultas de injeção de falhas ... 1697
Alterar tabelas no Amazon Aurora usando a DDL rápida ... 1702
Exibir o status do volume para um cluster de banco de dados Aurora ... 1709
Ajustar o Aurora MySQL ... 1711
Conceitos essenciais para ajuste do Aurora MySQL ... 1711
Ajustar o Aurora MySQL com eventos de espera ... 1715

Ajustar o Aurora MySQL com estados de threads 1774
Ajustar o Aurora MySQL com insights proativos do Amazon DevOps Guru 1782
Consulta paralela do Aurora MySQL 1788
Visão geral de consultas paralelas 1788
Criação de um cluster de consulta paralela 1793
Habilitar e desabilitar a consulta paralela 1797
Otimizar a consulta paralela 1801
Verificar o uso da consulta paralela 1805
Monitoramento de consulta paralela 1810
Constructos do SQL para consulta paralela 1817
Auditoria avançada no Aurora MySQL 1840
Como habilitar a Auditoria avançada 1840
Visualizar logs de auditoria 1844
Detalhes do log de auditoria 1844
Replicação com o Aurora MySQL 1846
Réplicas do Aurora 1846
Opções de replicação 1848
Performance de replicação 1849
Zero-downtime restart (ZDR – Reinício com tempo de inatividade zero) 1850
Filtros de replicação 1852
Monitorar a replicação do 1860
Replicação entre regiões 1861
Replicação de logs binários (binlogs) 1879
Replicação baseada em GTID 1926
Encaminhamento de gravação local 1934
Habilitar o encaminhamento de gravação local 1935
Conferir se um cluster de banco de dados tem o encaminhamento de gravação habilitado 1937
Compatibilidade SQL e de aplicações com o encaminhamento de gravação 1938
Níveis de isolamento para encaminhamento de gravação 1940
Consistência de leituras 1941
Executar instruções de várias partes com o encaminhamento de gravação 1945
Transações com o encaminhamento de gravação 1945
Parâmetros de configuração para o encaminhamento de gravação 1946
Métricas do encaminhamento de gravação 1947
Identificar transações e consultas encaminhadas 1952
Integrar o Aurora MySQL com serviços da AWS 1955

Autorizar o Aurora MySQL a acessar produtos da AWS 1956
Carregar dados de arquivos de texto no Amazon S3 1976
Salvar dados em arquivos de texto no Amazon S3 1991
Invocar uma função do Lambda a partir do Aurora MySQL 2003
Publicar logs do Aurora MySQL no CloudWatch Logs 2015
Modo de laboratório do Aurora MySQL 2021
Recursos em modo de laboratório do Aurora 2021
Práticas recomendadas do Aurora MySQL 2023
Como determinar a qual instância de banco de dados você está conectado 2024
Práticas recomendadas para performance e de escalabilidade 2024
Práticas recomendadas para alta disponibilidade 2035
Recomendações para recursos do MySQL 2037
Avaliação do uso de instâncias de banco de dados para Aurora MySQL com métricas do Amazon CloudWatch 2044
Solucionar problemas de desempenho do Aurora MySQL 2047
Opções de monitoramento da AWS 2047
Motivos mais comuns para problemas de desempenho do banco de dados 2048
Solucionar problemas em workloads 2049
Registro em log do Aurora MySQL 2080
Solucionar de problemas de conexão de banco de dados 2083
Solucionar problemas de performance 2100
Referência do Aurora MySQL 2105
Parâmetros de configuração 2105
Variáveis de status globais 2170
Eventos de espera 2189
Estados de thread 2195
Níveis de isolamento 2200
Dicas 2207
Referência de procedimentos armazenados 2211
Tabelas information_schema 2261
Atualizações do Aurora MySQL 2270
Conferir os números de versão 2271
Suporte de longo prazo e versões beta 2273
Preparar-se para o fim da vida útil do Aurora MySQL versão 2 2275
Preparar para o fim da vida útil do Amazon Aurora versão 1 2281
Como atualizar os clusters de banco de dados de Amazon Aurora MySQL 2284

Atualizações e correções feitas no mecanismo de banco de dados do Amazon Aurora MySQL 2415
Trabalho com Aurora PostgreSQL 2416
Ambiente de visualização do banco de dados 2418
Tipos de classe de instância de banco de dados compatíveis 2418
Recursos não compatíveis no ambiente de pré-visualização 2419
Criar um cluster de banco de dados no ambiente de pré-visualização 2420
PostgreSQL versão 17 no Ambiente de Pré-visualização do Banco de Dados 2422
Segurança com o Aurora PostgreSQL 2423
Noções básicas de perfis e permissões do PostgreSQL 2424
Como proteger dados do Aurora PostgreSQL com SSL/TLS 2441
Mascaramento dinâmico do Aurora PostgreSQL 2455
Atualizar aplicações para novos certificados SSL/TLS 2491
Determinar se as aplicações estão se conectando a clusters de banco de dados PostgreSQL do Aurora usando SSL 2492
Determinar se um cliente requer verificação de certificado para se conectar 2493
Atualizar o armazenamento confiável de aplicações 2494
Usar conexões SSL/TLS para diferentes tipos de aplicações 2494
Usar a autenticação Kerberos 2496
Disponibilidade de região e versão 2497
Visão geral da autenticação Kerberos 2497
Configurando 2498
Gerenciar um cluster de banco de dados do Aurora PostgreSQL em um domínio do Active Directory 2514
Conectar-se com a autenticação Kerberos 2515
Usar grupos de segurança do AD para controle de acesso do Aurora PostgreSQL 2519
Migrar dados para o Aurora PostgreSQL 2532
Migrar uma instância de banco de dados do RDS for PostgreSQL usando um snapshot .. 2533
Como migrar uma instância de banco de dados do RDS for PostgreSQL usando uma réplica de leitura do Aurora 2541
Otimizar a performance de consultas no Aurora PostgreSQL 2556
Melhorar a performance das consultas com o Aurora Optimized Reads 2557
Otimizar subconsultas correlacionadas no Aurora PostgreSQL 2564
Melhorar o desempenho de consultas usando junção adaptável 2571
Usar o cache de plano compartilhado 2574
Trabalhar com tabelas não registradas em log no Aurora PostgreSQL 2576

Criar tabelas não registradas em log ... 2577
Manipular tabelas não registradas em log durante a migração ... 2577
Converter tabelas não registradas em log em tabelas registradas em log ... 2578
Tabelas não registradas em log e replicação lógica ... 2578
Como trabalhar com o autovacuum do PostgreSQL ... 2579
Alocar memória para autovacuum ... 2580
Reduzir a probabilidade de conclusão de IDs de transação ... 2581
Determinar se as tabelas no seu banco de dados precisam de vacuum ... 2582
Determinar quais tabelas são atualmente elegíveis para autovacuum ... 2584
Determinar se o autovacuum está em execução e por quanto tempo ... 2585
Realização de um congelamento manual de vacuum ... 2587
Reindexação de uma tabela quando o autovacuum está em execução ... 2589
Gerenciar o autovacuum com grandes índices ... 2590
Outros parâmetros que afetam o autovacuum ... 2594
Definir parâmetros de autovacuum em nível de tabela ... 2594
Registrar atividades do autovacuum e do vacuum em log ... 2595
Noções básicas sobre o comportamento do autovacuum com bancos de dados inválidos . 2596
Identificar bloqueadores de limpeza ... 2599
Gerenciar a contenção de TOAST OID ... 2623
Noções básicas sobre operações TOAST ... 2623
Identificar problemas de performance ... 2624
Recomendações ... 2626
Monitoramento ... 2627
Usar o Babelfish para Aurora PostgreSQL ... 2632
Limitações do Babelfish ... 2634
Noções básicas da arquitetura e da configuração do Babelfish ... 2635
Criar um cluster de banco de dados do Babelfish para Aurora PostgreSQL ... 2682
Migração de um banco de dados do SQL Server para o Babelfish ... 2693
Autenticação de banco de dados com o Babelfish para Aurora PostgreSQL ... 2704
Conectar-se a um cluster de banco de dados do Babelfish ... 2723
Trabalhar com o Babelfish ... 2736
Solução de problemas do Babelfish ... 2821
Desativar o Babelfish ... 2823
Gerenciar atualizações da versão do Babelfish ... 2824
Referência do Babelfish ... 2847
Performance e escalabilidade do Aurora PostgreSQL ... 2907

Dimensionar instâncias de bancos de dados Aurora PostgreSQL ... 2908
Conexões máximas ... 2909
Limites de armazenamento temporário ... 2910
Páginas grandes para o Aurora PostgreSQL ... 2915
Teste do Amazon Aurora PostgreSQL usando consultas de injeção de falhas ... 2916
Exibir o status do volume para um cluster de bancos de dados Aurora ... 2921
Como especificar o disco de RAM para o stats_temp_directory ... 2922
Gerenciar arquivos temporários com o PostgreSQL ... 2924
Ajustar com eventos de espera do Aurora PostgreSQL ... 2930
Conceitos essenciais para ajuste do Aurora PostgreSQL ... 2931
Eventos de espera do Aurora PostgreSQL ... 2937
Client:ClientRead ... 2939
Client:ClientWrite ... 2943
CPU ... 2945
IO:BufFileRead and IO:BufFileWrite ... 2951
IO:DataFileRead ... 2960
IO:XactSync ... 2971
IPC:DamRecordTxAck ... 2974
IPC: eventos de espera paralelos ... 2975
IPC:ProcArrayGroupUpdate ... 2982
Lock:advisory ... 2984
Lock:extend ... 2987
Lock:Relation ... 2990
Lock:transactionid ... 2997
Lock:tuple ... 3000
LWLock:buffer_content (BufferContent) ... 3005
LWLock:buffer_mapping ... 3009
LWLock:BufferIO (IPC:BufferIO) ... 3012
LWLock:lock_manager ... 3014
LWLock:MultiXact ... 3019
LWLock:pg_stat_statements ... 3023
Tempo limite:PgSleep ... 3027
Ajustar o Aurora PostgreSQL com insights proativos do Amazon DevOps Guru ... 3028
O banco de dados está inativo há muito tempo na conexão da transação ... 3028
Práticas recomendadas do Aurora PostgreSQL ... 3032

Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora PostgreSQL ... 3032
Diagnosticar a sobrecarga na tabela e no índice ... 3033
Gerenciar contagens altas de objeto no Amazon Aurora PostgreSQL ... 3037
Gerenciamento aprimorado de memória no Aurora PostgreSQL ... 3044
Failover rápido ... 3046
Recuperação rápida após failover ... 3058
Gerenciar a rotatividade de conexão ... 3065
Tratamento de conexões inativas no PostgreSQL ... 3074
Ajustar parâmetros de memória para o Aurora PostgreSQL ... 3077
Analisar o uso de recursos com métricas do CloudWatch ... 3086
Usar replicação lógica para upgrades de versão principal ... 3090
Gerenciar conversões personalizadas ... 3099
Práticas recomendadas para consultas paralelas no Aurora PostgreSQL ... 3103
Solução de problemas de armazenamento no Aurora PostgreSQL ... 3109
Replicação com o Aurora PostgreSQL ... 3110
Réplicas do Aurora ... 3111
Melhorar a disponibilidade de leitura das réplicas do Aurora ... 3112
Monitorar a replicação do ... 3114
Visão geral da replicação lógica ... 3114
Configurar a replicação lógica ... 3116
Desativar a replicação lógica ... 3118
Monitorar o cache de gravação simultânea e slots lógicos ... 3119
Exemplo: usar a replicação lógica ... 3122
Exemplo: replicação lógica usando o Aurora PostgreSQL e o AWS DMS ... 3124
Configurar a autenticação do IAM para conexões de replicação lógica ... 3126
Encaminhamento de gravação local no Aurora PostgreSQL ... 3129
Limitações e considerações referentes ao encaminhamento de gravação local no Aurora PostgreSQL ... 3131
Configurar o Aurora PostgreSQL para encaminhamento de gravação local ... 3132
Trabalhar com encaminhamento de gravação local para o Aurora PostgreSQL ... 3136
Monitorar o encaminhamento de gravação local no Aurora PostgreSQL ... 3140
Usar o Aurora PostgreSQL como base de conhecimento para o Amazon Bedrock ... 3146
Pré-requisitos ... 3147
Preparar o Aurora PostgreSQL para ser uma base de conhecimento ... 3148
Criação de uma Base de Conhecimento no console do Bedrock ... 3151

Criar rapidamente uma Base de Conhecimento para Amazon Bedrock do Aurora PostgreSQL ... 3153
Integração do Aurora PostgreSQL aos produtos da AWS ... 3156
Como importar dados do Amazon S3 para o Aurora PostgreSQL ... 3157
Exportação de dados do PostgreSQL para Amazon S3 ... 3177
Invocar uma função Lambda a partir do Aurora PostgreSQL ... 3195
Publicar logs do Aurora PostgreSQL no CloudWatch Logs ... 3211
Monitorar planos de execução de consultas e pico de memória do Aurora PostgreSQL ... 3224
Acessar planos de execução de consultas e pico de memória usando funções do Aurora ... 3225
Referência de parâmetros de planos de execução de consultas do Aurora PostgreSQL ... 3226
Gerenciar planos de execução de consultas do Aurora PostgreSQL ... 3230
Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL ... 3231
Práticas recomendadas para gerenciamento de planos de consultas do Aurora PostgreSQL ... 3240
Gerenciamento de planos de consultas ... 3242
Capturar planos de execução do Aurora PostgreSQL ... 3244
Usar planos gerenciados do Aurora PostgreSQL ... 3247
Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans ... 3252
Melhorar os planos ... 3253
Excluir planos ... 3257
Exportar e importar planos gerenciados ... 3259
Referência a parâmetros ... 3261
Referência de funções ... 3268
Referência da visualização apg_plan_mgmt.dba_plans ... 3279
Atributos avançados do gerenciamento de planos de consultas ... 3284
Trabalhar com extensões e invólucros de dados externos ... 3299
Usar o suporte de extensão delegado do Amazon Aurora para PostgreSQL ... 3300
Gerenciar objetos grandes com maior eficiência com o módulo lo ... 3314
Gerenciar dados espaciais com PostGIS ... 3318
Gerenciar partições com a extensão pg_partman ... 3328
Agendar manutenção com a extensão pg_cron ... 3334
Usar pgAudit para registrar a atividade do banco de dados ... 3344
Usar pglogical para sincronizar dados ... 3358
Wrappers de dados externos compatíveis no Amazon Aurora PostgreSQL ... 3372
Trabalhar com Trusted Language Extensions para PostgreSQL ... 3388
Terminologia ... 3389

Requisitos para usar o Trusted Language Extensions ... 3390
Configurar o Trusted Language Extensions ... 3393
Visão geral do Trusted Language Extensions ... 3397
Criar extensões TLE ... 3399
Descartar suas extensões TLE de um banco de dados ... 3404
Desinstalar o Trusted Language Extensions ... 3405
Usar ganchos do PostgreSQL com suas extensões TLE ... 3406
Referência de funções para Trusted Language Extensions ... 3413
Referência de ganchos para Trusted Language Extensions ... 3426
Referência do Aurora PostgreSQL ... 3430
Agrupamentos do Aurora PostgreSQL para EBCDIC e outras migrações de mainframe ... 3430
Agrupamentos permitidos no Aurora PostgreSQL ... 3432
Referência de funções do Aurora PostgreSQL ... 3433
Aurora PostgreSQL parameters ... 3491
Eventos de espera do Aurora PostgreSQL ... 3555
Atualizações do Aurora PostgreSQL ... 3585
Identificar as versões do Amazon Aurora PostgreSQL ... 3585
Versões do Aurora PostgreSQL ... 3587
Versões de extensão para o Aurora PostgreSQL ... 3588
Como atualizar os clusters de banco de dados de Amazon Aurora MySQL ... 3588
Usar um release de de suporte de longo prazo (LTS) ... 3617
Uso do Aurora PostgreSQL Limitless Database ... 3620
Arquitetura do Limitless Database ... 3621
Principais termos ... 3623
Tipos de tabela ... 3625
Faturamento ... 3625
Introdução ao Limitless Database ... 3627
Requisitos e considerações do Aurora PostgreSQL Limitless Database ... 3628
Requisitos ... 3628
Considerações ... 3629
Recursos não compatíveis ... 3632
Pré-requisitos para usar o Limitless Database ... 3633
Habilitação de operações de grupo de fragmentos de banco de dados ... 3633
Criação de um cluster de banco de dados que usa o Limitless Database ... 3634
Correlação da capacidade máxima com roteadores e fragmentos ... 3634
Criar um cluster de banco de dados ... 3638

Trabalho com grupos de fragmentos de banco de dados ... 3648
Conexão ao cluster de banco de dados do Aurora PostgreSQL Limitless Database ... 3649
Localização do número de roteadores e fragmentos em um grupo de fragmentos de banco de dados ... 3650
Descrição de grupos de fragmentos de banco de dados ... 3650
Reinicialização de um grupo de fragmentos de banco de dados ... 3651
Alteração da capacidade de um grupo de fragmentos de banco de dados ... 3652
Dividir um fragmento ... 3654
Adicionar um roteador ... 3660
Exclusão de um grupo de fragmentos de banco de dados ... 3664
Adição de um grupo de fragmentos de banco de dados a um cluster de banco de dados do Limitless Database existente ... 3665
Criação de tabelas do Limitless Database ... 3671
Criação de tabelas ilimitadas usando variáveis ... 3672
Converter tabelas padrão em tabelas ilimitadas ... 3677
Exemplos de esquemas ... 3681
Carregar dados no Limitless Database ... 3683
Usar o comando COPY com o Limitless Database ... 3685
Usar o utilitário de carregamento de dados do Limitless Database ... 3687
Consulta do Limitless Database ... 3716
Consultas de fragmento único ... 3717
Consultas distribuídas ... 3727
Rastreamento de consultas distribuído ... 3728
Deadlocks distribuídos ... 3737
Gerenciamento do Limitless Database ... 3740
Considerações sobre o tamanho do banco de dados e das tabelas ... 3740
Recuperar espaço de armazenamento por meio de limpeza ... 3741
Monitoramento do Limitless Database ... 3745
Monitorar o Limitless Database com o CloudWatch ... 3746
Monitorar o Limitless Database com o Database Insights ... 3751
Monitoramento do Limitless Database com o CloudWatch Logs ... 3764
Monitoramento do Limitless Database com Monitoramento aprimorado ... 3765
Monitoramento do Limitless Database com o Insights de Performance ... 3766
Monitoramento do Limitless Database com o GuardDuty RDS Protection ... 3774
Funções e visualizações do Limitless Database ... 3775
Eventos de espera do Limitless Database ... 3801

Usar funções para aumentar a eficiência ... 3814
Monitoramento de instâncias ... 3822
Visão geral ... 3822
Fundamentos da arquitetura de instâncias ... 3822
Monitorar o desempenho da instância ... 3823
Monitoramento de roteadores e solução de problemas ... 3825
Monitoramento de fragmentos e solução de problemas ... 3825
Limitações ... 3827
Fazer backup e restaurar o Limitless Database ... 3828
Fazer backup de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database ... 3828
Restaurar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database ... 3831
Utilitários de backup e restauração do PostgreSQL não são compatíveis ... 3834
Atualização do Limitless Database ... 3835
Atualização de clusters de banco de dados que usam o Aurora PostgreSQL Limitless Database ... 3835
Referência do Limitless Database ... 3837
Comandos de DDL ... 3837
Limitações e informações de DDL ... 3846
Comandos de DML ... 3886
Limitações e informações de DML ... 3889
Variáveis ... 3893
Parâmetros do cluster de banco de dados ... 3894
Usar o Aurora Global Database ... 3895
Visão geral do Aurora Global Database ... 3895
Vantagens do Amazon Aurora Global Database ... 3897
Disponibilidade de região e versão ... 3898
Limitações do Aurora Global Database ... 3898
Conceitos básicos do Aurora Global Database ... 3901
Requisitos de configuração ... 3902
Criar um banco de dados global ... 3904
Adicionar um cluster secundário ... 3918
Criar um cluster sem periféricos/sem interface gráfica em uma região secundária ... 3924
Criar um banco de dados por meio de um snapshot ... 3927
Gerenciar um banco de dados global Aurora ... 3928

Modificar um banco de dados global Aurora ... 3929
Modificando parâmetros de banco de dados globais ... 3931
Remover um cluster de um banco de dados global Aurora ... 3932
Excluir um banco de dados global Aurora ... 3935
Marcação para o Aurora Global Database ... 3937
Conectar-se ao Aurora Global Database ... 3942
Escolher o endpoint que atenda às necessidades da aplicação ... 3943
Visualizar os endpoints de um banco de dados global ... 3944
Considerações sobre o uso de endpoints do gravador globais ... 3948
Como usar o encaminhamento de gravação em um banco de dados global Aurora ... 3948
Como usar o encaminhamento de gravação no Aurora MySQL ... 3949
Usar o encaminhamento de gravação no Aurora PostgreSQL ... 3972
Transição ou failover no Aurora Global Database ... 3989
Planejar BCDR ... 3989
Realizar transições ... 3990
Recuperar de uma interrupção não planejada ... 3997
Gerenciar RPOs (Aurora PostgreSQL) ... 4008
Resiliência entre regiões para clusters secundários ... 4014
Monitorar um banco de dados global Aurora ... 4015
Monitorando um banco de dados do Aurora global com Performance Insights ... 4016
Monitorar bancos de dados globais Aurora com fluxos de atividades de banco de dados ... 4017
Monitorar bancos de dados globais baseados no Aurora MySQL ... 4017
Monitorar banco de dados globais baseados no Aurora PostgreSQL ... 4021
Usar implantações azul/verde para o Aurora Global Database ... 4024
Benefícios de usar implantações azul/verde com o Aurora Global Database ... 4025
Como implantações azul/verde funcionam com o Aurora Global Database ... 4025
Usar bancos de dados globais do Aurora com outros produtos da AWS ... 4028
Usar o Controlador de Recuperação de Aplicações (ARC) da Amazon com o Aurora Global Database ... 4030
Atualizar um Amazon Aurora Global Database ... 4030
Atualizações de versão principal ... 4030
Atualizações de versões secundárias ... 4031
Amazon RDS Proxy ... 4037
Disponibilidade de região e versão ... 4039
Cotas e limitações ... 4039
Limitações do MySQL ... 4041

Limitações do PostgreSQL 4042
Planejar onde usar o RDS Proxy 4043
Conceitos e terminologia do RDS Proxy 4045
Visão geral dos conceitos do RDS Proxy 4046
Agrupamento de conexões 4047
Segurança 4047
Failover 4050
Transações 4051
Conceitos básicos do RDS Proxy 4052
Configurar uma rede de proxy 4053
Configurar de credenciais de banco de dados 4058
Configurar a autenticação do IAM 4062
Criar um proxy 4069
Visualizar um proxy 4082
Conectar-se por meio do RDS Proxy 4084
Gerenciar um RDS Proxy 4088
Modificar um RDS Proxy 4088
Adicionar um usuário de banco de dados 4096
Migrar para a autenticação de ponta a ponta do IAM 4098
Considerações sobre a conexão do RDS Proxy 4101
Evitar a fixação de um RDS Proxy 4105
Excluir um RDS Proxy 4111
Como trabalhar com endpoints do RDS Proxy 4112
Visão geral dos endpoints de proxy 4113
Limitações de endpoints de proxy 4114
Uso de endpoints de leitor com clusters do Aurora 4114
Acessar bancos de dados Aurora entre VPCs 4119
Criação de um endpoint de proxy 4121
Visualização dos endpoints de proxy 4124
Modificação de um endpoint de proxy 4125
Exclusão de um endpoint de proxy 4126
Monitorar o proxy do RDS com o CloudWatch 4128
Trabalhar com eventos do RDS Proxy 4137
Eventos do RDS Proxy 4137
Exemplos do RDS Proxy 4140
Solução de problemas do RDS Proxy 4143

Verificar a conectividade para um proxy 4143
Problemas e soluções comuns 4145
Solução de problemas do RDS para MySQL 4147
Solução de problemas do RDS para PostgreSQL 4149
Usar o proxy do RDS com o AWS CloudFormation 4156
Usa o RDS Proxy com bancos de dados globais do Aurora 4157
Limitações do RDS Proxy com bancos de dados globais 4157
Como os endpoints do RDS Proxy funcionam com bancos de dados globais 4157
Integrações ETL zero 4159
Benefícios 4161
Principais conceitos 4161
Limitações 4162
Limitações gerais 4162
Limitações do Aurora MySQL 4164
Limitações do Aurora PostgreSQL 4164
Limitações do Amazon Redshift 4165
Amazon SageMaker AILimitações referentes a lakehouses do 4165
Cotas 4165
Regiões com suporte 4166
Conceitos básicos das integrações ETL zero 4166
Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado 4167
Etapa 2: Selecionar ou criar um cluster de banco de dados de origem 4168
Etapa 3a: criar um data warehouse de destino 4169
Configurar uma integração usando SDKs da AWS 4170
Etapa 3b: criar um catálogo do AWS Glue para uma Integração ETL zero do Amazon SageMaker Lakehouse 4180
Próximas etapas 4183
Criar Integrações ETL zero com o Amazon Redshift 4184
Pré-requisitos 4184
Permissões obrigatórias 4184
Criar integrações ETL zero 4187
Criptografar integrações 4192
Próximas etapas 4194
Criar Integrações ETL zero com um lakehouse do Amazon SageMaker 4194
Pré-requisitos 4195
Permissões obrigatórias 4195

Criar Integrações ETL zero com um lakehouse do Amazon SageMaker 4198
Criptografar integrações 4203
Próximas etapas 4205
Filtragem de dados para integrações ETL zero 4205
Formato de um filtro de dados 4206
Lógica de filtros 4209
Precedência do filtro 4209
Exemplos do Aurora MySQL 4210
Exemplos do Aurora PostgreSQL 4211
Adicionar filtros de dados 4212
Remover filtros de dados 4214
Adicionar e consultar dados 4214
Criar um banco de dados de destino 4215
Adicionar dados ao cluster de banco de dados de origem 4215
Consultar os dados do Aurora no Amazon Redshift 4216
Diferenças dos tipos de dados 4218
Operações de DDL para o Aurora PostgreSQL 4232
Visualizar e monitorar integrações ETL zero 4235
Visualizar integrações 4235
Monitorar usando tabelas do sistema 4237
Monitoramento com o EventBridge 4238
Modificar integrações ETL zero 4238
Excluir integrações ETL zero 4239
Solução de problemas em integrações ETL zero 4241
Não consigo criar uma integração ETL zero. 4241
Minha integração está travada em um estado de `Syncing`. 4242
Minhas tabelas não estão sendo replicadas para o Amazon Redshift. 4242
Uma ou mais das minhas tabelas do Amazon Redshift exigem ressincronização. 4243
Problemas de falha em Integrações ETL zero com um lakehouse do Amazon SageMaker AI 4246
As operações de DDL que não estão concluídas aparecem no Amazon Redshift 4247
Utilizar o Aurora Serverless v2 4248
Casos de uso do Aurora Serverless v2 4248
Converter workloads provisionadas 4251
Vantagens do Aurora Serverless v2 4251
Como funciona o Aurora Serverless v2 4253

Visão geral ... 4253
Configurações de cluster ... 4255
Capacidade ... 4256
Escalabilidade ... 4259
Alta disponibilidade ... 4262
Armazenamento ... 4263
Parâmetros de configuração ... 4263
Requisitos e limitações do Aurora Serverless v2 ... 4264
Disponibilidade de região e versão ... 4264
Clusters que usam o Aurora Serverless v2 devem ter um intervalo de capacidade especificado. ... 4265
Alguns recursos provisionados não são compatíveis com o Aurora Serverless v2 ... 4265
Alguns aspectos do Aurora Serverless v2 são diferentes do Aurora Serverless v1. ... 4266
Criar um cluster de banco de dados do Aurora Serverless v2 ... 4266
Configurações ... 4267
Criar um cluster de banco de dados do Aurora Serverless v2 ... 4268
Criar um gravador do Aurora Serverless v2 ... 4273
Gerenciar o Aurora Serverless v2 ... 4274
Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster ... 4275
Conferir o intervalo de capacidade do Aurora Serverless v2 ... 4282
Adicionar um leitor do Aurora Serverless v2 ... 4283
Conversão de provisionado em Aurora Serverless v2 ... 4285
Conversão de Aurora Serverless v2 em provisionado ... 4286
Pausar instâncias de banco de dados do Aurora Serverless v2 ... 4287
Escolher o nível de promoção para um leitor do Aurora Serverless v2 ... 4287
Usar TLS/SSL com o Aurora Serverless v2 ... 4289
Visualizar gravadores e leitores do Aurora Serverless v2 ... 4290
Registro em log para o Aurora Serverless v2 ... 4292
Performance e escalabilidade no Aurora Serverless v2 ... 4299
Escolher o intervalo de capacidade ... 4300
Trabalhar com grupos de parâmetros para o Aurora Serverless v2 ... 4314
Evitar erros de falta de memória ... 4320
Métricas importantes do CloudWatch ... 4321
Monitorar a performance do Aurora Serverless v2 com o Performance Insights ... 4326
Solução de problemas de capacidade do Aurora Serverless v2 ... 4327
Escalar para 0 ACUs com pausa e retomada automáticas ... 4330

Visão geral da pausa automática 4331
Pré-requisitos e limitações 4332
Ativar e desativar a pausa automática 4333
Como funciona a pausa automática 4337
Pausa automática e configurações do cluster do Aurora 4342
Monitorar a pausa automática 4346
Solucionar problemas da pausa automática 4349
Design da aplicação para pausa automática 4352
Migrar para o Aurora Serverless v2 4353
Usar o Aurora Serverless v2 com um cluster existente 4354
Alternar de um cluster provisionado 4357
Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1 4364
Atualizar a partir do Aurora Serverless v1 para o Aurora Serverless v2 4375
Migrar de um banco de dados on-premises para o Aurora Serverless v2 4378
Usar o Aurora Serverless v1 4379
Disponibilidade de regiões e versões do Aurora Serverless v1 4380
Vantagens do Aurora Serverless v1 4380
Casos de uso do Aurora Serverless v1 4381
Limitações do Aurora Serverless v1 4381
Requisitos de configuração do Aurora Serverless v1 4384
Usar TLS/SSL com o Aurora Serverless v1 4385
Pacotes de cifras compatíveis com clusters de banco de dados do Aurora Serverless v1 .. 4388
Como funciona o Aurora Serverless v1 4388
Aurora Serverless v1Arquitetura do 4389
Auto Scaling 4391
Ação de tempo limite 4392
Pausa e retomada 4394
Determinar max_connections 4394
Grupos de parâmetros 4397
Registro em log 4401
Manutenção 4404
Failover 4405
Snapshots do 4406
Criar um cluster de banco de dados do Aurora Serverless v1 4406
Restaurar um cluster de banco de dados do Aurora Serverless v1 4410
Modificar um cluster de banco de dados do Aurora Serverless v1 4414

Modificar a configuração de ajuste de escala 4415
Atualizar a versão principal 4417
Conversão de Aurora Serverless v1 em provisionado 4420
Considerações ao converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado 4422
Dimensionar manualmente a capacidade do cluster de banco de dados do Aurora Serverless v1 4424
Visualizar clusters de banco de dados do Aurora Serverless v1 4427
Monitorar clusters de banco de dados do Aurora Serverless v1 com o CloudWatch 4430
Excluir um cluster de banco de dados do Aurora Serverless v1 4431
Aurora Serverless v1Versões de mecanismos de banco de dados do e do Aurora 4435
Aurora MySQL Serverless 4435
Aurora PostgreSQL Serverless 4436
Atualizações automáticas da versão secundária 4436
Usar a API de dados do Amazon RDS 4437
Disponibilidade de região e versão para a API de dados do Amazon RDS 4438
Suporte a IPv6 4439
Suporte a endpoints IPv6 4439
Usar endpoints IPv6 4440
Usar AWS PrivateLink com o IPv6 4442
Considerações sobre a migração 4443
Solução de problemas de conectividade IPv6 4443
Limitações 4444
API de dados com o Aurora Serverless v2 em comparação com o Aurora Serverless v1 4445
Número máximo de solicitações por segundo 4445
Habilitar ou desabilitar a API de dados do Amazon RDS em um banco de dados existente 4445
Eventos do CloudTrail 4446
Suporte a várias declarações 4446
Solicitações simultâneas para o mesmo ID de transação 4447
Comportamento de BatchExecuteStatement 4448
Comportamento do ExecuteSQL 4449
Comportamento de ExecuteStatement 4449
Comportamento do parâmetro de esquema 4450
Autenticação e autorização 4450
Autorização baseada em tags 4452
Armazenar credenciais em um segredo do Secrets Manager 4454

Habilitar a API de dados ... 4455
Habilitar a API de dados do RDS ao criar um banco de dados ... 4455
Habilitar a API Data do RDS em um banco de dados existente ... 4457
Criar um endpoint da Amazon VPC (PrivateLink) ... 4460
Chamar a API de dados ... 4463
Referência de operações ... 4464
Chamar por meio da AWS CLI ... 4469
Chamar por meio de aplicações Python ... 4480
Chamar por meio de aplicações Java ... 4484
Controlar o comportamento de tempo limite ... 4488
Biblioteca de cliente Java ... 4490
Baixar a biblioteca cliente Java para API de dados ... 4490
Exemplos de biblioteca cliente Java ... 4491
Processar resultados de consulta em formato JSON ... 4493
Recuperar resultados de consulta no formato JSON ... 4494
Mapeamento de tipo de dados ... 4494
Solução de problemas ... 4495
Exemplos ... 4496
Solução de problemas com a API de dados ... 4501
Transação <transaction_ID> não encontrada ... 4501
O pacote para consulta é muito grande ... 4502
A resposta do banco de dados excedeu o limite de tamanho ... 4502
HttpEndpoint não está habilitado para o cluster <cluster_ID> ... 4502
DatabaseErrorException: a transação ainda está executando uma consulta ... 4503
Exceção de resultado não aceito ... 4503
Declarações múltiplas não são aceitas ... 4503
O parâmetro schema não é aceito ... 4504
Problemas de conectividade IPv6 ... 4504
Registrar em log chamadas da API de dados ... 4505
Trabalhar com informações da API Data no CloudTrail ... 4505
Incluir e excluir eventos da API de dados de uma trilha do CloudTrail ... 4506
Noções básicas sobre as entradas do arquivo de log da API Data ... 4509
Monitorar consultas da API de dados ... 4511
Como as consultas da API de dados do RDS são representadas no Insights de Performance ... 4511
Usar o Query Editor ... 4513

Disponibilidade do editor de consultas 4513
Autorizar acesso 4513
Executar consultas 4515
Referência da API DBQMS 4519
CreateFavoriteQuery 4520
CreateQueryHistory 4520
CreateTab 4520
DeleteFavoriteQueries 4520
DeleteQueryHistory 4520
DeleteTab 4520
DescribeFavoriteQueries 4520
DescribeQueryHistory 4521
DescribeTabs 4521
GetQueryString 4521
UpdateFavoriteQuery 4521
UpdateQueryHistory 4521
UpdateTab 4521
Usar o machine learning do Aurora 4522
Usar o machine learning do Aurora com o Aurora MySQL 4523
Requisitos para usar o machine learning do Aurora 4524
Disponibilidade de regiões e versões 4525
Recursos compatíveis e limitações 4526
Configurar o cluster do Aurora para o machine learning do Aurora 4527
Usar o Amazon Bedrock com o cluster de banco de dados do Aurora MySQL 4541
Usar o Amazon Comprehend com seu cluster de banco de dados do Aurora MySQL 4543
Uso do SageMaker AI com o cluster de banco de dados do Aurora MySQL 4546
Considerações sobre performance 4550
Monitoramento 4552
Usar o machine learning do Aurora com o Aurora PostgreSQL 4553
Requisitos para usar o machine learning do Aurora 4554
Recursos compatíveis e limitações 4555
Configurar o cluster de banco de dados do Aurora para usar machine learning do Aurora . 4556
Usar o Amazon Bedrock com o cluster de banco de dados do Aurora PostgreSQL 4569
Usar o Amazon Comprehend com o cluster de banco de dados do Aurora PostgreSQL 4571
Uso do SageMaker AI com o cluster de banco de dados do Aurora PostgreSQL 4573

Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado) ... 4578
Considerações sobre performance ... 4579
Monitoramento ... 4584
Exemplos de código ... 4586
Conceitos básicos ... 4587
Olá, Aurora ... 4588
Conheça os conceitos básicos ... 4598
Ações ... 4768
Cenários ... 4978
Criar uma API REST de biblioteca de empréstimos ... 4979
Crie um rastreador de itens de trabalho do Aurora Sem Servidor ... 4980
Melhores práticas do Aurora ... 4985
Diretrizes operacionais básicas do Amazon Aurora ... 4986
Recomendações de RAM para a instância de banco de dados ... 4986
Drivers de banco de dados da AWS ... 4987
Monitoramento do Amazon Aurora ... 4988
Como trabalhar com grupos de parâmetros de banco de dados e grupos de parâmetros de cluster de banco de dados ... 4988
Vídeo de práticas recomendadas do Amazon Aurora ... 4988
Executar uma prova de conceito do Aurora ... 4990
Visão geral de uma prova de conceito do Aurora ... 4990
1. Identifique seus objetivos ... 4991
2. Entenda as características de workloads ... 4992
3. Pratique com o console ou a CLI ... 4993
Praticar com o console ... 4993
Praticar com o AWS CLI ... 4994
4. Crie seu cluster do Aurora ... 4995
5. Configure seu esquema ... 4996
6. Importe seus dados ... 4997
7. Porte seu código SQL ... 4998
8. Especifique definições da configuração ... 4999
9. Conectar-se ao Aurora ... 5000
10. Execute sua workload ... 5001
11. Avalie a performance ... 5002
12. Exercite a alta disponibilidade do Aurora ... 5006

13. O que fazer em seguida 5008
Segurança 5010
Database authentication (Autenticação do banco de dados) 5012
Autenticação com senha 5013
Autenticação do banco de dados do IAM 5014
Autenticação de Kerberos 5014
Gerenciamento de senhas com Aurora e o Secrets Manager 5016
Disponibilidade de região e versão 5016
Limitações 5016
Visão geral 5017
Benefícios 5018
Permissões 5018
Impor o gerenciamento pelo Aurora 5019
Gerenciar senha para um cluster de banco de dados 5020
Alternar segredos para clusters de banco de dados 5025
Visualizar detalhes do segredo para clusters de banco de dados 5027
Proteção de dados 5030
Criptografia de dados 5031
Privacidade do tráfego entre redes 5063
Gerenciamento de identidade e acesso 5065
Público 5065
Autenticar com identidades 5066
Gerenciamento do acesso usando políticas 5069
Como o Amazon Aurora funciona com o IAM 5071
Exemplos de políticas baseadas em identidade 5078
Políticas gerenciadas pela AWS 5101
Atualizações da política 5108
Prevenção do problema "confused deputy" entre serviços 5119
Autenticação do banco de dados do IAM 5121
Solução de problemas 5172
Registro em log e monitoramento 5174
Validação de conformidade 5177
Resiliência 5178
Backup e restauração 5178
Replicação 5179
Failover 5179

Segurança da infraestrutura 5180
Grupos de segurança 5180
Public accessibility 5180
VPC endpoints (AWS PrivateLink) 5182
Considerações 1205
Disponibilidade 1205
Criar um VPC endpoint de interface 1206
Criar uma política de VPC endpoint 1206
Práticas recomendadas de segurança 5186
Controlar acesso com grupos de segurança 5187
Visão geral dos grupos de segurança de VPC 5187
Cenário de grupos de segurança 5188
Criar um grupo de segurança de VPC 5190
Associação a um cluster de banco de dados 5190
Privilégios da conta de usuário mestre 5190
Funções vinculadas ao serviço 5193
Permissões de função vinculada ao serviço do Amazon Aurora 5193
Permissões de perfis vinculados ao serviço para o Amazon RDS Beta 5196
Perfil vinculado ao serviço da Pré-visualização do Amazon RDS 5197
Uso do Amazon Aurora com a Amazon VPC 5199
Trabalhar com um cluster de banco de dados em uma VPC 5199
Cenários para acessar um cluster de banco de dados em uma VPC 5219
Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4) 5226
Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla) 5234
Cotas e restrições 5246
Cotas no Amazon Aurora 5246
Restrições de nomenclatura no Amazon Aurora 5252
Limites de tamanho do Amazon Aurora 5253
Solução de problemas 5255
Não é possível conectar-se à instância de banco de dados do 5255
Teste de conexão com a instância de banco de dados 5258
Solução de problemas de autenticação da conexão 5259
Problemas de segurança 5259
Mensagem de erro "Falha ao recuperar atributos da conta, certas funções do console podem ser prejudicadas." 5259

Redefinir a senha de proprietário da instância de banco de dados 5259
Interrupção ou reinicialização da instância de banco de dados 5260
Alterações de parâmetros que não entram em vigor 5261
Problemas de memória liberável do Aurora 5261
Problemas de replicação do Aurora MySQL 5263
Diagnosticar e resolver atrasos entre réplicas de leitura 5263
Diagnosticar e resolver uma falha de replicação de leitura do MySQL 5265
Erro de replicação interrompida 5267
A replicação da réplica de leitura falha ao inicializar a estrutura de metadados 5268
Referência da API do Amazon RDS 5269
Uso da API de consulta 5269
Parâmetros de consulta 5269
Autenticação de solicitação de consulta 5270
Solução de problemas de aplicações 5270
Recuperação de erros 5270
Dicas de solução de problemas 5271
Histórico do documento 5272
Glossário da AWS 5385

O que é o Amazon Aurora?

O Amazon Aurora (Aurora) é um mecanismo de banco de dados relacional gerenciado compatível com o MySQL e o PostgreSQL. Você já sabe como o MySQL e o PostgreSQL aliam a velocidade e a confiabilidade dos bancos de dados comerciais avançados com a simplicidade e a economia dos bancos de dados de código aberto. O código, as ferramentas e as aplicações que você usa atualmente em seus bancos de dados existentes do MySQL e do PostgreSQL podem ser usados com o Aurora. Com algumas cargas de trabalho, o Aurora pode oferecer até cinco vezes a taxa de processamento do MySQL e até três vezes a taxa de processamento do PostgreSQL, sem exigir alterações na maioria das aplicações existentes.

O Aurora inclui um subsistema de armazenamento de alta performance. Seus mecanismos de banco de dados compatíveis com o MySQL e o PostgreSQL são personalizados para tirar proveito do rápido armazenamento distribuído. O armazenamento subjacente aumenta automaticamente, conforme necessário. Um volume de cluster do Aurora pode aumentar até o tamanho máximo de 128 tebibytes (TiB). O Aurora também automatiza e padroniza a clusterização e a replicação de bancos de dados que, normalmente, são os aspectos mais desafiantes da configuração e da administração de bancos de dados.

O Aurora faz parte do serviço de banco de dados gerenciado Amazon Relational Database Service (Amazon RDS). O Amazon RDS facilita a configuração, operação e escala de um banco de dados relacional na nuvem. Se você não estiver familiarizado com o Amazon RDS, consulte o Guia do usuário do Amazon Relational Database Service. Para saber mais sobre a variedade de opções de bancos de dados disponíveis na Amazon Web Services, consulte Como escolher o banco de dados certo para sua organização na AWS.

Tópicos

- Modelo de responsabilidade compartilhada do Amazon RDS
- Como o Amazon Aurora funciona com o Amazon RDS
- Clusters de banco de dados do Amazon Aurora
- Versões do Amazon Aurora
- Regiões e zonas de disponibilidade
- Recursos compatíveis com o Amazon Aurora por Região da AWS e com o mecanismo de banco de dados do Aurora
- Conexões de endpoints do Amazon Aurora

- Classes de instâncias de banco de dados do Amazon Aurora
- Armazenamento do Amazon Aurora
- Confiabilidade do Amazon Aurora
- Segurança do Amazon Aurora
- Alta disponibilidade do Amazon Aurora
- Replicação com o Amazon Aurora
- Faturamento da instância de banco de dados para Aurora
- Amazon Aurora no nível gratuito da AWS

Modelo de responsabilidade compartilhada do Amazon RDS

O Amazon RDS é responsável por hospedar os componentes de software e a infraestrutura de instâncias de banco de dados e clusters de banco de dados. Você é responsável pelo ajuste das consultas, que é o processo de ajustar as consultas SQL para melhorar a performance. A performance de consulta é altamente dependente do design do banco de dados, do tamanho dos dados, da distribuição dos dados, da workload da aplicação e dos padrões de consulta, que podem variar muito. Monitoramento e ajuste são processos altamente individualizados que você controla para seus bancos de dados do RDS. É possível usar o recurso Insights de Performance do Amazon RDS e outras ferramentas para identificar consultas problemáticas.

Como o Amazon Aurora funciona com o Amazon RDS

Os seguintes tópicos ilustram a forma como o Amazon Aurora se relaciona com os mecanismos padrão MySQL e PostgreSQL disponíveis no Amazon RDS:

- Escolha o Aurora MySQL ou o Aurora PostgreSQL como opção de mecanismo de banco de dados durante a configuração de novos servidores de banco de dados por meio do Amazon RDS.
- O Aurora aproveita os recursos conhecidos de gerenciamento e administração do Amazon Relational Database Service (Amazon RDS). O Aurora usa a interface do Console de gerenciamento da AWS do Amazon RDS, comandos da AWS CLI e operações de API para processar as tarefas de rotina de bancos de dados, como provisionamento, aplicação de patches, backup, recuperação, detecção de falhas e reparo.
- As operações de gerenciamento do Aurora normalmente envolvem clusters inteiros de servidores de bancos de dados que são sincronizados por meio de replicação, em vez de instâncias de banco de dados individuais. O clustering, a replicação e a alocação de armazenamento automáticos

facilitam e tornam a configuração, operação e escalabilidade de suas maiores implantações de MySQL e PostgreSQL mais rentáveis.

- Você pode trazer os dados do Amazon RDS para MySQL e do Amazon RDS para PostgreSQL para o Aurora criando e restaurando snapshots ou configurando a replicação unidirecional. Você pode usar as ferramentas de migração do tipo botão para converter sua aplicações do RDS para MySQL e do RDS para PostgreSQL para o Aurora.

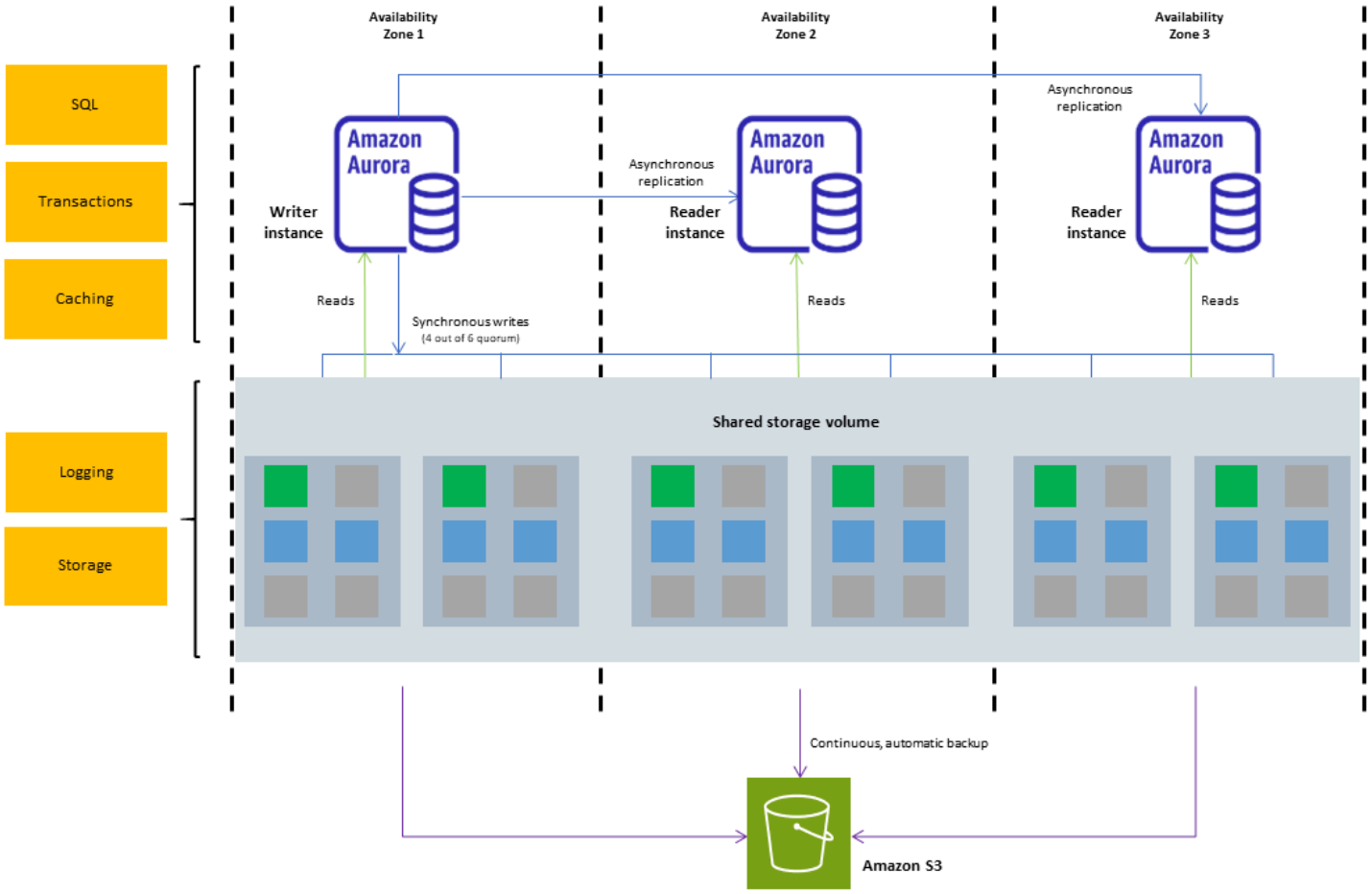
Antes de usar o Amazon Aurora, conclua as etapas em Configuração de seu ambiente para Amazon Aurora e revise os conceitos e os recursos do Aurora em Clusters de banco de dados do Amazon Aurora.

Clusters de banco de dados do Amazon Aurora

Um cluster de banco de dados do Amazon Aurora consiste em uma ou mais instâncias de banco de dados e em um volume de cluster que gerencia os dados para essas instâncias de banco de dados. Um volume de cluster do Aurora é um volume de armazenamento de banco de dados virtual que abrange várias zonas de disponibilidade, em que cada zona de disponibilidade conta com uma cópia dos dados do cluster de banco de dados. Um cluster de banco de dados do Aurora é composto por dois tipos de instâncias de banco de dados:

- Instância de banco de dados primária (gravador): comporta operações de leitura e de gravação, além de realizar todas as modificações de dados no volume do cluster. Cada cluster de banco de dados do Aurora tem uma instância de banco de dados primária.
- Réplica do Aurora (instância de banco de dados de leitor): conecta-se ao mesmo volume de armazenamento da instância de banco de dados primária, mas comporta apenas operações de leitura. Cada cluster de banco de dados do Aurora pode ter até 15 réplicas do Aurora, além da instância de banco de dados primária. Mantenha a alta disponibilidade localizando réplicas do Aurora em zonas de disponibilidade separadas. O Aurora faz o failover automaticamente para uma réplica do Aurora, caso a instância do banco de dados primário torne-se indisponível. Você pode especificar a prioridade de failover para réplicas do Aurora. As réplicas do Aurora também podem descarregar cargas de trabalho de leitura da instância de banco de dados primária.

O diagrama a seguir ilustra a relação entre o volume do cluster, a instância de banco de dados de gravador e as instâncias de banco de dados de leitor em um cluster de banco de dados do Aurora.



> **Note**
>
> As informações anteriores se aplicam a todos os clusters de banco de dados do Aurora: provisionados, de consulta paralela, do Aurora Global Database, do Aurora Serverless, compatíveis com o Aurora MySQL e com o Aurora PostgreSQL.

O cluster de banco de dados do Aurora ilustra a separação entre o armazenamento e a capacidade computacional. Por exemplo, uma configuração do Aurora com apenas uma única instância de banco de dados ainda é um cluster, pois o volume de armazenamento subjacente envolve vários nós de armazenamento distribuídos em diversas zonas de disponibilidade (AZs).

As operações de entrada e saída (E/S) nos clusters de banco de dados do Aurora são contadas da mesma forma, independentemente de estarem em uma instância de banco de dados de gravador ou leitor. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.

Versões do Amazon Aurora

Com o Amazon Aurora, é possível escolher o mecanismo de banco de dados relacional compatível que melhor atenda aos requisitos da aplicação, mantendo a compatibilidade com os mecanismos subjacentes. O Aurora reutiliza o código do mecanismo de banco de dados dos mecanismos compatíveis, para que você possa aproveitar as habilidades, ferramentas e bibliotecas existentes para esses mecanismos. Ao criar um cluster, você especifica a versão do mecanismo de banco de dados do Amazon Aurora que deseja usar. A versão escolhida determina a compatibilidade e os recursos disponíveis.

Esta documentação explica os pontos comuns e as diferenças entre o Amazon Aurora e os mecanismos de banco de dados correspondentes. Com essas informações, você pode determinar a versão do software que deve selecionar e como verificar os recursos disponíveis e as correções de bugs em cada versão. Você também pode usar esta referência para determinar uma cadência de atualização apropriada e planejar sua atualização.

Mecanismos de banco de dados compatíveis com clusters de banco de dados do Amazon Aurora

Os bancos de dados relacionais a seguir estão disponíveis no Amazon Aurora. O Aurora reutiliza o código e mantém a compatibilidade com os mecanismos de banco de dados subjacentes. Porém, o Aurora tem números de versão exclusivos, ciclos de lançamento e cronograma para defasagem de versão. Cada nova versão do Aurora vem com notas de lançamento que listam os novos recursos, correções e outras alterações e aprimoramentos que se aplicam a cada versão.

Banco de dados do Aurora	Guia do usuário	Versões disponíveis	Notas da versão
Amazon Aurora Edição Compatível com MySQL	Como trabalhar com o Amazon Aurora MySQL	Atualizações do mecanismo de banco de dados Amazon Aurora MySQL	Notas de lançamento do Aurora MySQL
Amazon Aurora Edição Compatível com PostgreSQL	Trabalho com Amazon Aurora PostgreSQL	Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL	Notas de lançamento do Aurora PostgreSQL

Especificar a versão do banco de dados do Amazon Aurora para seu cluster de banco de dados

Ao criar um cluster de banco de dados usando a operação Criar banco de dados no Console de gerenciamento da AWS ou na AWS CLI ou com a operação `CreateDBCluster` de API, você pode especificar qualquer versão de banco de dados do Aurora atualmente disponível (principal ou secundária).

Para saber como criar clusters do Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora. Para saber como alterar a versão de um cluster do Aurora existente, consulte Modificar um cluster de bancos de dados Amazon Aurora.

> **Note**
>
> Nem todas as versões do banco de dados do Aurora estão disponíveis em cada Região da AWS. Para saber mais sobre as regiões e as versões disponíveis em cada Região da AWS, consulte Regiões e zonas de disponibilidade e Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.

Versionamento do Amazon Aurora

As versões do Amazon Aurora são diferentes dos bancos de dados comunitários upstream com os quais são compatíveis. Para ajudar você a manter a compatibilidade das aplicações e aproveitar os recursos mais recentes do mecanismo de banco de dados, as seções a seguir explicam as convenções de versionamento do Aurora e como as versões do Aurora são mapeadas para os respectivos bancos de dados comunitários.

Para ver uma lista dos bancos de dados relacionais que estão disponíveis no Amazon Aurora, consulte Mecanismos de banco de dados compatíveis com clusters de banco de dados do Amazon Aurora.

Diferenças nos números de versão entre bancos de dados da comunidade e o Aurora

Cada versão do Amazon Aurora é compatível com uma versão específica do banco de dados da comunidade correspondente. Você pode encontrar a versão da comunidade de seu banco de dados com a função `version` e a versão do Aurora com a função `aurora_version`.

Os exemplos a seguir mostram como encontrar a versão comunitária do seu banco de dados para o Aurora MySQL e o Aurora PostgreSQL.

Aurora MySQL

A função `version` retorna a versão comunitária do seu banco de dados para o Aurora MySQL.

```
mysql> select version();
```

Exemplo de saída:

```
+------------------+
|   version()      |
+------------------+
|  8.0.32          |
+------------------+
```

E a função `aurora_version` retorna a versão do Aurora:

```
mysql> select aurora_version(), @@aurora_version;
```

Exemplo de saída:

```
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

Aurora PostgreSQL

A função `version` retorna a versão comunitária do seu banco de dados para o Aurora PostgreSQL.

```
postgres=> select version();
```

Exemplo de saída:

```
-----------------------------------------------------------------------------
PostgreSQL 11.7 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.9.3, 64-bit
(1 row)
```

E a função `aurora_version` retorna a versão do Aurora:

```
postgres=> select aurora_version();
```

Exemplo de saída:

```
aurora_version
---------------
3.2.2
```

Para obter mais informações, consulte Verificar versões do Aurora MySQL usando SQL e Identificar as versões do Amazon Aurora PostgreSQL.

Versões padrão do Amazon Aurora

A versão padrão é a versão que o Aurora escolhe automaticamente para a criação ou atualização do banco de dados quando você não especifica manualmente uma versão do mecanismo de destino. Por exemplo, o comando a seguir mostra a versão padrão do mecanismo para o Aurora PostgreSQL (exemplo de saída incluído).

```
aws rds describe-db-engine-versions \
    --engine aurora-postgresql \
    --default-only \
    --query 'DBEngineVersions[0].EngineVersion' \
    --output text

16.4
```

Cada versão principal tem uma versão secundária padrão correspondente. Assim, a versão secundária padrão é 16.n para o Aurora PostgreSQL 16, e o número da versão n muda quando o Aurora lança novas versões secundárias padrão. Normalmente, o Aurora lança duas versões secundárias padrão por ano para cada versão principal. O script de shell do Bash a seguir mostra as versões secundárias padrão de um conjunto de versões principais do Aurora PostgreSQL (exemplo de saída incluído).

```
for major in 16 15 14 13 12 11; do
  echo -n "Default for Aurora PostgreSQL major version $major: "
  aws rds describe-db-engine-versions \
    --engine aurora-postgresql \
    --engine-version "$major" \
```

```
    --default-only \
    --query 'DBEngineVersions[0].EngineVersion' \
    --output text
done

Default for Aurora PostgreSQL major version 16: 16.4
Default for Aurora PostgreSQL major version 15: 15.8
Default for Aurora PostgreSQL major version 14: 14.13
Default for Aurora PostgreSQL major version 13: 13.16
Default for Aurora PostgreSQL major version 12: 12.20
Default for Aurora PostgreSQL major version 11: 11.21
```

Se você habilitar atualizações automáticas de versões secundárias para o cluster de banco de dados do Aurora, o Aurora utilizará a versão secundária padrão ou uma versão secundária mais recente para determinada versão principal. Por exemplo, se a versão secundária padrão do Aurora PostgreSQL 15 for 15.8 e a versão mais recente 15.10 também estiver disponível, o Aurora poderá atualizar automaticamente para 15.8 ou 15.10.

Versões principais do Amazon Aurora

As versões do Aurora usam o esquema *major*.*minor*.*patch*. Uma versão principal do Aurora refere-se à versão principal da comunidade MySQL ou PostgreSQL com a qual o Aurora é compatível. As principais versões do Aurora MySQL e Aurora PostgreSQL estarão disponíveis por meio do suporte padrão pelo menos até o fim da vida útil da comunidade para a versão da comunidade correspondente. Você pode continuar executando uma versão principal após a data de término do suporte padrão do Aurora mediante o pagamento de uma taxa. Para ter mais informações, consulte Suporte estendido do Amazon RDS com Amazon Aurora e Preços do Amazon Aurora.

Para obter mais informações sobre as versões principais e o calendário de lançamento do Aurora MySQL e do Aurora PostgreSQL, consulte as seguintes páginas nas respectivas Notas de lançamento:

- Calendário de lançamento das versões principais do Aurora MySQL
- Calendário de lançamento das versões principais do Aurora PostgreSQL

Você também pode visualizar informações sobre datas de suporte para as versões principais do mecanismo executando o comando describe-db-major-engine-versionsAWS CLI ou usando a operação de API DescribeDBMajorEngineVersions do RDS.

> **Note**
>
> O Suporte estendido do Amazon RDS para o Aurora MySQL versão 2 inicia-se em 1.º de novembro de 2024, mas só será cobrado a partir de 1.º de dezembro de 2024. Entre 1.º e 30 de novembro de 2024, todos os clusters de banco de dados do Aurora MySQL versão 2 serão cobertos pelo Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do Amazon RDS para versões selecionadas do Aurora.

Por quanto tempo as versões principais do Amazon Aurora permanecem disponíveis

As principais versões do Amazon Aurora continuam disponíveis pelo menos até o fim da vida útil da comunidade para a versão da comunidade correspondente. É possível usar as mesmas datas de teste e atualização do Aurora para planejar seus ciclos de testes e atualização. Essas datas representam a data mais antiga em que uma atualização para uma versão mais recente pode ser necessária. Para obter mais informações sobre as datas, consulte Versões principais do Amazon Aurora.

Antes de o Aurora solicitar a atualização para uma versão principal mais recente e ajudar você a se planejar, você recebe um lembrete com pelo menos 12 meses de antecedência. Os lembretes comunicam o seguinte sobre o processo de atualização:

- O momento de determinados marcos
- O impacto nos seus clusters de banco de dados
- Ações recomendadas

Recomendamos testar detalhadamente suas aplicações com as novas versões do banco de dados antes de atualizar o cluster para uma nova versão principal.

Depois que a versão principal atingir o fim do suporte padrão do Aurora, qualquer cluster de banco de dados que ainda esteja executando a versão mais antiga será automaticamente atualizado para uma versão do Suporte estendido durante uma janela de manutenção programada. Cobranças do Suporte estendido são aplicáveis. Para ter mais informações sobre o Suporte estendido do Amazon RDS, consulte Usar o suporte estendido do Amazon RDS.

Versões secundárias do Amazon Aurora

As versões do Aurora usam o esquema *major*.*minor*.*patch*. Uma versão secundária do Aurora fornece melhorias incrementais da comunidade e específicas do Aurora para o serviço, por exemplo, novos recursos e correções.

Para obter mais informações sobre as versões secundárias e o calendário de lançamento do Aurora MySQL e do Aurora PostgreSQL, consulte as seguintes páginas nas respectivas Notas de lançamento:

- Calendário de lançamento das versões secundárias do Aurora MySQL
- Calendário de lançamento das versões secundárias do Aurora PostgreSQL

As seções a seguir descrevem detalhes sobre a cadência e a vida útil das versões secundárias do Aurora.

Tópicos

- Com que frequência são lançadas versões secundárias do Amazon Aurora
- Por quanto tempo as versões secundárias do Amazon Aurora permanecem disponíveis

Com que frequência são lançadas versões secundárias do Amazon Aurora

Em geral, as versões secundárias do Amazon Aurora são lançadas trimestralmente. O cronograma de lançamento pode variar para coletar mais recursos ou correções.

Por quanto tempo as versões secundárias do Amazon Aurora permanecem disponíveis

Normalmente, o Amazon Aurora disponibiliza cada versão secundária de uma versão principal específica durante pelo menos 12 meses. Ao final desse período, o Aurora poderá atualizar automaticamente seu banco de dados para a versão secundária padrão ou uma versão subsequente. O Aurora inicia a atualização durante a janela de manutenção programada para qualquer cluster de banco de dados que esteja executando a versão secundária mais antiga.

Em alguns casos, o Aurora pode substituir uma versão secundária de uma versão principal específica antes do período usual de 12 meses. Os motivos podem incluir problemas críticos de segurança ou término do prazo de suporte para uma versão principal.

Antes de iniciar atualizações automáticas de versões secundárias que estão se aproximando do fim da vida útil, geralmente o Aurora envia um lembrete com três meses de antecedência. O Aurora detalha o seguinte sobre o processo de atualização:

- O momento de determinados marcos
- O impacto nos seus clusters de banco de dados
- Ações recomendadas

As notificações com menos de três meses de antecedência descrevem questões críticas, como problemas de segurança, que exigem uma ação mais rápida.

Se a configuração Upgrade automático de versões secundárias estiver habilitada, você receberá um lembrete, mas nenhuma notificação de evento do RDS. Após o término do prazo obrigatório de atualização, o Aurora atualizará o banco de dados durante uma janela de manutenção.

Se a configuração Upgrade automático de versões secundárias não estiver habilitada, você receberá um lembrete e uma notificação de evento de cluster de banco de dados do Amazon RDS com categoria `maintenance` e ID `RDS-EVENT-0156`. O Aurora atualiza o banco de dados durante a próxima janela de manutenção.

Observe que, depois que uma versão secundária atingir o fim do suporte padrão do Aurora, nenhuma outra versão de patch será lançada para essa versão secundária. Para receber correções de erros críticos ou CVEs, você deve atualizar para uma versão secundária com suporte padrão.

Para ter mais informações sobre atualizações de versões secundárias, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.

Versões de patch do Amazon Aurora

As versões do Aurora usam o esquema *`major`*`.`*`minor`*`.`*`patch`*. Uma versão de patch do Aurora contém correções importantes adicionadas a uma versão secundária após seu lançamento inicial (por exemplo, Aurora MySQL 3.04.0, 3.04.1, ..., 3.04.3). Embora cada nova versão secundária forneça novos recursos do Aurora, as novas versões de patch contidas em uma versão secundária específica são usadas principalmente para resolver problemas importantes.

Para obter mais informações sobre aplicação de patches, consulte Manutenção de um cluster de banco de dados do Amazon Aurora.

Atualização dos clusters de banco de dados do Amazon Aurora

Com o Amazon Aurora, você pode controlar e testar atualizações em seus clusters de banco de dados. O Amazon Aurora oferece opções para atualizações automáticas de versões secundárias, controle manual de atualizações, atualizações obrigatórias e testes pré-atualização. Você pode manter seus clusters atualizados com a versão secundária mais recente, adiando atualizações não críticas, forçando atualizações em caso de problemas graves e validando o comportamento de atualização em ambientes não produtivos. As seções a seguir detalham como gerenciar e testar as atualizações do cluster de banco de dados do Aurora usando esses recursos.

Atualizações automáticas da versão secundária do Aurora

As atualizações automáticas de versões secundárias atualizam periodicamente o banco de dados para versões recentes do respectivo mecanismo. No entanto, a atualização nem sempre inclui a versão mais recente do mecanismo de banco de dados. Se precisar manter seus bancos de dados em determinadas versões em momentos específicos, recomendamos atualizar manualmente para as versões de banco de dados necessárias de acordo com o cronograma exigido. Em casos de problemas críticos de segurança ou quando uma versão atinge a data de fim do suporte, o Amazon Aurora pode aplicar uma atualização de versão secundária, mesmo que você não tenha habilitado a opção Upgrade automático de versões secundárias. Para ter mais informações, consulte a documentação sobre atualização do seu mecanismo de banco de dados específico.

Consulte Atualizando a versão secundária ou o nível de patch de um cluster de banco de dados de Aurora MySQL e Realizar uma atualização da versão secundária.

Você pode se manter atualizado com as versões secundárias do Aurora ativando Auto minor version upgrade (Atualização automática de versão secundária) para cada instância de banco de dados no cluster do Aurora. O Aurora só executará a atualização automática se todas as instâncias de banco de dados do cluster estiverem com essa configuração ativada.

Se Upgrade automático de versões secundárias estiver como Sim para o cluster de banco de dados, o Aurora atualizará automaticamente para a versão secundária padrão ou uma versão secundária mais recente. Por exemplo, se a versão secundária padrão for 15.8 para o Aurora PostgreSQL 15 e a versão 15.10 existir, o alvo para a atualização automática poderá ser a 15.8 ou 15.10.

Normalmente, programamos atualizações automáticas duas vezes por ano para clusters de banco de dados que têm a opção “Upgrade automático de versões secundárias” habilitada. Essas atualizações ocorrem durante a janela de manutenção especificada para o cluster. Para obter mais

informações, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.

As atualizações automáticas de versões secundárias são comunicadas com antecedência por meio de um evento de cluster de banco de dados do Amazon RDS com a categoria `maintenance` e o ID `RDS-EVENT-0156`. Para obter mais informações, consulte Categorias de eventos e mensagens de eventos para o Amazon RDS.

Controlar manualmente as atualizações de clusters de banco de dados para novas versões

Se você tiver a configuração Upgrade automático de versões secundárias habilitada, o Aurora atualizará automaticamente o cluster de banco de dados para a versão secundária padrão ou uma versão secundária mais recente. Normalmente, programamos atualizações automáticas duas vezes por ano para clusters de banco de dados que têm a definição Upgrade automático de versões secundárias habilitada. Essas atualizações são iniciadas durante as janelas de manutenção especificadas pelo cliente.

Para desativar atualizações automáticas de versões secundárias, desabilite Upgrade automático de versões secundárias em qualquer instância de banco de dados em um cluster do Aurora. O Aurora só executará um upgrade automático de versão secundária se todas as instâncias de banco de dados do cluster estiverem com a configuração habilitada.

> **Note**
>
> Com relação a atualizações obrigatórias, como o fim da vida útil da versão secundária, o Aurora atualizará o cluster de banco de dados mesmo se a configuração Upgrade automático de versões secundárias estiver desabilitada. Você recebe um lembrete, mas nenhuma notificação de evento do RDS. Após o término do prazo obrigatório de atualização, o Aurora atualizará o cluster durante uma janela de manutenção.

Como as atualizações de versões principais apresentam certo risco de compatibilidade, elas não ocorrem automaticamente. Você deve iniciá-las, exceto se houver descontinuação de uma versão principal. Recomendamos testar detalhadamente suas aplicações com as novas versões do banco de dados antes de atualizar o cluster para uma versão principal.

Para obter mais informações sobre como atualizar um cluster de banco de dados para uma nova versão principal do Aurora, consulte Como atualizar os clusters de banco de dados de Amazon Aurora MySQL e Como atualizar os clusters de banco de dados de Amazon Aurora MySQL.

Atualizações obrigatórias do Amazon Aurora

Com relação a determinadas correções críticas, o Aurora pode executar uma atualização gerenciada para um nível de patch na mesma versão secundária. Nesse caso, o Aurora atualizará o cluster mesmo se a configuração Upgrade automático de versões secundárias estiver desativada. Antes disso, o Aurora oferece informações detalhadas sobre o processo de atualização. Os detalhes incluem o momento de determinados marcos, o impacto nos clusters de banco de dados e as ações recomendadas. Essas atualizações gerenciadas ocorrem automaticamente durante a janela de manutenção do cluster.

Testar o cluster de banco de dados com uma nova versão do Aurora antes da atualização

É possível testar o processo de atualização e como a nova versão funciona com sua aplicação e workload. Use um dos seguintes métodos:

- Clone seu cluster com o recurso de clone rápido do banco de dados do Amazon Aurora. Execute a atualização e qualquer teste pós-atualização no novo cluster.
- Restaurar a partir de um snapshot do cluster para criar um novo cluster do Aurora. Você mesmo pode criar um snapshot do cluster em um cluster do Aurora existente. O Aurora também cria automaticamente snapshots periódicos por você para cada um dos seus clusters. Em seguida, é possível iniciar uma atualização de versão para o novo cluster. Você pode experimentar a cópia atualizada do cluster antes de decidir se deseja atualizar o cluster original.

Para obter mais informações sobre essas maneiras de criar novos clusters para teste, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora e Criar um snapshot de cluster de banco de dados.

Suporte para a versão do Amazon Aurora

Se o seu banco de dados do Amazon Aurora tiver dependências complexas em comportamentos específicos do mecanismo de banco de dados, recomendamos que realize testes extensivos antes de atualizar para versões mais recentes do mecanismo de banco de dados. Existem opções de

suporte de longo prazo para que você possa manter seus clusters de banco de dados em algumas versões do mecanismo do Aurora, mesmo depois que elas tenham sido substituídas por versões mais recentes. As seções a seguir explicam as opções de suporte de longo prazo para clusters de banco de dados do Aurora.

Suporte a longo prazo para versões secundárias selecionadas do Amazon Aurora

Para cada versão principal do Aurora, algumas versões secundárias são designadas como versões de suporte a longo prazo (LTS) e disponibilizadas por pelo menos três anos. Ou seja, pelo menos uma versão secundária por versão principal é disponibilizada por mais tempo do que os 12 meses habituais. Normalmente, o Aurora envia um lembrete seis meses antes do fim desse período. O Aurora comunica o seguinte sobre o processo de atualização:

- O momento de determinados marcos
- O impacto nos seus clusters de banco de dados
- Ações recomendadas

Notificações com menos de seis meses de antecedência comunicam questões críticas, como problemas de segurança, que exigem uma ação mais rápida.

As versões secundárias de LTS incluem apenas correções essenciais (por meio de versões de patch). Uma versão de LTS não contém novos recursos lançados após sua introdução. Uma vez por ano, aplicam-se patches nos clusters de banco de dados executados em uma versão secundária de LTS para a versão de patch mais recente de release de LTS. O Aurora aplica patches em seus clusters para que você se beneficie de correções cumulativas de segurança e estabilidade. O Aurora poderá corrigir uma versão secundária do LTS com maior frequência se houver correções críticas, como relativas à segurança, que precisem ser aplicadas.

> **Note**
>
> Se quiser permanecer em uma versão secundária de LTS durante o ciclo de vida, desative a atualização automática de versão secundária para as instâncias de banco de dados. Para não atualizar automaticamente o cluster de banco de dados da versão secundária de LTS, desmarque a caixa de seleção Habilitar a atualização automática da versão secundária em todas as instâncias de banco de dados do cluster do Aurora.

Para obter os números de versão de todas as versões do LTS do Aurora, consulte Versões de suporte de longo prazo (LTS) do Aurora MySQL e Usar uma versão de suporte de longo prazo (LTS) do Aurora PostgreSQL.

Suporte estendido do Amazon RDS para versões selecionadas do Aurora

Com o suporte estendido do Amazon RDS, é possível continuar executando o banco de dados em uma versão principal do mecanismo após a data de término do suporte padrão do Aurora por um custo adicional. Durante o Suporte estendido do RDS, o Amazon RDS fornecerá patches para CVEs graves e altos, conforme definido pelas classificações de gravidade CVSS do National Vulnerability Database (NVD). Para obter mais informações, consulte Suporte estendido do Amazon RDS com Amazon Aurora.

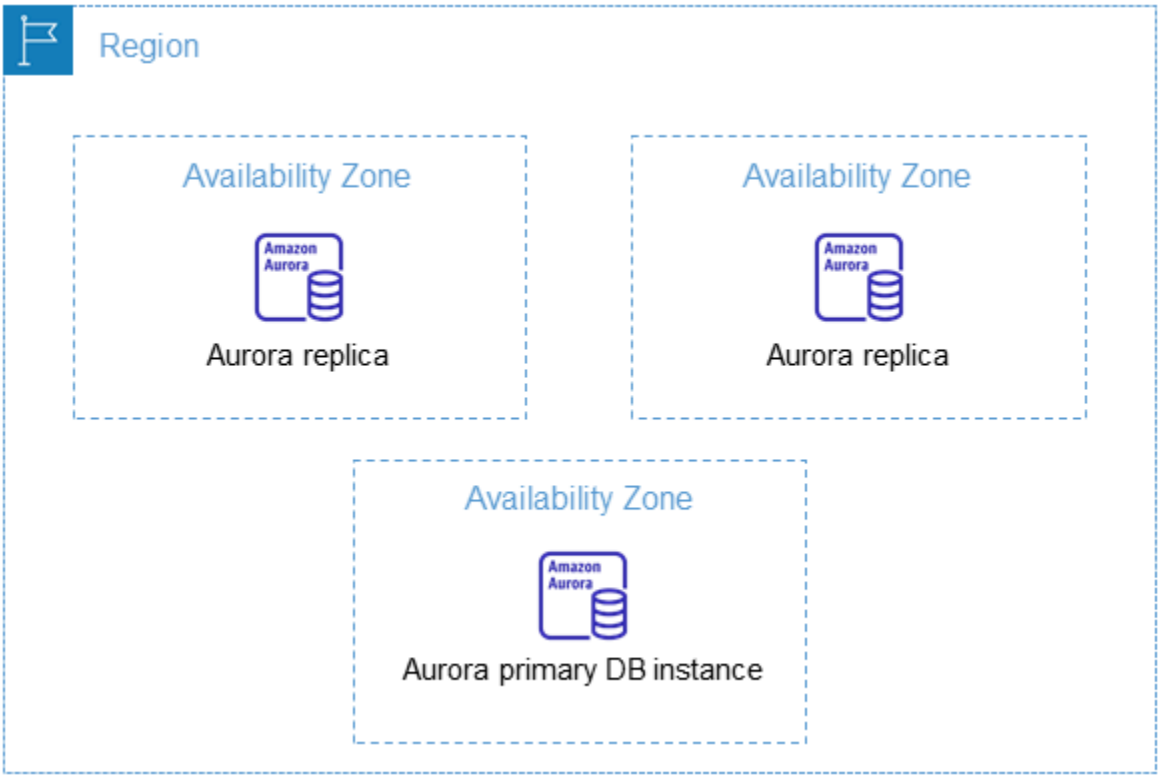
O Suporte estendido do RDS só está disponível em algumas versões do Aurora. Para obter mais informações, consulte Versões principais do Amazon Aurora.

Regiões e zonas de disponibilidade

Os recursos de computação em nuvem da Amazon são hospedados em vários locais no mundo todo. Esses locais são compostos por regiões da AWS e zonas de disponibilidade. Cada região da AWS é uma área geográfica separada. Cada região da AWS contém vários locais isolados conhecidos como zonas de disponibilidade.

> **Note**
>
> Para obter informações sobre como localizar as zonas de disponibilidade para uma região da AWS, consulte Como descrever zonas de disponibilidade na documentação do Amazon EC2.

A Amazon opera datacenters de última geração e altamente disponíveis. Embora sejam raras, podem ocorrer falhas que afetam a disponibilidade das instâncias de banco de dados que estiverem no mesmo local. Se você hospeda todas as instâncias de banco de dados em um único local afetado por essa falha, nenhuma delas estará disponível.



É importante lembrar que cada região AWS é totalmente independente. Qualquer atividade do Amazon RDS que você iniciar (por exemplo, criar instâncias de banco de dados ou listar

instâncias de banco de dados disponíveis) ocorre apenas em sua região da AWS padrão atual. A região da AWS padrão pode ser alterada no console ou definindo a variável de ambiente `AWS_DEFAULT_REGION`. Ou ele pode ser substituído usando o parâmetro `--region` pela AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Configurar a AWS Command Line Interface, especificamente as seções sobre variáveis de ambiente e opções de linha de comando.

O Amazon RDS é compatível com regiões especiais da AWSchamadas AWS GovCloud (US). Elas são projetadas para permitir que clientes e agências governamentais dos EUA transfiram workloads mais confidenciais para a nuvem. A região AWS GovCloud (US) atende aos seus requisitos específicos de regulamentação e conformidade do governo dos EUA. Para obter mais informações, consulte O que é o AWS GovCloud (US)?

Para criar ou trabalhar com uma instância de bancos de dados do Amazon RDS em uma região da AWS específica, use o endpoint de serviço regional correspondente.

> **Note**
>
> O Aurora não oferece suporte a zonas locais.

AWSRegiões de

Cada região da AWS é projetada para ser isolada das outras regiões da AWS. Esse design proporciona o máximo de tolerância a falhas e estabilidade possível.

Ao visualizar os recursos, você vê apenas os recursos que estão vinculados à região da AWS especificada. Isso ocorre porque as regiões da AWS são isoladas entre si e nós não replicamos os recursos entre regiões da AWS automaticamente.

Disponibilidade de regiões

Quando você trabalha com um cluster de bancos de dados Aurora usando a interface de linha de comando ou operações de API, certifique-se de especificar o endpoint regional.

Tópicos

- Disponibilidade de regiões do Aurora MySQL
- Disponibilidade de regiões do Aurora PostgreSQL

Disponibilidade de regiões do Aurora MySQL

A tabela a seguir mostra as regiões da AWS em que o Aurora MySQL está disponível no momento e o endpoint para cada região.

Nome da região	Região	Endpoint	Protocolo	
Leste dos EUA (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
Leste dos EUA (Norte da Virgínia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Oeste dos EUA (N. da Califórnia)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
Oeste dos EUA (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
África (Cidade do Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Ásia-Pacífico	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
(Hyderabad)				
Ásia-Pacífico (Jacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Ásia-Pacífico (Malásia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Ásia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Ásia-Pacífico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Ásia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Ásia-Pacífico (Seul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
Ásia-Pacífico (Singapura)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Tailândia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Ásia-Pacífico (Tóquio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canadá (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Oeste do Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
Europa (Milão)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
Europa (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
Europa (Espanha)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
Europa (Zurique)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
México (Central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Oriente Médio (Barém)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
Oriente Médio (Emirados Árabes Unidos)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS

Nome da região	Região	Endpoint	Protocolo	
América do Sul (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Leste dos EUA)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Disponibilidade de regiões do Aurora PostgreSQL

A tabela a seguir mostra as regiões da AWS em que o Aurora PostgreSQL está disponível no momento e o endpoint para cada região.

Nome da região	Região	Endpoint	Protocolo	
Leste dos EUA (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
Leste dos EUA (Norte da Virgínia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Oeste dos EUA (N. da	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
Califórnia)				
Oeste dos EUA (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
África (Cidade do Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Jacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Ásia-Pacífico (Malásia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Ásia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
Ásia-Pacífico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Ásia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Ásia-Pacífico (Seul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Singapura)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Tailândia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
Ásia-Pacífico (Tóquio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canadá (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Oeste do Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	
Europa (Milão)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	
Europa (Espanha)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
Europa (Zurique)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
México (Central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Oriente Médio (Barém)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
Oriente Médio (Emirados Árabes Unidos)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS
América do Sul (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (Leste dos EUA)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS

Zonas de disponibilidade

Uma zona de disponibilidade é um local isolado em determinada Região da AWS. Cada região tem várias zonas de disponibilidade (AZ) projetadas para fornecer alta disponibilidade. Uma AZ é

identificada pelo código de região da AWS seguido por um identificador alfabético (por exemplo, `us-east-1a`). Se você criar uma VPC e as sub-redes em vez de usar a VPC padrão, definirá cada sub-rede em uma AZ específica. Quando você cria um cluster de banco de dados do Aurora, o Aurora cria a instância principal em uma das sub-redes no grupo de sub-redes de banco de dados da VPC. Assim, ele associa essa instância a uma AZ específica selecionada pelo Aurora.

Cada cluster de banco de dados do Aurora hospeda cópias do armazenamento em três AZs distintas selecionadas automaticamente pelo Aurora por meio das AZs no grupo de sub-redes de banco de dados. Cada instância de banco de dados no cluster deve estar em uma dessas três AZs.

Ao criar uma instância de banco de dados no cluster, o Aurora escolherá automaticamente uma AZ apropriada para essa instância se você não especificar uma AZ.

Use o comando describe-availability-zones do Amazon EC2, como se segue, para descrever as zonas de disponibilidade da região especificada que estão disponíveis para sua conta.

```
aws ec2 describe-availability-zones --region region-name
```

Por exemplo, para descrever as zonas de disponibilidade da região Leste dos EUA (N. da Virgínia) (us-east-1) ativado para sua conta, execute o seguinte comando:

```
aws ec2 describe-availability-zones --region us-east-1
```

Para saber como especificar a AZ ao criar um cluster ou adicionar instâncias a ele, consulte Configurar a rede para o cluster de banco de dados.

Fuso horário local para os clusters de bancos de dados Amazon Aurora

Por padrão, o fuso horário de um cluster de bancos de dados Amazon Aurora é o Tempo Universal Coordenado (UTC). Você pode definir o fuso horário de instâncias em seu cluster de banco de dados como o fuso horário local de seu aplicativo.

Para definir o fuso horário local de um cluster de banco de dados, defina o parâmetro de fuso horário como um dos valores compatíveis. Você deve definir esse parâmetro no grupo de parâmetros do cluster de banco de dados.

- Para o Aurora MySQL, o nome desse parâmetro é `time_zone`. Para receber informações sobre as práticas recomendadas para definir o parâmetro `time_zone`, consulte Otimizando as operações de carimbo de data/hora.
- Para o Aurora PostgreSQL, o nome desse parâmetro é `timezone`.

Ao configurar o parâmetro de fuso horário para um cluster de banco de dados, todas as instâncias no cluster de banco de dados mudam para utilizar o novo fuso horário local. Em alguns casos, outros clusters de banco de dados do Aurora podem estar usando o mesmo grupo de parâmetros de cluster. Se for o caso, todas as instâncias nesses clusters de banco de dados também mudarão para usar o novo fuso horário local. Para obter informações sobre parâmetros no nível do cluster, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora.

Após definir o fuso horário local, todas as novas conexões ao banco de dados refletirão a alteração. Em alguns casos, você pode ter conexões abertas com o banco de dados ao alterar o fuso horário local. Se for o caso, você não verá a atualização do fuso horário local até que encerre a conexão e abra uma nova conexão.

Se você estiver replicando entre regiões da AWS, o cluster de banco de dados de origem da replicação e a réplica usam grupos de parâmetros diferentes. Os grupos de parâmetros são exclusivos de uma região AWS. Para utilizar o mesmo fuso horário local para cada instância, você deve definir o parâmetro de fuso horário nos grupos de parâmetros tanto para a origem da replicação quanto para a réplica.

Quando você restaura um cluster de banco de dados de um DB snapshot do cluster, o fuso horário local é definido como UTC. Você pode atualizar o fuso horário para o fuso horário local após a conclusão da restauração. Em alguns casos, você pode restaurar um cluster de banco de dados para um ponto no tempo. Se esse for o caso, o fuso horário local para o cluster de banco de dados restaurado será a configuração de fuso horário do grupo de parâmetros do cluster de banco de dados restaurado.

A tabela a seguir lista alguns dos valores para definir seu fuso horário local. Para listar todos os fusos horários disponíveis, use as seguintes consultas SQL:

- Aurora MySQL: `select * from mysql.time_zone_name;`
- Aurora PostgreSQL: `select * from pg_timezone_names;`

> **Note**
>
> Para alguns fusos horários, os valores de tempo de determinados intervalos de datas podem ter sido reportados incorretamente, conforme observado na tabela. Para o fuso horário da Austrália, a abreviatura do fuso horário retornado é um valor desatualizado, conforme observado na tabela.

Time zone (Fuso horário)	Observações
`Africa/Harare`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 21:49:40 GMT do dia 28 de fevereiro de 1903 até às 21:55:48 GMT do dia 28 de fevereiro de 1903.
`Africa/Monrovia`	
`Africa/Nairobi`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 21:30:00 GMT do dia 31 de dezembro de 1939 até às 21:15:15 GMT do dia 31 de dezembro de 1959.
`Africa/Windhoek`	
`America/Bogota`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 04:56:16 GMT do dia 23 de novembro de 1914 até às 04:56:20 GMT do dia 23 de novembro de 1914.
`America/Caracas`	
`America/Chihuahua`	
`America/Cuiaba`	
`America/Denver`	
`America/Fortaleza`	Em alguns casos, para um cluster de banco de dados na região da América do Sul (São Paulo), a hora não é exibida corretamente para um fuso horário do Brasil alterado recentemente. Nesse caso, redefina o parâmetro de fuso horário do cluster de banco de dados para `America/Fortaleza` .
`America/Guatemala`	

Time zone (Fuso horário)	Observações
America/Halifax	Esta configuração do fuso horário pode retornar valores incorretos a partir das 05:00:00 GMT do dia 27 de outubro de 1918 até às 05:00:00 GMT do dia 31 de outubro de 1918.
America/Manaus	Se o cluster de banco de dados estiver na região América do Sul (Cuiabá) e a hora esperada não for exibida corretamente para o fuso horário do Brasil alterado recentemente, redefina o parâmetro de fuso horário do cluster de banco de dados como America/Manaus .
America/Matamoros	
America/Monterrey	
America/Montevideo	
America/Noronha	
America/Phoenix	
America/Tijuana	
Asia/Ashgabat	
Asia/Baghdad	
Asia/Baku	
Asia/Bangkok	
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	

Time zone (Fuso horário)	Observações
`Asia/Karachi`	
`Asia/Kathmandu`	
`Asia/Muscat`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 20:05:36 GMT do dia 31 de dezembro de 1919 até às 20:05:40 GMT do dia 31 de dezembro de 1919.
`Asia/Riyadh`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 20:53:08 GMT do dia 13 de março de 1947 até às 20:53:08 GMT do dia 31 de dezembro de 1949.
`Asia/Seoul`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 15:30:00 GMT do dia 30 de novembro de 1904 até às 15:00:00 GMT do dia 07 de setembro de 1945.
`Asia/Shanghai`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 15:54:08 GMT do dia 31 de dezembro de 1927 até às 16:00:00 GMT do dia 02 de junho de 1940.
`Asia/Singapore`	
`Asia/Taipei`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 16:00:00 GMT do dia 30 de setembro de 1937 até às 15:00:00 GMT do dia 29 de setembro de 1979.
`Asia/Tehran`	
`Asia/Tokyo`	Esta configuração do fuso horário pode retornar valores incorretos a partir das 15:00:00 GMT do dia 30 de setembro de 1937 até às 15:00:00 GMT do dia 31 de dezembro de 1937.
`Asia/Ulaanbaatar`	

Time zone (Fuso horário)	Observações
Atlantic/Azores	Esta configuração do fuso horário pode retornar valores incorretos a partir das 01:54:32 GMT do dia 24 de maio de 1911 até às 01:54:32 GMT do dia 1º de janeiro de 1912.
Australia/Adelaide	A abreviação deste fuso horário é retornada como CST em vez de ACDT/ACST.
Australia/Brisbane	A abreviação deste fuso horário é retornada como EST em vez de AEDT/AEST.
Australia/Darwin	A abreviação deste fuso horário é retornada como CST em vez de ACDT/ACST.
Australia/Hobart	A abreviação deste fuso horário é retornada como EST em vez de AEDT/AEST.
Australia/Perth	A abreviação deste fuso horário é retornada como WST em vez de AWDT/AWST.
Australia/Sydney	A abreviação deste fuso horário é retornada como EST em vez de AEDT/AEST.
Brazil/East	
Canada/Saskatchewan	Esta configuração do fuso horário pode retornar valores incorretos a partir das 08:00:00 GMT do dia 27 de outubro de 1918 até às 08:00:00 GMT do dia 31 de outubro de 1918.
Europe/Amsterdam	
Europe/Athens	
Europe/Dublin	

Time zone (Fuso horário)	Observações
Europe/Helsinki	Esta configuração do fuso horário pode retornar valores incorretos a partir das 22:20:08 GMT do dia 30 de abril de 1921 até às 22:20:11 GMT do dia 30 de abril de 1921.
Europe/Paris	
Europe/Prague	
Europe/Sarajevo	
Pacific/Auckland	
Pacific/Guam	
Pacific/Honolulu	Esta configuração do fuso horário pode retornar valores incorretos a partir das 11:30:00 GMT do dia 21 de maio de 1933 até às 11:30:00 GMT do dia 30 de setembro de 1945.
Pacific/Samoa	Esta configuração do fuso horário pode retornar valores incorretos a partir das 11:22:48 GMT do dia 1º de janeiro de 1911 até às 01:30:00 GMT do dia 1º de janeiro de 1950.
US/Alaska	
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Recursos compatíveis com o Amazon Aurora por Região da AWS e com o mecanismo de banco de dados do Aurora

Aurora Os mecanismos de banco de dados compatíveis com MySQL e PostgreSQL suportam várias opções e recursos Amazon Aurora e Amazon RDS. A compatibilidade varia entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para identificar a compatibilidade e a disponibilidade das versões do mecanismo de banco de dados do Aurora para um recurso em determinada Região da AWS, você pode usar as seções a seguir.

Alguns desses recursos são recursos somente Aurora. Por exemplo, o Aurora Serverless, os bancos de dados globais do Aurora e o suporte para integração com os produtos de machine learning da AWS não são compatíveis com o Amazon RDS. Outros recursos, como o proxy do Amazon RDS, são compatíveis com o Amazon Aurora e o Amazon RDS.

Regiões e mecanismos de banco de dados compatíveis

- Convenções de tabela
- Regiões e mecanismos de banco de dados do Aurora compatíveis com implantações azul/verde
- Regiões e mecanismos de banco de dados do Aurora compatíveis com configurações de armazenamento em cluster
- Regiões e mecanismos de banco de dados do Aurora compatíveis com fluxos de atividades de banco de dados
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de cluster para o Amazon S3
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de snapshots para o Amazon S3
- Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a autenticação de banco de dados do IAM
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a autenticação Kerberos
- Regiões e mecanismos de banco de dados compatíveis com o machine learning do Aurora
- Regiões e mecanismos de banco de dados do Aurora compatíveis com o Insights de Performance
- Regiões e mecanismos de banco de dados do Aurora compatíveis com Integrações ETL zero
- Regiões e mecanismos de banco de dados do Aurora compatíveis com o Amazon RDS Proxy

- Regiões e mecanismos de banco de dados do Aurora compatíveis com a integração com o Secrets Manager
- Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2
- Aurora Serverless v1 da
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a API de dados do RDS
- Regiões e mecanismos de banco de dados do Aurora compatíveis com a aplicação de patches com tempo de inatividade zero (ZDP)
- Base de datos ilimitada de Aurora PostgreSQL de regiones admitidas
- Regiões e mecanismos de banco de dados compatíveis com recursos nativos do mecanismo do Aurora

Convenções de tabela

As tabelas nas seções de recursos usam os seguintes padrões para especificar números de versão e nível de suporte:

- Versão x.y: a versão específica por si só é compatível.
- Versão x.y e posteriores: a versão especificada e todas as versões secundárias posteriores da mesma versão principal são compatíveis. Por exemplo, “versão 10.11 e posterior” significa que as versões 10.11, 10.11.1 e 10.12 são compatíveis.

Regiões e mecanismos de banco de dados do Aurora compatíveis com implantações azul/verde

Uma implantação azul/verde copia um ambiente de banco de dados de produção em um ambiente de teste separado e sincronizado. Usando as implantações azul/verde do Amazon, você pode fazer alterações no banco de dados no ambiente de teste sem afetar o ambiente de produção. Por exemplo, você pode atualizar a versão principal ou secundária do mecanismo de banco de dados, alterar os parâmetros do banco de dados ou fazer alterações no esquema no ambiente de teste. Quando estiver pronto, você poderá promover o ambiente de teste como o novo ambiente de banco de dados de produção. Para obter mais informações, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Implantações azul/verde com o Aurora MySQL

O recurso de implantações azul/verde está disponível para todas as versões do Aurora MySQL em todas as Regiões da AWS, incluindo clusters do Aurora MySQL configurados como Aurora Global Database.

Implantações azul/verde com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para implantações azul/verde com o Aurora PostgreSQL, incluindo clusters do Aurora PostgreSQL configurados como Aurora Global Database.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Todas as Regiões da AWS	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.4 e posterior	Versão 14.9 e posterior	Versão 13.12 e posterior	Versão 12.16 e posterior	Versão 11.21 e posterior

Regiões e mecanismos de banco de dados do Aurora compatíveis com configurações de armazenamento em cluster

O Amazon Aurora tem duas configurações de armazenamento de cluster de banco de dados, Aurora I/O-Optimized e Aurora Standard. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.

Aurora I/O-Optimized

O Aurora I/O-Optimized está disponível em todas as Regiões da AWS para as seguintes versões do Amazon Aurora:

- Aurora MySQL versão 3.03.1 e posterior
- Aurora PostgreSQL versões 17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores

Aurora Standard

O Aurora Standard está disponível em todas as Regiões da AWS para todas as versões do Aurora MySQL e do Aurora PostgreSQL.

Regiões e mecanismos de banco de dados do Aurora compatíveis com fluxos de atividades de banco de dados

Ao usar fluxos de atividade de banco de dados no Aurora, você pode monitorar e definir alarmes para atividades de auditoria em seu banco de dados do Aurora. Para ter mais informações, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.

Fluxos de atividade de banco de dados não são compatíveis com os seguintes recursos:

- Aurora Serverless v1
- Aurora Serverless v2
- Babelfish for Aurora PostgreSQL

Tópicos

- Fluxos de atividade do banco de dados com o Aurora MySQL
- Fluxos de atividade do banco de dados com o Aurora PostgreSQL

Fluxos de atividade do banco de dados com o Aurora MySQL

As regiões e as versões do mecanismo a seguir estão disponíveis para fluxos de atividade de banco de dados com o Aurora MySQL.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Leste dos EUA (Ohio)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Oeste dos EUA (N. da Califórnia)	Todas as versões disponíveis	Aurora versão 2.11 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Oeste dos EUA (Oregon)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
África (Cidade do Cabo)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Hong Kong)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Hyderabad)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Jacarta)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Malásia)	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Não disponível	Não disponível
Ásia-Pacífico (Mumbai)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Seul)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Singapura)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Sydney)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Ásia-Pacífico (Taipei)	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Canadá (Central)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Oeste do Canadá (Calgary)	Não disponível	Não disponível
China (Pequim)	Não disponível	Não disponível
China (Ningxia)	Não disponível	Não disponível

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Frankfurt)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Irlanda)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Londres)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Milão)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Paris)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Espanha)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Estocolmo)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
Europa (Zurique)	Não disponível	Não disponível
Israel (Tel Aviv)	Não disponível	Não disponível
Oriente Médio (Bahrein)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
México (Centro)	Não disponível	Não disponível
Oriente Médio (Emirados Árabes Unidos)	Todas as versões disponíveis	Aurora versão 2.11 e posterior
América do Sul (São Paulo)	Todas as versões disponíveis	Aurora versão 2.11 e posterior

Fluxos de atividade do banco de dados com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para fluxos de atividade de banco de dados com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
África (Cidade do Cabo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
							posteriores
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Malásia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	
Ásia-Pacífico (Melbourne)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Taipei)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Oeste do Canadá (Calgary)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
China (Pequim)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
China (Ningxia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Europa (Milão)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Europa (Zurique)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Israel (Tel Aviv)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
México (Centro)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Todas as versões disponíveis	Todas as versões disponíveis	Todas as versões disponíveis	Versões 11.9 e 11.13 e posteriores

Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de cluster para o Amazon S3

É possível exportar dados de cluster de banco de dados do Aurora para um bucket do Amazon S3. Depois que os dados são exportados, você pode analisar os dados exportados diretamente por meio de ferramentas, como Amazon Athena ou Amazon Redshift Spectrum. Para obter mais informações, consulte Exportar dados do cluster de banco de dados para o Amazon S3.

A exportação dados do cluster para o S3 está disponível nas seguintes Regiões da AWS:

- Leste dos EUA (Norte da Virgínia)
- Leste dos EUA (Ohio)
- Oeste dos EUA (N. da Califórnia)
- Oeste dos EUA (Oregon)
- Ásia-Pacífico (Hong Kong)
- Ásia-Pacífico (Hyderabad)
- Ásia-Pacífico (Jacarta)
- Ásia-Pacífico (Melbourne)
- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Nova Zelândia)
- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)

- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Oeste do Canadá (Calgary)
- China (Ningxia)
- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Paris)
- Europa (Espanha)
- Europa (Estocolmo)
- Europa (Zurique)
- Israel (Tel Aviv)
- Oriente Médio (Emirados Árabes Unidos)
- América do Sul (São Paulo)

Tópicos

- Exportar dados do cluster para o S3 com o Aurora MySQL
- Exportar dados do cluster para o S3 com o Aurora PostgreSQL

Exportar dados do cluster para o S3 com o Aurora MySQL

Todas as versões do mecanismo do Aurora MySQL disponíveis no momento são compatíveis com a exportação de dados de cluster de banco de dados para o Amazon S3. Para obter mais informações sobre as versões, consulte as Release Notes for Aurora MySQL (Notas de versão do Aurora MySQL).

Exportar dados do cluster para o S3 com o Aurora PostgreSQL

Todas as versões do mecanismo do Aurora PostgreSQL disponíveis no momento são compatíveis com a exportação de dados de cluster de banco de dados para o Amazon S3. Para obter mais informações sobre as versões, consulte as Notas de versão do Aurora PostgreSQL.

Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de snapshots para o Amazon S3

É possível exportar dados de snapshots de cluster de banco de dados do Aurora para um bucket do Amazon S3. Você pode exportar snapshots manuais e snapshots automatizados do sistema. Depois que os dados são exportados, você pode analisar os dados exportados diretamente por meio de ferramentas, como Amazon Athena ou Amazon Redshift Spectrum. Para obter mais informações, consulte Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.

A exportação de snapshots para o S3 está disponível em todas as Regiões da AWS, exceto as seguintes:

- Ásia-Pacífico (Malásia)
- Ásia-Pacífico (Nova Zelândia)
- Ásia-Pacífico (Taipei)
- Ásia-Pacífico (Tailândia)
- México (Centro)

Tópicos

- Exportar dados de snapshot para o S3 com o Aurora MySQL
- Exportar dados de snapshot para o S3 com o Aurora PostgreSQL

Exportar dados de snapshot para o S3 com o Aurora MySQL

Todas as versões do mecanismo do Aurora MySQL disponíveis no momento são compatíveis com a exportação de dados de snapshot de cluster de banco de dados para o Amazon S3. Para obter mais informações sobre as versões, consulte as Release Notes for Aurora MySQL (Notas de versão do Aurora MySQL).

Exportar dados de snapshot para o S3 com o Aurora PostgreSQL

Todas as versões do mecanismo do Aurora PostgreSQL disponíveis no momento são compatíveis com a exportação de dados de snapshot de cluster de banco de dados para o Amazon S3. Para obter mais informações sobre as versões, consulte as Notas de versão do Aurora PostgreSQL.

Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora

Um banco de dados global do Aurora é um banco de dados único que abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e recuperação de desastres com interrupções de uma região inteira. Ele fornece tolerância a falhas incorporada para sua implantação porque a instância de banco de dados não depende de uma única Região da AWS, mas de várias regiões e zonas de disponibilidade diferentes. Para ter mais informações, consulte Usar o Amazon Aurora Global Database.

Tópicos

- Bancos de dados globais do Aurora com o Aurora MySQL
- Bancos de dados globais do Aurora com o Aurora PostgreSQL

Bancos de dados globais do Aurora com o Aurora MySQL

As regiões e as versões de mecanismo a seguir estão disponíveis para bancos de dados globais Aurora com o Aurora MySQL.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Leste dos EUA (Ohio)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Oeste dos EUA (Oregon)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
África (Cidade do Cabo)	Versão 3.01.0 e posterior	Versão 2.07.1 e posterior
Ásia-Pacífico (Hong Kong)	Versão 3.01.0 e posterior	Versão 2.07.1 e posterior
Ásia-Pacífico (Hyderabad)	Versão 3.02.0 e posterior	Versão 2.11.2 e posterior
Ásia-Pacífico (Jacarta)	Versão 3.01.0 e posterior	Versão 2.07.6 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Ásia-Pacífico (Malásia)	Versão 3.04.0 e posterior	Versão 2.07.6 e posterior
Ásia-Pacífico (Melbourne)	Versão 3.03.0 e posterior	Versão 2.07.6 e posterior
Ásia-Pacífico (Mumbai)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Nova Zelândia)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Osaka)	Versão 3.01.0 e posterior	Versão 2.07.3 e posterior
Ásia-Pacífico (Seul)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Singapura)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Sydney)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Taipei)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Tailândia)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Ásia-Pacífico (Tóquio)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Canadá (Central)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Oeste do Canadá (Calgary)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
China (Pequim)	Versão 3.01.0 e posterior	Versão 2.07.2 e posterior
China (Ningxia)	Versão 3.01.0 e posterior	Versão 2.07.2 e posterior
Europa (Frankfurt)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Europa (Irlanda)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Europa (Londres)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Europa (Milão)	Versão 3.01.0 e posterior	Versão 2.07.1 e posterior
Europa (Paris)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Espanha)	Versão 3.02.0 e posterior	Versão 2.07.6 e posterior
Europa (Estocolmo)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
Europa (Zurique)	Versão 3.02.0 e posterior	Versão 2.07.6 e posterior
Israel (Tel Aviv)	Versão 3.02.0 e posterior	Versão 2.07.6 e posterior
México (Centro)	Versão 3.02.0 e posterior	Versão 2.07.6 e posterior
Oriente Médio (Bahrein)	Versão 3.01.0 e posterior	Versão 2.07.1 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 3.02.0 e posterior	Versão 2.07.6 e posterior
América do Sul (São Paulo)	Versão 3.01.0 e posterior	Versão 2.07.1 e posterior
AWS GovCloud (Leste dos EUA)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 3.01.0 e posterior	Versão 2.07.0 e posterior

Bancos de dados globais do Aurora com o Aurora PostgreSQL

As regiões e as versões de mecanismo a seguir estão disponíveis para bancos de dados globais Aurora com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
África (Cidade do Cabo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Malásia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Melbourne)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Nova Zelândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Taipei)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Tailândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oeste do Canadá (Calgary)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
China (Pequim)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
China (Ningxia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milão)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Zurique)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Israel (Tel Aviv)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
México (Centro)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Leste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
AWS GovCloud (Oeste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Regiões e mecanismos de banco de dados do Aurora compatíveis com a autenticação de banco de dados do IAM

Com a autenticação de banco de dados do IAM no Aurora, você pode se autenticar em seu no cluster de banco de dados usando a autenticação de banco de dados do AWS Identity and Access Management (IAM). Com esse método de autenticação, você não precisa usar uma senha ao conectar-se a um cluster de banco de dados. Em vez disso, você usa um token de autenticação. Para ter mais informações, consulte Autenticação do banco de dados do IAM.

Tópicos

- Autenticação de banco de dados do IAM com o Aurora MySQL
- Autenticação de banco de dados do IAM com o Aurora PostgreSQL

Autenticação de banco de dados do IAM com o Aurora MySQL

A autenticação de banco de dados do IAM com o Aurora MySQL está disponível em todas as regiões para as seguintes versões:

- Aurora MySQL 3: todas as versões disponíveis
- Aurora MySQL 2: todas as versões disponíveis

Autenticação de banco de dados do IAM com o Aurora PostgreSQL

A autenticação de banco de dados do IAM com o Aurora PostgreSQL está disponível em todas as regiões para as seguintes versões de mecanismo:

- Aurora PostgreSQL 17: todas as versões disponíveis
- Aurora PostgreSQL 16: todas as versões disponíveis
- Aurora PostgreSQL 15: todas as versões disponíveis
- Aurora PostgreSQL 14: todas as versões disponíveis
- Aurora PostgreSQL 13: todas as versões disponíveis
- Aurora PostgreSQL 12: todas as versões disponíveis
- Aurora PostgreSQL 11: todas as versões disponíveis

Regiões e mecanismos de banco de dados do Aurora compatíveis com a autenticação Kerberos

Usando a autenticação do Kerberos com o Aurora, você pode dar suporte à autenticação externa de usuários de banco de dados usando Kerberos e Microsoft Active Directory. Usar o Kerberos e o Active Directory oferece os benefícios do logon único e da autenticação centralizada dos usuários do banco de dados. O Kerberos e o Active Directory estão disponíveis com AWS Directory Service for Microsoft Active Directory, um recurso do Directory Service. Para ter mais informações, consulte Autenticação de Kerberos.

Tópicos

- Autenticação do Kerberos com o Aurora MySQL
- Autenticação do Kerberos com o Aurora PostgreSQL
- Grupos de segurança do Active Directory (AD) com o Aurora PostgreSQL

Autenticação do Kerberos com o Aurora MySQL

As regiões e as versões do mecanismo a seguir estão disponíveis para a autenticação Kerberos com o Aurora MySQL.

Região	Aurora MySQL versão 3
Leste dos EUA (Norte da Virgínia)	Versão 3.03.0 e posterior
Leste dos EUA (Ohio)	Versão 3.03.0 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.03.0 e posterior
Oeste dos EUA (Oregon)	Versão 3.03.0 e posterior
África (Cidade do Cabo)	Indisponível
Ásia-Pacífico (Hong Kong)	Indisponível
Ásia-Pacífico (Jacarta)	Indisponível
Ásia-Pacífico (Malásia)	Indisponível
Ásia-Pacífico (Melbourne)	Indisponível
Ásia-Pacífico (Mumbai)	Versão 3.03.0 e posterior
Ásia-Pacífico (Nova Zelândia)	Indisponível
Ásia-Pacífico (Osaka)	Indisponível
Ásia-Pacífico (Seul)	Versão 3.03.0 e posterior
Ásia-Pacífico (Singapura)	Versão 3.03.0 e posterior
Ásia-Pacífico (Sydney)	Versão 3.03.0 e posterior
Ásia-Pacífico (Taipei)	Indisponível
Ásia-Pacífico (Tailândia)	Indisponível
Ásia-Pacífico (Tóquio)	Versão 3.03.0 e posterior
Canadá (Central)	Versão 3.03.0 e posterior
Oeste do Canadá (Calgary)	Indisponível

Região	Aurora MySQL versão 3
China (Pequim)	Versão 3.03.0 e posterior
China (Ningxia)	Versão 3.03.0 e posterior
Europa (Frankfurt)	Versão 3.03.0 e posterior
Europa (Irlanda)	Versão 3.03.0 e posterior
Europa (Londres)	Versão 3.03.0 e posterior
Europa (Milão)	Indisponível
Europa (Paris)	Versão 3.03.0 e posterior
Europa (Espanha)	Indisponível
Europa (Estocolmo)	Versão 3.03.0 e posterior
Europa (Zurique)	Indisponível
Israel (Tel Aviv)	Indisponível
México (Centro)	Indisponível
Oriente Médio (Bahrein)	Indisponível
Oriente Médio (Emirados Árabes Unidos)	Indisponível
América do Sul (São Paulo)	Versão 3.03.0 e posterior
AWS GovCloud (Leste dos EUA)	Versão 3.03.0 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 3.03.0 e posterior

Autenticação do Kerberos com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para a autenticação Kerberos com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Leste dos EUA (Ohio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (N. da Califórnia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (Oregon)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
África (Cidade do Cabo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Hong Kong)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Hyderabad)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Jacarta)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Malásia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Mumbai)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Seul)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Singapura)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Taipei)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Canadá (Central)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste do Canadá (Calgary)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
China (Pequim)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
China (Ningxia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Frankfurt)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Londres)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Milão)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Paris)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Espanha)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Estocolmo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Zurique)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Israel (Tel Aviv)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
México (Centro)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Bahrein)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Oriente Médio (Emirados Árabes Unidos)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
América do Sul (São Paulo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
AWS GovCloud (Leste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
AWS GovCloud (Oeste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Grupos de segurança do Active Directory (AD) com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para o Active Directory com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
(Norte da Virgínia)							
Leste dos EUA (Ohio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (N. da Califórnia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (Oregon)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
África (Cidade do Cabo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Hong Kong)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Hyderabad)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Jacarta)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Malásia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Mumbai)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Seul)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Singapura)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Taipei)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Canadá (Central)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste do Canadá (Calgary)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
China (Pequim)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
China (Ningxia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Frankfurt)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Irlanda)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Londres)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milão)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Paris)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Espanha)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Estocolmo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Zurique)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Israel (Tel Aviv)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
México (Centro)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Bahrein)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oriente Médio (Emirados Árabes Unidos)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
América do Sul (São Paulo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
AWS GovCloud (Leste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
AWS GovCloud (Oeste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Regiões e mecanismos de banco de dados compatíveis com o machine learning do Aurora

Ao usar o machine learning do Amazon Aurora, é possível integrar o cluster de banco de dados do Aurora a um dos serviços de machine learning da AWS a seguir, dependendo das necessidades. Eles oferecem compatibilidade com casos de uso de machine learning específicos.

O Amazon Bedrock é um serviço totalmente gerenciado que disponibiliza os principais modelos de base de empresas de IA por meio de uma API, junto com ferramentas para desenvolvedores para ajudar a criar e escalar aplicações de IA generativa.

O Amazon Comprehend é um serviço gerenciado de processamento de linguagem natural (PLN) utilizado para extrair insights de documentos. Ao usar o machine learning do Aurora com o Amazon Comprehend, você pode determinar o sentimento do texto nas tabelas de seu banco de dados.

O SageMaker AI é um serviço de machine learning com todos os recursos. Cientistas de dados usam o Amazon SageMaker AI para criar, treinar e testar modelos de machine learning para uma série de tarefas de inferência, como detecção de fraudes. Ao usar o machine learning do Aurora

com o SageMaker AI, os desenvolvedores de banco de dados podem invocar a funcionalidade do SageMaker AI no código SQL.

Nem todas as Regiões da AWS comportam todos os serviços de machine learning. Somente determinadas Regiões da AWS comportam o machine learning do Aurora e, portanto, concedem acesso a esses serviços em um cluster de banco de dados do Aurora. O processo de integração do machine learning do Aurora também difere de acordo com o mecanismo de banco de dados. Para ter mais informações, consulte Usar o machine learning do Amazon Aurora.

Tópicos

- Machine learning do Aurora com o MySQL
- Machine learning do Aurora com o Aurora PostgreSQL

Machine learning do Aurora com o MySQL

O Amazon Bedrock é compatível somente com o Aurora MySQL versão 3.06 e posterior. Consulte informações sobre a disponibilidade de regiões do Amazon Bedrock em Model support by Região da AWS no Guia do usuário do Amazon Bedrock.

O machine learning do Aurora com o Amazon Comprehend e o Amazon SageMaker AI é compatível com o Aurora MySQL nas Regiões da AWS listadas na tabela. Além de ter sua versão do Aurora MySQL disponível, a Região da AWS também deve ser compatível com o serviço que você deseja usar. Para obter uma lista das Regiões da AWS em que o Amazon SageMaker AI está disponível, consulte Endpoints e cotas do Amazon SageMaker AI na Referência geral da Amazon Web Services. Para ter uma lista das Regiões da AWS em que o Amazon Comprehend está disponível, consulte Amazon Comprehend endpoints and quotas na Referência geral da Amazon Web Services.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Leste dos EUA (Ohio)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Oeste dos EUA (Oregon)	Versão 3.01.0 e posterior	Versão 2.07 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
África (Cidade do Cabo)	Não disponível	Não disponível
Ásia-Pacífico (Hong Kong)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Hyderabad)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Jacarta)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Malásia)	Versão 3.04.0 e posterior	Indisponível
Ásia-Pacífico (Melbourne)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Mumbai)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Versão 3.01.0 e posterior	Versão 2.07.3 e posterior
Ásia-Pacífico (Seul)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Singapura)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Sydney)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Ásia-Pacífico (Taipei)	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Canadá (Central)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Oeste do Canadá (Calgary)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
China (Pequim)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
China (Ningxia)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Frankfurt)	Versão 3.01.0 e posterior	Versão 2.07 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Irlanda)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Londres)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Milão)	Não disponível	Não disponível
Europa (Paris)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Espanha)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Estocolmo)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Europa (Zurique)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Israel (Tel Aviv)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
México (Centro)	Não disponível	Não disponível
Oriente Médio (Bahrein)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
América do Sul (São Paulo)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
AWS GovCloud (Leste dos EUA)	Versão 3.01.0 e posterior	Versão 2.07 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 3.01.0 e posterior	Versão 2.07 e posterior

Machine learning do Aurora com o Aurora PostgreSQL

Para ter informações sobre a compatibilidade com versões do Amazon Bedrock no Aurora PostgreSQL, consulte Usar o Aurora PostgreSQL como base de conhecimento para o Amazon Bedrock.

O machine learning do Aurora com o Amazon Comprehend e o Amazon SageMaker AI é compatível com o Aurora PostgreSQL nas Regiões da AWS listadas na tabela. Além de ter sua versão do Aurora PostgreSQL disponível, a Região da AWS também deve ser compatível com o serviço que você deseja usar. Para obter uma lista das Regiões da AWS em que o Amazon SageMaker AI está disponível, consulte Endpoints e cotas do Amazon SageMaker AI na Referência geral da Amazon Web Services. Para ter uma lista das Regiões da AWS em que o Amazon Comprehend está disponível, consulte Amazon Comprehend endpoints and quotas na Referência geral da Amazon Web Services.

As regiões e as versões do mecanismo a seguir estão disponíveis para o machine learning do Aurora com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
África (Cidade do Cabo)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Malásia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Melbourne)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Nova Zelândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Taipei)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Ásia-Pacífico (Tailândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Oeste do Canadá (Calgary)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
China (Pequim)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
China (Ningxia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Milão)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Europa (Zurique)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Israel (Tel Aviv)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
México (Centro)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
AWS GovCloud (Leste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.3 e posterior	Versão 12.4 e posterior	Versão 11.9, 11.12 e posterior

Regiões e mecanismos de banco de dados do Aurora compatíveis com o Insights de Performance

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.

Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.

Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

O Performance Insights expande os recursos de monitoramento do Amazon RDS existentes para ilustrar e ajudar você a analisar a performance do banco de dados. Com o painel do Performance Insights, você pode visualizar a carga do banco de dados em sua carga de instâncias de banco de dados do Amazon RDS filtrá-la por esperas, declarações SQL, hosts ou usuários. Para obter mais informações, consulte Visão geral do Performance Insights no Amazon Aurora.

Para receber informações sobre a compatibilidade da região, do mecanismo de banco de dados e da classe de instância com os atributos do Insights de Performance, consulte Mecanismo de banco

de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance.

Tópicos

- Performance Insights com o Aurora MySQL
- Performance Insights com o Aurora PostgreSQL
- Performance Insights com o Aurora Serverless

Performance Insights com o Aurora MySQL

> **Note**
>
> O suporte à versão do mecanismo será diferente para o Performance Insights com o Aurora MySQL se você tiver a consulta paralela ativada. Para ter mais informações sobre consulta paralela, consulte Consulta paralela do Amazon Aurora MySQL.

Tópicos

- Performance Insights com Aurora MySQL e consulta paralela desativados
- Performance Insights com Aurora MySQL e consulta paralela ativados

Performance Insights com Aurora MySQL e consulta paralela desativados

As regiões e as versões do mecanismo a seguir estão disponíveis para o Performance Insights com Aurora MySQL e consulta paralela desativados.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Todas as versões	Todas as versões
Leste dos EUA (Ohio)	Todas as versões	Todas as versões
Oeste dos EUA (N. da Califórnia)	Todas as versões	Todas as versões
Oeste dos EUA (Oregon)	Todas as versões	Todas as versões

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
África (Cidade do Cabo)	Todas as versões	Todas as versões
Ásia-Pacífico (Hong Kong)	Todas as versões	Todas as versões
Ásia-Pacífico (Hyderabad)	Todas as versões	Todas as versões
Ásia-Pacífico (Jacarta)	Todas as versões	Todas as versões
Ásia-Pacífico (Malásia)	Todas as versões	Todas as versões
Ásia-Pacífico (Melbourne)	Todas as versões	Todas as versões
Ásia-Pacífico (Mumbai)	Todas as versões	Todas as versões
Ásia-Pacífico (Nova Zelândia)	Todas as versões	Todas as versões
Ásia-Pacífico (Osaka)	Todas as versões	Todas as versões
Ásia-Pacífico (Seul)	Todas as versões	Todas as versões
Ásia-Pacífico (Singapura)	Todas as versões	Todas as versões
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões
Ásia-Pacífico (Taipei)	Todas as versões	Todas as versões
Ásia-Pacífico (Tailândia)	Todas as versões	Todas as versões
Ásia-Pacífico (Tóquio)	Todas as versões	Todas as versões
Canadá (Central)	Todas as versões	Todas as versões
Oeste do Canadá (Calgary)	Todas as versões	Todas as versões
China (Pequim)	Todas as versões	Todas as versões
China (Ningxia)	Todas as versões	Todas as versões
Europa (Frankfurt)	Todas as versões	Todas as versões

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Irlanda)	Todas as versões	Todas as versões
Europa (Londres)	Todas as versões	Todas as versões
Europa (Milão)	Todas as versões	Todas as versões
Europa (Paris)	Todas as versões	Todas as versões
Europa (Espanha)	Todas as versões	Todas as versões
Europa (Estocolmo)	Todas as versões	Todas as versões
Europa (Zurique)	Todas as versões	Todas as versões
Israel (Tel Aviv)	Todas as versões	Todas as versões
México (Centro)	Todas as versões	Todas as versões
Oriente Médio (Bahrein)	Todas as versões	Todas as versões
Oriente Médio (Emirados Árabes Unidos)	Todas as versões	Todas as versões
América do Sul (São Paulo)	Todas as versões	Todas as versões
AWS GovCloud (Leste dos EUA)	Todas as versões	Todas as versões
AWS GovCloud (Oeste dos EUA)	Todas as versões	Todas as versões

Performance Insights com Aurora MySQL e consulta paralela ativados

As regiões e as versões do mecanismo a seguir estão disponíveis para o Performance Insights com Aurora MySQL e consulta paralela ativados.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Indisponível	Versão 2.09.0 e posterior
Leste dos EUA (Ohio)	Indisponível	Versão 2.09.0 e posterior
Oeste dos EUA (N. da Califórnia)	Indisponível	Versão 2.09.0 e posterior
Oeste dos EUA (Oregon)	Indisponível	Versão 2.09.0 e posterior
África (Cidade do Cabo)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Hong Kong)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Hyderabad)	Indisponível	Todas as versões
Ásia-Pacífico (Jacarta)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Malásia)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Melbourne)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Mumbai)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Nova Zelândia)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Osaka)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Seul)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Singapura)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Sydney)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Taipei)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Tailândia)	Indisponível	Versão 2.09.0 e posterior
Ásia-Pacífico (Tóquio)	Indisponível	Versão 2.09.0 e posterior

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Canadá (Central)	Indisponível	Versão 2.09.0 e posterior
Oeste do Canadá (Calgary)	Indisponível	Versão 2.09.0 e posterior
China (Pequim)	Indisponível	Versão 2.09.0 e posterior
China (Ningxia)	Indisponível	Versão 2.09.0 e posterior
Europa (Frankfurt)	Indisponível	Versão 2.09.0 e posterior
Europa (Irlanda)	Indisponível	Versão 2.09.0 e posterior
Europa (Londres)	Indisponível	Versão 2.09.0 e posterior
Europa (Milão)	Indisponível	Versão 2.09.0 e posterior
Europa (Paris)	Indisponível	Versão 2.09.0 e posterior
Europa (Espanha)	Indisponível	Versão 2.09.0 e posterior
Europa (Estocolmo)	Indisponível	Versão 2.09.0 e posterior
Europa (Zurique)	Indisponível	Versão 2.09.0 e posterior
Israel (Tel Aviv)	Indisponível	Versão 2.09.0 e posterior
México (Centro)	Indisponível	Versão 2.09.0 e posterior
Oriente Médio (Bahrein)	Indisponível	Versão 2.09.0 e posterior
Oriente Médio (Emirados Árabes Unidos)	Indisponível	Versão 2.09.0 e posterior
América do Sul (São Paulo)	Indisponível	Versão 2.09.0 e posterior
AWS GovCloud (Leste dos EUA)	Indisponível	Versão 2.09.0 e posterior
AWS GovCloud (Oeste dos EUA)	Indisponível	Versão 2.09.0 e posterior

Performance Insights com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para o Performance Insights com o Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Leste dos EUA (Norte da Virgínia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Leste dos EUA (Ohio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (N. da Califórnia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oeste dos EUA (Oregon)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
África (Cidade do Cabo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Ásia-Pacífico (Hong Kong)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Hyderabad)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Jacarta)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Malásia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Melbourne)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Mumbai)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Nova Zelândia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Ásia-Pacífico (Osaka)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Seul)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Singapura)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Taipei)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Tailândia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Ásia-Pacífico (Tóquio)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Canadá (Central)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Oeste do Canadá (Calgary)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
China (Pequim)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
China (Ningxia)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Frankfurt)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Irlanda)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Londres)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Milão)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Paris)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Europa (Espanha)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Estocolmo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Europa (Zurique)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Israel (Tel Aviv)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
México (Centro)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oriente Médio (Bahrein)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
Oriente Médio (Emirados Árabes Unidos)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
América do Sul (São Paulo)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
AWS GovCloud (Leste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões
AWS GovCloud (Oeste dos EUA)	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões	Todas as versões

Performance Insights com o Aurora Serverless

O Aurora Serverless v2 oferece suporte ao Performance Insights para todas as versões compatíveis com MySQL e PostgreSQL. É recomendável definir a capacidade mínima para pelo menos 2 unidades de capacidade do Aurora (ACUs).

Aurora Serverless v1O não é compatível com o Performance Insights.

Regiões e mecanismos de banco de dados do Aurora compatíveis com Integrações ETL zero

As Integrações ETL zero do Amazon Aurora são uma solução totalmente gerenciada para disponibilizar dados transacionais no Amazon Redshift ou no Amazon SageMaker depois que eles são gravados em um cluster do Aurora. Para obter mais informações, consulte Integrações ETL zero.

Tópicos

- Integrações ETL zero do Aurora MySQL
- Integrações ETL zero do Aurora PostgreSQL

Integrações ETL zero do Aurora MySQL

As regiões e versões de mecanismo a seguir estão disponíveis para Integrações ETL zero do Aurora MySQL com o Amazon Redshift e o Amazon SageMaker.

Região	Integração ETL zero com o Amazon Redshift para o Aurora MySQL versão 3	Integração ETL zero com o Amazon SageMaker para o Aurora MySQL versão 3
Leste dos EUA (Norte da Virgínia)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Leste dos EUA (Ohio)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.05.2 e posterior	Indisponível
Oeste dos EUA (Oregon)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
África (Cidade do Cabo)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Hong Kong)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Ásia-Pacífico (Hyderabad)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Jacarta)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Malásia)	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Mumbai)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Versão 3.05.2 e posterior	Indisponível
Ásia-Pacífico (Seul)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Ásia-Pacífico (Singapura)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior

Região	Integração ETL zero com o Amazon Redshift para o Aurora MySQL versão 3	Integração ETL zero com o Amazon SageMaker para o Aurora MySQL versão 3
Ásia-Pacífico (Sydney)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Ásia-Pacífico (Taipei)	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Canadá (Central)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Oeste do Canadá (Calgary)	Versão 3.05.2 e posterior	Indisponível
China (Pequim)	Versão 3.05.2 e posterior	Indisponível
China (Ningxia)	Versão 3.05.2 e posterior	Indisponível
Europa (Frankfurt)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Europa (Irlanda)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Europa (Londres)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Europa (Milão)	Versão 3.05.2 e posterior	Indisponível
Europa (Paris)	Versão 3.05.2 e posterior	Indisponível
Europa (Espanha)	Versão 3.05.2 e posterior	Indisponível
Europa (Estocolmo)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
Europa (Zurique)	Versão 3.05.2 e posterior	Indisponível
Israel (Tel Aviv)	Versão 3.05.2 e posterior	Indisponível
México (Centro)	Não disponível	Não disponível
Oriente Médio (Bahrein)	Versão 3.05.2 e posterior	Indisponível

Região	Integração ETL zero com o Amazon Redshift para o Aurora MySQL versão 3	Integração ETL zero com o Amazon SageMaker para o Aurora MySQL versão 3
Oriente Médio (Emirados Árabes Unidos)	Versão 3.05.2 e posterior	Indisponível
América do Sul (São Paulo)	Versão 3.05.2 e posterior	Versão 3.05.2 e posterior
AWS GovCloud (Leste dos EUA)	Não disponível	Não disponível
AWS GovCloud (Oeste dos EUA)	Não disponível	Não disponível

Integrações ETL zero do Aurora PostgreSQL

As regiões e as versões de mecanismo a seguir estão disponíveis para integrações ETL zero do Aurora PostgreSQL com o Amazon Redshift e o Amazon Sagemaker.

Região	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 16	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 16
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível

Região	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 16	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 16
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
África (Cidade do Cabo)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.4 e posterior	Não disponível	Não disponível
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Ásia-Pacífico (Melbourne)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Ásia-Pacífico (Malásia)	Não disponível	Não disponível		
Ásia-Pacífico (Malásia)	Não disponível	Não disponível		
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível		

Região	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 16	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 16
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Ásia-Pacífico (Taipei)	Não disponível	Não disponível		
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível		
Ásia-Pacífico (Taipei)	Não disponível	Não disponível		
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível		
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Canadá (Central)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior

Região	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 16	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 16
Oeste do Canadá (Calgary)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
China (Pequim)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
China (Ningxia)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Europa (Irlanda)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Europa (Londres)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
Europa (Milão)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Europa (Paris)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Europa (Espanha)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior

Região	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 17	Integração ETL zero com o Amazon Redshift para o Aurora PostgreSQL versão 16	Integração ETL zero com o Amazon SageMaker para o Aurora PostgreSQL versão 16
Europa (Zurique)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Israel (Tel Aviv)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
México (Centro)	Não disponível	Não disponível		
México (Centro)	Não disponível	Não disponível		
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Indisponível	Versão 16.4 e posterior	Indisponível
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 16.4 e posterior
AWS GovCloud (Leste dos EUA)	Não disponível	Não disponível	Não disponível	Não disponível
AWS GovCloud (Oeste dos EUA)	Não disponível	Não disponível	Não disponível	Não disponível

Regiões e mecanismos de banco de dados do Aurora compatíveis com o Amazon RDS Proxy

O proxy de Amazon RDS é um proxy de banco de dados totalmente gerenciado e altamente disponível que torna as aplicações mais escaláveis ao agrupar e compartilhar conexões de banco de dados estabelecidas. Para ter mais informações sobre o proxy do RDS, consulte Amazon RDS Proxypara Aurora.

Tópicos

- Proxy do Amazon RDS com Aurora MySQL
- Proxy do Amazon RDS com Aurora PostgreSQL

Proxy do Amazon RDS com Aurora MySQL

As regiões e as versões do mecanismo a seguir estão disponíveis para o RDS Proxy com Aurora MySQL.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Leste dos EUA (Ohio)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Oeste dos EUA (N. da Califórnia)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Oeste dos EUA (Oregon)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
África (Cidade do Cabo)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Hong Kong)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Ásia-Pacífico (Hyderabad)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Jacarta)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Malásia)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Melbourne)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Mumbai)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Seul)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Singapura)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Sydney)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Taipei)	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Ásia-Pacífico (Tóquio)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Canadá (Central)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Oeste do Canadá (Calgary)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
China (Pequim)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
China (Ningxia)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Frankfurt)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Irlanda)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Londres)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Milão)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Paris)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Espanha)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Estocolmo)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Europa (Zurique)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Israel (Tel Aviv)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
México (Centro)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Oriente Médio (Bahrein)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
Oriente Médio (Emirados Árabes Unidos)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
América do Sul (São Paulo)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
AWS GovCloud (Leste dos EUA)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores
AWS GovCloud (Oeste dos EUA)	Versão 3.01.0 e posterior	Versões 2.07 e 2.11 e posteriores

Proxy do Amazon RDS com Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para o RDS Proxy com Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
							posteriores
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
África (Cidade do Cabo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Malásia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Melbourne)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Taipei)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oeste do Canadá (Calgary)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
China (Pequim)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
China (Ningxia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Milão)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Europa (Zurique)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Israel (Tel Aviv)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
México (Centro)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
AWS GovCloud (Leste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores
AWS GovCloud (Oeste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.4 e posterior	Versão 12.8 e posterior	Versões 11.9 e 11.13 e posteriores

Regiões e mecanismos de banco de dados do Aurora compatíveis com a integração com o Secrets Manager

Com o AWS Secrets Manager, é possível substituir credenciais codificadas em seu código, inclusive senhas de banco de dados, por uma chamada de API ao Secrets Manager para recuperar o segredo de forma programática. Para obter mais informações sobre o Secrets Manager, consulte o Guia do usuário do AWS Secrets Manager.

Você pode especificar que o Amazon Aurora gerencie a senha de usuário mestre no Secrets Manager para um cluster de banco de dados do Aurora. O Aurora gera a senha, a armazena no Secrets Manager e a alterna regularmente. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.

A integração do Secrets Manager está disponível em todas as Regiões da AWS.

Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2

Aurora Serverless v2O é um recurso de escalabilidade automática sob demanda, projetado para ser uma abordagem econômica e executar workloads intermitentes ou imprevisíveis no Amazon Aurora. Ele ajusta a capacidade automaticamente de acordo com as necessidades de suas aplicações. A escalabilidade é mais rápida e mais granular quando comparada com o Aurora Serverless v1. Com o Aurora Serverless v2, cada cluster pode conter uma instância de banco de dados de gravador e várias instâncias de banco de dados de leitor. Você pode combinar o Aurora Serverless v2 e instâncias de banco de dados provisionadas tradicionais no mesmo cluster. Para ter mais informações, consulte Usar o Aurora Serverless v2.

Tópicos

- Aurora Serverless v2 com o Aurora MySQL
- Aurora Serverless v2 com o Aurora PostgreSQL

Aurora Serverless v2 com o Aurora MySQL

As regiões e as versões do mecanismo a seguir estão disponíveis para Aurora Serverless v2 com Aurora MySQL.

Região	Aurora MySQL versão 3
Leste dos EUA (Norte da Virgínia)	Versão 3.02.0 e posterior
Leste dos EUA (Ohio)	Versão 3.02.0 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.02.0 e posterior
Oeste dos EUA (Oregon)	Versão 3.02.0 e posterior

Região	Aurora MySQL versão 3
África (Cidade do Cabo)	Versão 3.02.0 e posterior
Ásia-Pacífico (Hong Kong)	Versão 3.02.0 e posterior
Ásia-Pacífico (Hyderabad)	Versão 3.02.3 e posterior
Ásia-Pacífico (Jacarta)	Versão 3.02.0 e posterior
Ásia-Pacífico (Malásia)	Versões 3.04.3, 3.05.2, 3.06.1, 3.07.1 e posterior
Ásia-Pacífico (Melbourne)	Versão 3.02.3 e posterior
Ásia-Pacífico (Mumbai)	Versão 3.02.0 e posterior
Ásia-Pacífico (Nova Zelândia)	Versões 3.04.3 e posterior, 3.08.0 e posterior
Ásia-Pacífico (Osaka)	Versão 3.02.0 e posterior
Ásia-Pacífico (Seul)	Versão 3.02.0 e posterior
Ásia-Pacífico (Singapura)	Versão 3.02.0 e posterior
Ásia-Pacífico (Sydney)	Versão 3.02.0 e posterior
Ásia-Pacífico (Taipei)	Versões 3.04.3 e posterior, 3.08.1 e posterior
Ásia-Pacífico (Tailândia)	Versões 3.04.3 e posterior, 3.08.0 e posterior
Ásia-Pacífico (Tóquio)	Versão 3.02.0 e posterior
Canadá (Central)	Versão 3.02.0 e posterior
Oeste do Canadá (Calgary)	Versão 3.04.0 e posterior
China (Pequim)	Versão 3.02.2 e posterior
China (Ningxia)	Versão 3.02.2 e posterior
Europa (Frankfurt)	Versão 3.02.0 e posterior

Região	Aurora MySQL versão 3
Europa (Irlanda)	Versão 3.02.0 e posterior
Europa (Londres)	Versão 3.02.0 e posterior
Europa (Milão)	Versão 3.02.0 e posterior
Europa (Paris)	Versão 3.02.0 e posterior
Europa (Espanha)	Versão 3.02.3 e posterior
Europa (Estocolmo)	Versão 3.02.0 e posterior
Europa (Zurique)	Versão 3.02.3 e posterior
Israel (Tel Aviv)	Versões 3.02.3 e posterior, 3.03.1 e posterior
México (Centro)	Versões 3.04.3 e posterior, 3.08.0 e posterior
Oriente Médio (Bahrein)	Versão 3.02.0 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 3.02.3 e posterior
América do Sul (São Paulo)	Versão 3.02.0 e posterior
AWS GovCloud (Leste dos EUA)	Versão 3.02.2 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 3.02.2 e posterior

Os limites de capacidade máximo e mínimo em ACU do Aurora Serverless v2 podem variar dependendo da versão do mecanismo. Para obter detalhes, consulte Aurora Serverless v2Capacidade do.

Aurora Serverless v2 com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para Aurora Serverless v2 com Aurora PostgreSQL.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
África (Cidade do Cabo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Hyderabad)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Malásia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.4 e posterior	Versões 14.6, 14.9 e posterior	Versão 13.9, 13.12 e posterior
Ásia-Pacífico (Melbourne)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Nova Zelândia)	Versão 17.4 e posterior	Versão 16.8 e posterior	Versão 15.12 e posterior	Versão 14.17 e posterior	Versão 13.20 e posterior
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Taipei)	Versão 17.4 e posterior	Versão 16.6 e posterior	Versão 15.10 e posterior	Versão 14.15 e posterior	Indisponível
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Ásia-Pacífico (Tailândia)	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 15.8 e posterior	Versão 14.13 e posterior	Indisponível
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Oeste do Canadá (Calgary)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.6, 14.8 e posterior	Versão 13.9, 13.11 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
China (Pequim)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
China (Ningxia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Milão)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Europa (Zurique)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior
Israel (Tel Aviv)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior
México (Centro)	Versão 17.4 e posterior	Versão 16.4 e posterior	Versão 15.8 e posterior	Versão 14.13 e posterior	Indisponível

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.6 e posterior	Versão 13.9 e posterior
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
AWS GovCloud (Leste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior
AWS GovCloud (Oeste dos EUA)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.2 e posterior	Versão 14.3 e posterior	Versão 13.6 e posterior

Os limites de capacidade máximo e mínimo em ACU do Aurora Serverless v2 podem variar dependendo da versão do mecanismo. Para obter mais detalhes, consulte Aurora Serverless v2Capacidade do.

Aurora Serverless v1 da

Important

A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Recomendamos atualizar quaisquer clusters de banco de dados do Aurora Serverless v1

para Aurora Serverless v2 antes dessa data. A atualização pode envolver uma alteração no número da versão principal do mecanismo de banco de dados. Portanto, é importante planejar, testar e implementar essa transição antes da data de fim da vida útil. A partir de 8 de janeiro de 2025, os clientes não poderão mais criar novos clusters ou instâncias do Aurora Serverless v1 com o Console de gerenciamento da AWS ou a CLI. Para obter informações sobre o processo de migração, consulte Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.
O Aurora Serverless v2 escala mais rapidamente e de forma mais granular. O Aurora Serverless v2 também tem mais compatibilidade com outros recursos do Aurora, como instâncias de banco de dados de leitura. Saiba mais sobre o Aurora Serverless v2 em Usar o Aurora Serverless v2.

Aurora Serverless v1O é um recurso de escalabilidade automática sob demanda, projetado para ser uma abordagem econômica e executar workloads intermitentes ou imprevisíveis no Amazon Aurora. Ele automaticamente inicia, encerra e aumenta ou reduz a escala da capacidade na vertical de acordo com as necessidades de sua aplicação, usando uma única instância de banco de dados em cada cluster. Para ter mais informações, consulte Usar o Amazon Aurora Serverless v1.

Tópicos

- Aurora Serverless v1 com o Aurora MySQL
- Aurora Serverless v1 com o Aurora PostgreSQL

Aurora Serverless v1 com o Aurora MySQL

As regiões e as versões do mecanismo a seguir estão disponíveis para Aurora Serverless v1 com Aurora MySQL.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Indisponível	Versão 2.11.4
Leste dos EUA (Ohio)	Indisponível	Versão 2.11.4
Oeste dos EUA (Norte da Califórnia)	Indisponível	Versão 2.11.4

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Oeste dos EUA (Oregon)	Indisponível	Versão 2.11.4
África (Cidade do Cabo)	Não disponível	Não disponível
Ásia-Pacífico (Hong Kong)	Não disponível	Não disponível
Ásia-Pacífico (Hyderabad)	Não disponível	Não disponível
Ásia-Pacífico (Jacarta)	Não disponível	Não disponível
Ásia-Pacífico (Malásia)	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Não disponível	Não disponível
Ásia-Pacífico (Mumbai)	Indisponível	Versão 2.11.4
Ásia-Pacífico (Osaka)	Não disponível	Não disponível
Ásia-Pacífico (Seul)	Indisponível	Versão 2.11.4
Ásia-Pacífico (Singapura)	Indisponível	Versão 2.11.4
Ásia-Pacífico (Sydney)	Indisponível	Versão 2.11.4
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível
Ásia-Pacífico (Tóquio)	Indisponível	Versão 2.11.4
Canadá (Central)	Indisponível	Versão 2.11.4
Oeste do Canadá (Calgary)	Não disponível	Não disponível
China (Pequim)	Não disponível	Não disponível
China (Ningxia)	Indisponível	Versão 2.11.4
Europa (Frankfurt)	Indisponível	Versão 2.11.4
Europa (Irlanda)	Indisponível	Versão 2.11.4

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Londres)	Indisponível	Versão 2.11.4
Europa (Milão)	Não disponível	Não disponível
Europa (Paris)	Indisponível	Versão 2.11.4
Europa (Espanha)	Não disponível	Não disponível
Europa (Estocolmo)	Não disponível	Não disponível
Europa (Zurique)	Não disponível	Não disponível
Israel (Tel Aviv)	Não disponível	Não disponível
Oriente Médio (Bahrein)	Não disponível	Não disponível
Oriente Médio (Emirados Árabes Unidos)	Não disponível	Não disponível
América do Sul (São Paulo)	Não disponível	Não disponível
AWS GovCloud (Leste dos EUA)	Não disponível	Não disponível
AWS GovCloud (Oeste dos EUA)	Não disponível	Não disponível

Aurora Serverless v1 com o Aurora PostgreSQL

As regiões e as versões do mecanismo a seguir estão disponíveis para Aurora Serverless v1 com Aurora PostgreSQL.

Região	Aurora PostgreSQL 13
Leste dos EUA (Norte da Virgínia)	Versão 13.12
Leste dos EUA (Ohio)	Versão 13.12

Região	Aurora PostgreSQL 13
Oeste dos EUA (Norte da Califórnia)	Versão 13.12
Oeste dos EUA (Oregon)	Versão 13.12
África (Cidade do Cabo)	Indisponível
Ásia-Pacífico (Hong Kong)	Indisponível
Ásia-Pacífico (Hyderabad)	Indisponível
Ásia-Pacífico (Jacarta)	Indisponível
Ásia-Pacífico (Malásia)	Indisponível
Ásia-Pacífico (Melbourne)	Indisponível
Ásia-Pacífico (Mumbai)	Versão 13.12
Ásia-Pacífico (Osaka)	Indisponível
Ásia-Pacífico (Seul)	Versão 13.12
Ásia-Pacífico (Singapura)	Versão 13.12
Ásia-Pacífico (Sydney)	Versão 13.12
Ásia-Pacífico (Tailândia)	Indisponível
Ásia-Pacífico (Tóquio)	Versão 13.12
Canadá (Central)	Versão 13.12
Oeste do Canadá (Calgary)	Indisponível
China (Pequim)	Indisponível
China (Ningxia)	Indisponível
Europa (Frankfurt)	Versão 13.12

Região	Aurora PostgreSQL 13
Europa (Irlanda)	Versão 13.12
Europa (Londres)	Versão 13.12
Europa (Milão)	Indisponível
Europa (Paris)	Versão 13.12
Europa (Espanha)	Indisponível
Europa (Estocolmo)	Indisponível
Europa (Zurique)	Indisponível
Israel (Tel Aviv)	Indisponível
Oriente Médio (Bahrein)	Indisponível
Oriente Médio (Emirados Árabes Unidos)	Indisponível
América do Sul (São Paulo)	Indisponível
AWS GovCloud (Leste dos EUA)	Indisponível
AWS GovCloud (Oeste dos EUA)	Indisponível

Regiões e mecanismos de banco de dados do Aurora compatíveis com a API de dados do RDS

A API de dados do RDS (API de dados) oferece uma interface de serviços da web a um cluster de banco de dados do Amazon Aurora. Em vez de gerenciar conexões de banco de dados em aplicações cliente, é possível executar comandos SQL em um endpoint HTTPS. Para obter mais informações, consulte Usar a API de dados do Amazon RDS.

Tópicos

- API de dados do RDS para o Aurora PostgreSQL Sem Servidor v2 e provisionado
- API de dados com o Aurora MySQL Sem Servidor v2 e provisionado

- API de dados com o Aurora PostgreSQL Sem Servidor v1
- API de dados com o Aurora MySQL Sem Servidor v1

API de dados do RDS para o Aurora PostgreSQL Sem Servidor v2 e provisionado

As regiões e as versões do mecanismo a seguir estão disponíveis para a API de dados com o Aurora PostgreSQL Sem Servidor v2 e clusters de banco de dados provisionados.

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Leste dos EUA (Norte da Virgínia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Leste dos EUA (Ohio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Oeste dos EUA (Oregon)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
África (Cidade do Cabo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Hong Kong)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Hyderabad)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Ásia-Pacífico (Jacarta)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Malásia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Melbourne)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Mumbai)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Nova Zelândia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Osaka)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Seul)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Singapura)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Sydney)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Ásia-Pacífico (Taipei)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Ásia-Pacífico (Tailândia)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Ásia-Pacífico (Tóquio)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Canadá (Central)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Oeste do Canadá (Calgary)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
China (Pequim)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
China (Ningxia)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Europa (Frankfurt)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Irlanda)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Londres)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Milão)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Paris)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Espanha)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior

Região	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europa (Estocolmo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
Europa (Zurique)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Israel (Tel Aviv)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Bahrein)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
México (Centro)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Oriente Médio (Emirados Árabes Unidos)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
América do Sul (São Paulo)	Versão 17.4 e posterior	Versão 16.1 e posterior	Versão 15.3 e posterior	Versão 14.8 e posterior	Versão 13.11 e posterior
AWS GovCloud (Leste dos EUA)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
AWS GovCloud (Oeste dos EUA)	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível

API de dados com o Aurora MySQL Sem Servidor v2 e provisionado

As regiões e as versões do mecanismo a seguir estão disponíveis para a API de dados com o Aurora MySQL Sem Servidor v2 e clusters de banco de dados provisionados.

Região	Aurora MySQL versão 3
Leste dos EUA (Ohio)	Versão 3.07 e posterior
Leste dos EUA (Norte da Virgínia)	Versão 3.07 e posterior
Oeste dos EUA (N. da Califórnia)	Versão 3.07 e posterior
Oeste dos EUA (Oregon)	Versão 3.07 e posterior
África (Cidade do Cabo)	Versão 3.07 e posterior
Ásia-Pacífico (Hong Kong)	Versão 3.07 e posterior
Ásia-Pacífico (Hyderabad)	Indisponível
Ásia-Pacífico (Jacarta)	Versão 3.07 e posterior
Ásia-Pacífico (Malásia)	Versão 3.07 e posterior
Ásia-Pacífico (Melbourne)	Indisponível
Ásia-Pacífico (Mumbai)	Versão 3.07 e posterior
Ásia-Pacífico (Nova Zelândia)	Indisponível
Ásia-Pacífico (Osaka)	Versão 3.07 e posterior
Ásia-Pacífico (Seul)	Versão 3.07 e posterior
Ásia-Pacífico (Singapura)	Versão 3.07 e posterior
Ásia-Pacífico (Sydney)	Versão 3.07 e posterior
Ásia-Pacífico (Taipei)	Indisponível
Ásia-Pacífico (Tailândia)	Versão 3.07 e posterior

Região	Aurora MySQL versão 3
Ásia-Pacífico (Tóquio)	Versão 3.07 e posterior
Canadá (Central)	Versão 3.07 e posterior
Oeste do Canadá (Calgary)	Indisponível
China (Pequim)	Indisponível
China (Ningxia)	Indisponível
Europa (Frankfurt)	Versão 3.07 e posterior
Europa (Irlanda)	Versão 3.07 e posterior
Europa (Londres)	Versão 3.07 e posterior
Europa (Milão)	Versão 3.07 e posterior
Europa (Paris)	Versão 3.07 e posterior
Europa (Espanha)	Versão 3.07 e posterior
Europa (Estocolmo)	Versão 3.07 e posterior
Europa (Zurique)	Indisponível
Israel (Tel Aviv)	Indisponível
México (Centro)	Indisponível
Oriente Médio (Bahrein)	Versão 3.07 e posterior
Oriente Médio (Emirados Árabes Unidos)	Versão 3.07 e posterior
América do Sul (São Paulo)	Versão 3.07 e posterior
AWS GovCloud (Leste dos EUA)	Indisponível
AWS GovCloud (Oeste dos EUA)	Indisponível

API de dados com o Aurora PostgreSQL Sem Servidor v1

As regiões e as versões do mecanismo a seguir estão disponíveis para a API de dados com o Aurora PostgreSQL Sem Servidor v1.

Região	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Leste dos EUA (Norte da Virgínia)	Versão 13.9	Versão 11.18
Leste dos EUA (Ohio)	Versão 13.9	Versão 11.18
Oeste dos EUA (N. da Califórnia)	Versão 13.9	Versão 11.18
Oeste dos EUA (Oregon)	Versão 13.9	Versão 11.18
África (Cidade do Cabo)	Não disponível	Não disponível
Ásia-Pacífico (Hong Kong)	Não disponível	Não disponível
Ásia-Pacífico (Hyderabad)	Não disponível	Não disponível
Ásia-Pacífico (Jacarta)	Não disponível	Não disponível
Ásia-Pacífico (Malásia)	Não disponível	Não disponível
Ásia-Pacífico (Melbourne)	Não disponível	Não disponível
Ásia-Pacífico (Mumbai)	Versão 13.9	Versão 11.18
Ásia-Pacífico (Osaka)	Não disponível	Não disponível
Ásia-Pacífico (Seul)	Versão 13.9	Versão 11.18
Ásia-Pacífico (Singapura)	Versão 13.9	Versão 11.18
Ásia-Pacífico (Sydney)	Versão 13.9	Versão 11.18
Ásia-Pacífico (Tailândia)	Não disponível	Não disponível

Região	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Ásia-Pacífico (Tóquio)	Versão 13.9	Versão 11.18
Canadá (Central)	Versão 13.9	Versão 11.18
China (Pequim)	Não disponível	Não disponível
China (Ningxia)	Não disponível	Não disponível
Europa (Frankfurt)	Versão 13.9	Versão 11.18
Europa (Irlanda)	Versão 13.9	Versão 11.18
Europa (Londres)	Versão 13.9	Versão 11.18
Europa (Milão)	Não disponível	Não disponível
Europa (Paris)	Versão 13.9	Versão 11.18
Europa (Espanha)	Versão 13.9	Versão 11.18
Europa (Estocolmo)	Não disponível	Não disponível
Europa (Zurique)	Não disponível	Não disponível
Israel (Tel Aviv)	Não disponível	Não disponível
Oriente Médio (Bahrein)	Não disponível	Não disponível
Oriente Médio (Emirados Árabes Unidos)	Não disponível	Não disponível
América do Sul (São Paulo)	Não disponível	Não disponível
AWS GovCloud (Leste dos EUA)	Não disponível	Não disponível
AWS GovCloud (Oeste dos EUA)	Não disponível	Não disponível

API de dados com o Aurora MySQL Sem Servidor v1

As regiões e as versões do mecanismo a seguir estão disponíveis para a API de dados com o Aurora MySQL Sem Servidor v1.

Região	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Versão 2.11.3
Leste dos EUA (Ohio)	Versão 2.11.3
Oeste dos EUA (N. da Califórnia)	Versão 2.11.3
Oeste dos EUA (Oregon)	Versão 2.11.3
África (Cidade do Cabo)	Indisponível
Ásia-Pacífico (Hong Kong)	Indisponível
Ásia-Pacífico (Hyderabad)	Indisponível
Ásia-Pacífico (Jacarta)	Indisponível
Ásia-Pacífico (Malásia)	Indisponível
Ásia-Pacífico (Melbourne)	Indisponível
Ásia-Pacífico (Mumbai)	Versão 2.11.3
Ásia-Pacífico (Osaka)	Indisponível
Ásia-Pacífico (Seul)	Versão 2.11.3
Ásia-Pacífico (Singapura)	Versão 2.11.3
Ásia-Pacífico (Sydney)	Versão 2.11.3
Ásia-Pacífico (Tailândia)	Indisponível
Ásia-Pacífico (Tóquio)	Versão 2.11.3
Canadá (Central)	Versão 2.11.3

Região	Aurora MySQL versão 2
Oeste do Canadá (Calgary)	Indisponível
China (Pequim)	Indisponível
China (Ningxia)	Versão 2.11.3
Europa (Frankfurt)	Versão 2.11.3
Europa (Irlanda)	Versão 2.11.3
Europa (Londres)	Versão 2.11.3
Europa (Milão)	Indisponível
Europa (Paris)	Versão 2.11.3
Europa (Espanha)	Versão 2.11.3
Europa (Estocolmo)	Indisponível
Europa (Zurique)	Indisponível
Israel (Tel Aviv)	Indisponível
Oriente Médio (Bahrein)	Indisponível
Oriente Médio (Emirados Árabes Unidos)	Indisponível
América do Sul (São Paulo)	Indisponível
AWS GovCloud (Leste dos EUA)	Indisponível
AWS GovCloud (Oeste dos EUA)	Indisponível

Regiões e mecanismos de banco de dados do Aurora compatíveis com a aplicação de patches com tempo de inatividade zero (ZDP)

A execução de upgrades para clusters de banco de dados do Aurora envolve a possibilidade de uma interrupção quando o banco de dados é desligado e enquanto ele está sendo atualizado. Por padrão, se você iniciar a atualização enquanto o banco de dados estiver ocupado, perderá todas as conexões e transações que o cluster de banco de dados está processando. Se você esperar até que o banco de dados fique ocioso para realizar a atualização, talvez seja necessário esperar muito tempo.

O atributo de aplicação de patches com tempo de inatividade zero (ZDP) tenta, com o melhor esforço, preservar as conexões de clientes por meio de um upgrade do Aurora. Se a ZDP for concluída com êxito, as sessões da aplicação serão preservadas, e o mecanismo de banco de dados será reiniciado enquanto o upgrade estiver em andamento. A reinicialização do mecanismo de banco de dados pode causar uma queda na taxa de transferência com duração de alguns segundos a aproximadamente um minuto.

Para obter informações detalhadas sobre as condições e as versões de mecanismo em que o recurso de ZDP está disponível para upgrades do Aurora MySQL, consulte Como usar os patches com tempo de inatividade zero.

Para obter informações detalhadas sobre as condições e as versões de mecanismo em que o recurso de ZDP está disponível para upgrades do Aurora PostgreSQL, consulte Atualizações de versões secundárias e patches com tempo de inatividade zero.

Base de datos ilimitada de Aurora PostgreSQL de regiones admitidas

Base de datos ilimitada de Amazon Aurora PostgreSQL ofrece un escalado horizontal automatizado para procesar millones de transacciones de escritura por segundo y administra petabytes de datos a la vez que mantiene la simplicidad de operar dentro de una única base de datos. Con Base de datos ilimitada de Aurora PostgreSQL, puede centrarse en crear aplicaciones a gran escala sin tener que crear y mantener soluciones complejas para escalar sus datos en varias instancias de bases de datos para respaldar sus cargas de trabajo.

Para obtener más información, consulte Uso do Amazon Aurora PostgreSQL Limitless Database.

La base de datos ilimitada de Aurora PostgreSQL está disponible en todas las Regiões da AWS excepto en Asia-Pacífico (Taipéi).

Regiões e mecanismos de banco de dados compatíveis com recursos nativos do mecanismo do Aurora

Os mecanismos de banco de dados do Aurora também oferecem compatibilidade com recursos e funcionalidades adicionais especificamente para o Aurora. Alguns recursos nativos do mecanismo podem ter suporte limitado ou privilégios restritos para um mecanismo, versão ou região específicos do banco de dados do Aurora.

Tópicos

- Recursos nativos do mecanismo para o Aurora MySQL
- Recursos nativos do mecanismo para o Aurora PostgreSQL

Recursos nativos do mecanismo para o Aurora MySQL

Veja a seguir recursos nativos do mecanismo para o Aurora MySQL.

- Auditoria avançada
- Retrocesso
- Consultas de injeção de falhas
- Encaminhamento de gravação em cluster
- Consulta paralela

Recursos nativos do mecanismo para o Aurora PostgreSQL

Veja a seguir recursos nativos do mecanismo para o Aurora PostgreSQL.

- Babelfish
- Consultas de injeção de falhas
- Gerenciamento de planos de consulta

Conexões de endpoints do Amazon Aurora

O Amazon Aurora normalmente envolve um cluster de instâncias de banco de dados, em vez de uma única instância. Cada conexão é processada por uma instância de banco de dados específica. Quando você se conecta a um cluster do Aurora, o nome do host e a porta especificados apontam

para um processador intermediário chamado endpoint. O Aurora usa o mecanismo de endpoint para abstrair essas conexões. Por isso, você não precisa codificar todos os nomes de host ou escrever a própria lógica para balancear e reorganizar conexões quando algumas instâncias de banco de dados não estão disponíveis.

Em determinadas tarefas do Aurora, as instâncias ou os grupos diferentes de instâncias realizam funções diferentes. Por exemplo, a instância primária processa todas as instruções Data Definition Language (DDL) e Data Manipulation Language (DML). Até 15 réplicas do Aurora processam tráfego de consulta somente leitura.

Tópicos

- Tipos de endpoints do Aurora
- Visualizar os endpoints de um cluster do Aurora
- Como os endpoints do Aurora funcionam com alta disponibilidade
- Endpoints de cluster para Amazon Aurora
- Endpoints de leitor para Amazon Aurora
- Endpoints de instância para Amazon Aurora
- Endpoints personalizados para o Amazon Aurora

Tipos de endpoints do Aurora

Usando endpoints, mapeie todas as conexões para a instância apropriada ou o grupo de instâncias baseado no caso de uso. Por exemplo, para realizar instruções DDL, conecte-se à instância que seja a primária. Para realizar consultas, conecte-se ao endpoint do leitor. O Aurora realizará automaticamente o balanceamento de conexão entre todas as respectivas réplicas. Em clusters com instâncias de banco de dados de capacidades ou configurações diferentes, conecte-se a endpoints personalizados associados a subconjuntos diferentes de instâncias de banco de dados. Para diagnóstico ou ajuste, conecte-se a um endpoint de instância específico para examinar detalhes sobre uma instância de banco de dados específica.

Um endpoint é representado como um URL específico do Aurora que contém um endereço de host e uma porta. Os tipos a seguir de endpoints estão disponíveis em um cluster de banco de dados do Aurora.

Endpoint do cluster

Conecte-se à instância primária do cluster para desenvolver e testar aplicações e realizar determinadas transformações, como declarações `INSERT` e operações de DDL, DML e ETL. Encontre a localização do endpoint do cluster usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS, conforme descrito em Visualizar os endpoints de um cluster do Aurora.

Para ter mais informações sobre endpoints do cluster, consulte Endpoints de cluster para Amazon Aurora.

Endpoint de leitor

Realize consultas. O Aurora executa automaticamente o balanceamento de conexão entre todas as réplicas do Aurora. Encontre a localização do endpoint do leitor usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS, conforme descrito em Visualizar os endpoints de um cluster do Aurora.

Para ter mais informações sobre os endpoints do leitor, consulte Endpoints de leitor para Amazon Aurora.

Endpoint da instância

Examine os detalhes sobre uma instância de banco de dados específica para diagnóstico ou ajuste. É possível encontrar a localização do endpoint da instância para cada uma de suas instâncias somente no Console de gerenciamento da AWS, na página de detalhes da instância.

Para ter mais informações sobre os endpoints de instâncias, consulte Endpoints de instância para Amazon Aurora.

Endpoint personalizado

Conecte-se a diferentes subconjuntos de instâncias de banco de dados no cluster de banco de dados. Isso é útil quando você tem diferentes recursos e configurações de instância no cluster de banco de dados. Encontre as localizações de endpoints personalizados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS, conforme descrito em Visualizar os endpoints de um cluster do Aurora.

Para ter mais informações sobre os endpoints personalizados, consulte Endpoints personalizados para o Amazon Aurora.

Endpoint do gravador do Aurora Global Database

O Aurora Global Database tem um tipo especial de endpoint que atende à mesma finalidade que o endpoint de um cluster autônomo do Aurora. Ele processa solicitações de gravação e de leitura. Quando um cluster secundário se torna o novo cluster primário devido a uma transição ou um failover, o Aurora muda automaticamente esse endpoint para apontar para o endpoint do novo cluster primário, na outra Região da AWS. Dessa forma, não é necessário codificar a região da AWS na string de conexão da aplicação nem alterar a string de conexão quando o layout do banco de dados global é alterado. O Aurora cria esse endpoint quando um Aurora Global Database é configurado, por exemplo, pela escolha de Adicionar região para um cluster do Aurora no Console de gerenciamento da AWS.

Para ter informações sobre como usar esse tipo de endpoint com o Aurora Global Database, consulte Conectar-se ao Amazon Aurora Global Database.

Visualizar os endpoints de um cluster do Aurora

Embora você só possa encontrar a localização do endpoint da instância na página de detalhes da instância no Console de gerenciamento da AWS, é possível usar o console, a AWS CLI ou a API do Amazon RDS para encontrar a localização do cluster, do leitor e dos endpoints personalizados.

Console

No Console de gerenciamento da AWS, encontre o endpoint do cluster, o endpoint do leitor e todos os endpoints personalizados na página de detalhes da instância do cluster. Você vê o endpoint de instância na página de detalhes de cada instância. Ao se conectar, acrescente o número da porta associada, seguido de dois-pontos, ao nome do endpoint mostrado na página de detalhes.

AWS CLI

Com a AWS CLI, você encontra o endpoint do gravador e do leitor e qualquer outro endpoint personalizado na saída do comando describe-db-clusters. Por exemplo, o comando a seguir mostra os atributos de endpoint para todos os clusters na região atual da AWS.

```
aws rds describe-db-clusters --query '*[].
{Endpoint:Endpoint,ReaderEndpoint:ReaderEndpoint,CustomEndpoints:CustomEndpoints}'
```

Amazon RDS API

Com a API do Amazon RDS, você recupera os endpoints chamando a operação DescribeDBClusterEndpoints.

Como os endpoints do Aurora funcionam com alta disponibilidade

Para clusters em que a alta disponibilidade é importante, use o endpoint do cluster para conexões de leitura/gravação ou de uso geral e o endpoint do leitor para conexões somente leitura. Os endpoints de leitor e de gravador gerenciam o failover da instância de banco de dados melhor do que os endpoints de instância. Ao contrário dos endpoints de instância, os endpoints de leitor e de gravador alteram automaticamente a qual instância de banco de dados eles se conectam caso uma instância de banco de dados no cluster fique indisponível. Para ter mais informações sobre endpoints do cluster e do leitor, consulte Endpoints de cluster para Amazon Aurora e Endpoints de leitor para Amazon Aurora.

Se a instância de banco de dados primária de um cluster de banco de dados falhar, o Aurora fará failover automático para uma nova instância de banco de dados primária. Ela faz isso promovendo uma réplica do Aurora existente para uma nova instância de banco de dados primária ou criando uma nova instância de banco de dados primária. Se ocorrer failover, você poderá usar o endpoint do cluster para se reconectar à instância de banco de dados primária recém-promovida ou criada ou usar o endpoint do leitor para se reconectar a uma das réplicas do Aurora no cluster de banco de dados. Durante um failover, o endpoint de leitor pode direcionar conexões para a nova instância de banco de dados primária de um cluster de banco de dados por um curto período depois que uma réplica do Aurora é promovida para a nova instância de banco de dados primária.

Caso projete a própria lógica de aplicativo para gerenciar conexões com endpoints de instância, descubra de maneira manual ou programática o conjunto resultante de instâncias de banco de dados disponíveis no cluster de banco de dados. Use o comando describe-db-clusters da AWS CLI ou a operação DescribeDBClusters da API do RDS para encontrar o cluster de banco de dados, os endpoints leitores e as instâncias de banco de dados, e identificar se as instâncias de banco de dados são leitoras e quais seus níveis de promoção. Em seguida, confirme as classes de instância após o failover e se conecte a um endpoint de instância apropriado.

Para obter mais informações sobre failovers, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

Para ter mais informações sobre alta disponibilidade no Amazon Aurora, consulte Alta disponibilidade do Amazon Aurora.

Endpoints de cluster para Amazon Aurora

Um endpoint de cluster (ou endpoint de gravador) de um cluster de banco de dados do Aurora se conecta à instância de banco de dados primária atual desse cluster de banco de dados. Esse endpoint é o único capaz de realizar operações de gravação como instruções DDL. Por isso, o endpoint cluster é o único que se conecta a quando você configura um cluster ou quando o cluster só contém uma única instância de banco de dados.

Cada cluster de banco de dados do Aurora tem um único endpoint cluster e uma única instância de banco de dados primária.

Use o endpoint cluster em todas as operações de gravação no cluster de banco de dados, inclusive inserções, atualizações, exclusões e alterações DDL. Você também pode usar o endpoint de cluster para operações de leitura, como consultas.

O endpoint de cluster dá suporte a failover para conexões de leitura/gravação para o cluster de banco de dados. Se a instância de banco de dados primária atual de um cluster de banco de dados falhar, o Aurora fará failover automático para uma nova instância de banco de dados primária. Durante um failover, o cluster de banco de dados continua atendendo a solicitações de conexão para o endpoint de cluster pela nova instância de banco de dados primária, com interrupção mínima de serviço.

O exemplo a seguir ilustra um endpoint de cluster de um cluster de banco de dados do Aurora MySQL.

```
mydbcluster.cluster-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Cada cluster do Aurora tem um único endpoint de cluster integrado, cujo nome e outros atributos são gerenciados pelo Aurora. Não crie, exclua nem modifique esse tipo de endpoint.

Use o endpoint cluster ao administrar o cluster, realize operações Extract, Transform, Load (ETL – Extração, transformação, carregamento) ou desenvolva e teste aplicações. O endpoint cluster se conecta à instância primária do cluster. A instância primária é a única instância de banco de dados em que você cria tabelas e índices, executa instruções `INSERT` e realiza outras operações DDL e DML.

O endereço IP físico apontado pelo endpoint do cluster muda quando o mecanismo de failover promove uma nova instância de banco de dados como a instância primária de leitura/gravação do cluster. Caso você use alguma forma de agrupamento de conexões ou outra multiplexação, prepare-

se para enviar ou reduzir a vida útil para todas as informações DNS armazenadas em cache. Isso garante que você não tente estabelecer uma conexão de leitura/gravação com uma instância de banco de dados que fique indisponível ou seja somente leitura após um failover.

Endpoints de leitor para Amazon Aurora

Um endpoint de leitor de um cluster de banco de dados do Aurora comporta o balanceamento de conexões somente leitura com o cluster de banco de dados. Use o endpoint do leitor para operações de leitura, como consultas. Ao processar essas instruções nas réplicas somente leitura do Aurora, esse endpoint reduz a sobrecarga na instância primária. Ele também ajuda o cluster a dimensionar a capacidade de processar consultas `SELECT` simultâneas, proporcional ao número de réplicas do Aurora no cluster. Cada cluster de banco de dados do Aurora tem um único endpoint leitor.

Se o cluster tiver uma ou mais réplicas do Aurora, o endpoint de leitor fará o balanceamento de cada solicitação de conexão entre as respectivas réplicas. Nesse caso, só é possível executar somente instruções somente leitura, como `SELECT` nessa sessão. Se o cluster tiver apenas uma instância primária e nenhuma réplica do Aurora, o endpoint de leitor se conectará à instância primária. Nesse caso, é possível executar operações de gravação por meio do endpoint.

O exemplo a seguir ilustra um endpoint de leitor de um cluster de banco de dados do Aurora MySQL.

```
mydbcluster.cluster-ro-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Use o endpoint leitor em conexões somente leitura do cluster do Aurora. O endpoint usa um mecanismo de balanceamento de conexão para ajudar o cluster a processar uma workload que realiza muitas consultas. O endpoint leitor é o endpoint fornecido para aplicações que geram relatórios ou fazem outra operações somente leitura sobre o cluster.

O endpoint de leitor faz o balanceamento das conexões com réplicas do Aurora disponíveis em um cluster de banco de dados do Aurora. Ele não faz o balanceamento de consultas individuais. Se você quiser fazer o balanceamento de cada consulta para distribuir a workload de leitura de um cluster de banco de dados, abra uma nova conexão do endpoint de leitor para cada consulta.

Cada cluster do Aurora tem um único endpoint leitor integrado, cujo nome e outros atributos são gerenciados pelo Aurora. Não crie, exclua nem modifique esse tipo de endpoint.

Se o cluster contiver apenas um destino primário (instância ou grupo de fragmentos de banco de dados) e nenhuma réplica do Aurora, o endpoint de leitor se conectará à instância primária. Nesse caso, é possível executar operações de gravação por meio desse endpoint.

> **Tip**
>
> Com o RDS Proxy, você pode criar endpoints somente leitura adicionais para um cluster do Aurora. Esses endpoints realizam o mesmo tipo do balanceamento de conexão que o endpoint de leitor do Aurora. As aplicações podem se reconectar mais rapidamente aos endpoints do proxy do que o endpoint de leitor do Aurora se as instâncias do leitor ficarem indisponíveis. Os endpoints do proxy também podem tirar proveito de outros recursos de proxy, como a multiplexação. Para obter mais informações, consulte Uso de endpoints de leitor com clusters do Aurora.

Endpoints de instância para Amazon Aurora

Um endpoint de instância se conecta a uma instância de banco de dados específica dentro de um cluster do Aurora. Cada instância de banco de dados em um cluster de banco de dados, tem o próprio endpoint de instância exclusivo. Portanto, existe um endpoint para a instância de banco de dados primária atual do cluster de banco de dados e um endpoint de instância para cada uma das réplicas do Aurora no cluster de banco de dados.

O endpoint de instância oferece controle direto sobre as conexões do cluster de banco de dados, em cenários nos quais usar o endpoint de cluster ou o endpoint de leitor talvez não seja apropriado. Por exemplo, a aplicação cliente pode exigir um balanceamento de conexão mais refinado com base no tipo de workload. Nesse caso, você pode configurar vários clientes para se conectarem a réplicas diferentes do Aurora em um cluster de banco de dados para distribuir cargas de trabalho de leitura. Para ver um exemplo que use endpoints de instância para aumentar a velocidade de conexão após um failover do Aurora PostgreSQL, consulte Failover rápido com o Amazon Aurora PostgreSQL. Para ver um exemplo que usa endpoints de instância para aumentar a velocidade de conexão após um failover do Aurora MySQL, consulte o tópico Suporte a failover do MariaDB Connector/J – caso do Amazon Aurora.

O exemplo a seguir ilustra um endpoint de instância de uma instância de banco de dados em um cluster de banco de dados do Aurora MySQL.

```
mydbinstance.c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Cada instância de banco de dados em um cluster do Aurora tem o próprio endpoint de instância integrado, cujo nome e outros atributos são gerenciados pelo Aurora. Não crie, exclua nem modifique esse tipo de endpoint. Se você usa o Amazon RDS, pode estar familiarizado com endpoints de

instância. No entanto, com o Aurora normalmente você usa os endpoints de leitor e de gravador com mais frequência do que os endpoints de instância.

Nas operações diárias do Aurora, a principal maneira de usar endpoints de instância é fazer o diagnóstico dos problemas de capacidade ou de performance que afetam uma instância específica em um cluster do Aurora. Conectado a uma instância específica, examine as variáveis de status, as métricas etc. Fazer isso pode ajudar a determinar o que está acontecendo nessa instância diferente do que está acontecendo com outras instâncias no cluster.

Em casos de uso avançados, convém configurar algumas instâncias de banco de dados de maneira diferente de outras. Nesse caso, use o endpoint de instância para se conectar diretamente a uma instância que seja menor, maior ou tenha características diferentes das outras. Além disso, configure a prioridade de failover, de maneira que essa instância de banco de dados especial seja a última opção a ser tomada como a instância primária. Recomendamos usar endpoints personalizados, em vez do endpoint de instância nesses casos. Isso simplifica o gerenciamento de conexões e a alta disponibilidade à medida que você adiciona mais instâncias de banco de dados ao cluster.

Endpoints personalizados para o Amazon Aurora

Um endpoint personalizado para um cluster do Aurora representa um conjunto de instâncias de banco de dados escolhido. Quando você se conecta ao endpoint, o Aurora realiza o balanceamento de conexão e escolhe uma das instâncias no grupo para processar a conexão. Você define a quais instâncias esse endpoint se refere e determina a finalidade do endpoint.

Um cluster de banco de dados do Aurora não terá endpoints personalizados até você criar um. Você pode criar até cinco endpoints personalizados para cada cluster do Aurora provisionado ou do Aurora Serverless v2. Não use endpoints personalizados em clusters do Aurora Serverless v1.

O endpoint personalizado oferece conexões de banco de dados balanceadas baseadas em critérios que não sejam o recurso somente leitura ou de leitura/gravação das instâncias de banco de dados. Por exemplo, convém definir um endpoint personalizado para se conectar a instâncias que usem uma determinada classe de instância da AWS ou determinado parameter group de banco de dados. Em seguida, convém informar grupos particulares sobre esse endpoint personalizado. Por exemplo, convém direcionar usuários internos para instâncias de baixa capacidade tendo em vista a geração de relatórios ou a consulta ad hoc (individual) e direcionar o tráfego da produção para instâncias de alta capacidade.

Como a conexão pode ir para qualquer instância de banco de dados associada ao endpoint personalizado, recomendamos verificar se todas as instâncias de banco de dados dentro desse

grupo compartilham algumas características semelhantes. Isso garante que a performance, a capacidade da memória etc. sejam consistentes para todos os conectados a esse endpoint.

Esse recurso se destina a usuários avançados com tipos personalizados de cargas de trabalho em que não seja prático manter todas as réplicas do Aurora no cluster idêntico. Com endpoints personalizados, preveja a capacidade da instância de banco de dados usada em cada conexão. Ao usar endpoints personalizados, você normalmente não usa o endpoint leitor nesse cluster.

O exemplo a seguir ilustra um endpoint personalizado de uma instância de banco de dados em um cluster de banco de dados do Aurora MySQL.

```
myendpoint.cluster-custom-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Use endpoints personalizados para simplificar o gerenciamento de conexões quando o cluster contém instâncias de banco de dados com configurações de capacidades e configurações diferentes.

Anteriormente, você usava o mecanismo CNAMES para configurar aliases Domain Name Service (DNS – Serviço do nome de domínio) do próprio domínio para atingir resultados semelhantes. Usando endpoints personalizados, evite atualizar registros CNAME quando o cluster cresce ou diminua. Os endpoints personalizados também indicam que é possível usar conexões Transport Layer Security/Secure Sockets Layer (TLS/SSL) criptografadas.

Em vez de usar uma instância de banco de dados para cada finalidade especializada e se conectar ao endpoint de instância, é possível ter vários grupos de instâncias de banco de dados especializadas. Nesse caso, cada grupo tem o próprio endpoint personalizado. Dessa maneira, o Aurora pode realizar o balanceamento de conexão entre todas as instâncias dedicadas a determinadas tarefas, como a geração de relatórios ou o processamento de consultas de produção ou internas. Os endpoints personalizados distribuem conexões entre instâncias de forma passiva, usando o DNS para retornar aleatoriamente o endereço IP de uma das instâncias. Caso uma das instâncias de banco de dados dentro de um grupo se torne indisponível, o Aurora direciona conexões de endpoint personalizadas subsequentes para uma das outras instâncias de banco de dados associadas ao mesmo endpoint.

Tópicos

- Considerações sobre endpoints personalizados no Amazon Aurora
- Criar um endpoint personalizado
- Visualizar endpoints personalizados

- Editar um endpoint personalizado
- Excluir um endpoint personalizado
- Exemplos da AWS CLI de endpoints personalizados para o Amazon Aurora

Considerações sobre endpoints personalizados no Amazon Aurora

Use as seções a seguir para gerenciar, especificar propriedades e usar regras de associação para endpoints personalizados.

Tópicos

- Gerenciar endpoints personalizados
- Especificar propriedades para endpoints personalizados
- Regras de associação para endpoints personalizados

Gerenciar endpoints personalizados

Como clusters do Aurora recém-criados não têm endpoints personalizados, crie e gerencie esses objetos por conta própria. Faça isso usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

> **Note**
>
> Também crie e gerencie endpoints personalizados para clusters do Aurora restaurados de snapshots. Os endpoints personalizados não estão incluídos no snapshot. Você os recriará depois de restaurar e escolherá novos nomes de endpoint se o cluster restaurado estiver na mesma região do original.

Para trabalhar com endpoints personalizados no Console de gerenciamento da AWS, navegue até a página de detalhes do cluster do Aurora e use os controles na seção Custom Endpoints (Endpoints personalizados).

Para trabalhar com endpoints personalizados da AWS CLI, use estas operações:

- create-db-cluster-endpoint
- describe-db-cluster-endpoints

- modify-db-cluster-endpoint
- delete-db-cluster-endpoint

Para trabalhar com endpoints personalizados por meio da API do Amazon RDS, use as seguintes funções:

- CreateDBClusterEndpoint
- DescribeDBClusterEndpoints
- ModifyDBClusterEndpoint
- DeleteDBClusterEndpoint

Especificar propriedades para endpoints personalizados

O tamanho máximo do nome de um endpoint personalizado é 63 caracteres. O formato do nome é o seguinte:

```
endpoint_name.cluster-custom-customer_DNS_identifier.AWS_Region.rds.amazonaws.com
```

Não é possível reutilizar o mesmo nome de endpoint personalizado em mais de um cluster na mesma Região da AWS. O identificador de DNS personalizado é um identificador exclusivo associado à sua Conta da AWS em uma Região da AWS específica.

Cada endpoint personalizado tem um tipo associado que determina quais instâncias de banco de dados estão qualificadas para serem associadas a esse endpoint. Atualmente, o tipo pode ser `READER` ou `ANY`. As seguintes considerações se aplicam aos tipos de endpoint personalizados:

- Não selecione o tipo de endpoint personalizado no Console de gerenciamento da AWS. Todos os endpoints personalizados criados por meio do Console de gerenciamento da AWS têm um tipo de `ANY`.

  Você pode definir e modificar o tipo de endpoint personalizado usando a AWS CLI ou a API do Amazon RDS.
- Somente as instâncias de banco de dados do leitor podem fazer parte de um endpoint personalizado `READER`.
- As instâncias de banco de dados do leitor e do gravador podem fazer parte de um endpoint personalizado `ANY`. O Aurora direciona conexões para endpoints de cluster do tipo `ANY`para

qualquer instância de banco de dados associada com probabilidade igual. O tipo `ANY` se aplica a clusters que usam uma topologia de replicação.

- Caso você tente criar um endpoint personalizado com um tipo que não seja apropriado com base na configuração de replicação para um cluster, Aurora retorna um erro.

Regras de associação para endpoints personalizados

Quando você adiciona uma instância de banco de dados a um endpoint personalizado ou a remove de um endpoint personalizado, todas as conexões existentes com essa instância de banco de dados permanecem ativas.

Defina uma lista de instâncias de banco de dados para serem incluídas em ou excluídas de um endpoint personalizado. Nós nos referimos a essas listas como estáticas e exclusão, respectivamente. Use o mecanismo de inclusão/exclusão para subdividir ainda mais os grupos de instâncias de banco de dados e verifique se o conjunto de endpoints personalizados abrange todas as instâncias de banco de dados no cluster. Cada endpoint personalizado só pode conter um desses tipos de lista.

No Console de gerenciamento da AWS:

- A escolha é representada pela caixa de seleção Attach future instances added to this cluster (Anexar instâncias futuras adicionadas a esse cluster). Quando você mantém a caixa de seleção desmarcada, o endpoint personalizado usa uma lista estática contendo apenas as instâncias de banco de dados especificadas na página. Quando você marca a caixa de seleção, o endpoint personalizado usa uma lista de exclusões. Nesse caso, o endpoint personalizado representa todas as instâncias de banco de dados no cluster (inclusive as adicionadas futuramente), exceto as desmarcadas na caixa de diálogo.
- O console não permite que você especifique o tipo de endpoint. Todo endpoint personalizado criado usando o console é do tipo `ANY`.

  Portanto, o Aurora não altera a associação do endpoint personalizado quando as instâncias de banco de dados mudam de perfil entre gravador e leitor devido a um failover ou uma promoção.

Na AWS CLI e na API do Amazon RDS:

- Você pode especificar o tipo de endpoint. Portanto, quando o tipo de endpoint é definido como `READER` a associação do endpoint é automaticamente ajustada durante failovers e promoções.

Por exemplo, um endpoint personalizado com o tipo `READER` inclui uma réplica do Aurora que, depois, é promovida para uma instância de banco de dados de gravador. A nova instância gravadora não faz mais parte do endpoint personalizado.

- Você pode adicionar membros individuais e removê-los das listas depois que eles mudam de perfil. Use o comando modify-db-cluster-endpoint da AWS CLI ou a operação de API ModifyDBClusterEndpoint.

Associe uma instância de banco de dados a mais de um endpoint personalizado. Por exemplo, suponhamos que você adicione uma nova instância de banco de dados a um cluster, ou esse Aurora adiciona uma instância de banco de dados automaticamente por meio do mecanismo dimensionável. Nesses casos, a instância de banco de dados é adicionada a todos os endpoints personalizados para os quais está qualificada. A quais endpoints a instância de banco de dados é adicionada depende do tipo de endpoint personalizado de `READER` ou `ANY`, além de todas as listas estáticas ou de exclusões definidas para cada endpoint. Por exemplo, se o endpoint incluir uma lista estática de instâncias de banco de dados, as réplicas do Aurora recém-adicionadas não serão adicionadas a esse endpoint. Por outro lado, se o endpoint tiver uma lista de exclusões, as réplicas do Aurora recém-adicionadas serão adicionadas ao endpoint se não estiverem nomeadas na lista de exclusões e as funções corresponderem ao tipo do endpoint personalizado.

Caso se torne indisponível, uma réplica do Aurora continua associada a todos os endpoints personalizados. Por exemplo, ela continua fazendo parte do endpoint personalizado quando não há integridade, ela está parada, na reinicialização e assim por diante. Porém, não será possível se conectar a ela por meio desses endpoints até ela estar disponível novamente.

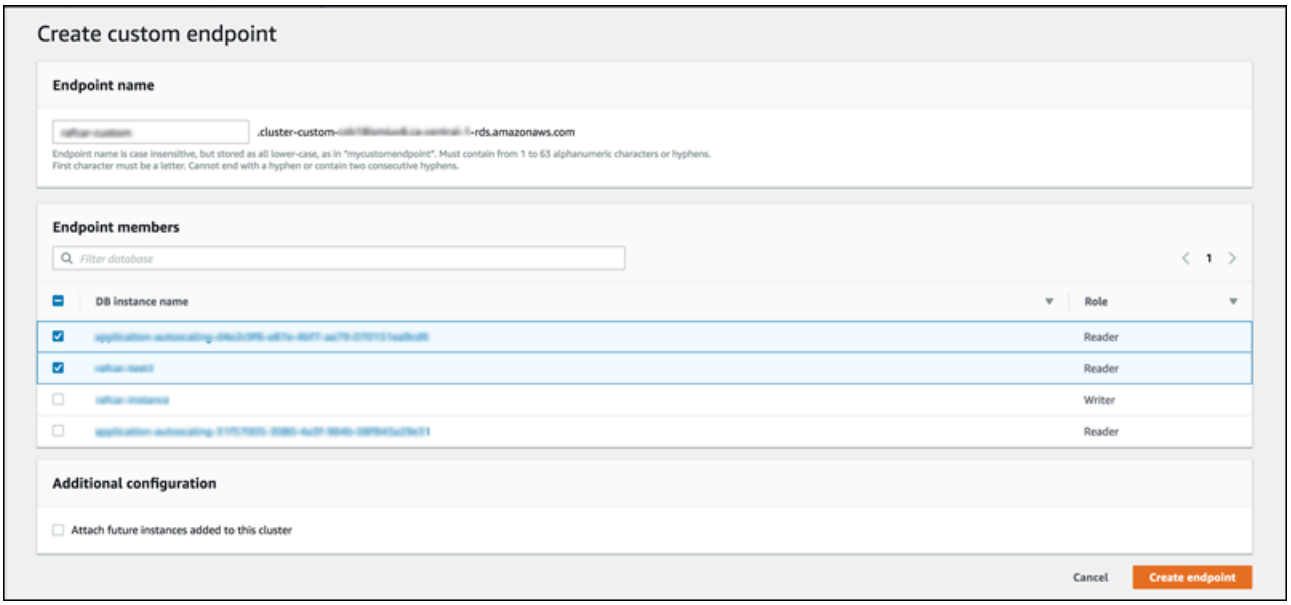
Criar um endpoint personalizado

Crie um endpoint personalizado usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

Console

Para criar um endpoint personalizado com o Console de gerenciamento da AWS, vá até a página de detalhes do cluster e escolha a ação `Create custom endpoint` na seção Endpoints. Escolha um nome para o endpoint personalizado, exclusivo para o ID do usuário e a região. Para escolher uma lista de instâncias de banco de dados que permaneça a mesma mesmo quando o cluster se expande, mantenha a caixa de seleção Attach future instances added to this cluster (Anexar instâncias futuras adicionadas a esse cluster) desmarcada. Quando você marca essa caixa de

seleção, o endpoint personalizado adiciona de maneira dinâmica todas as novas instâncias ao adicioná-las ao cluster.



Não selecione o tipo de endpoint personalizado no Console de gerenciamento da AWS. Todos os endpoints personalizados criados por meio do Console de gerenciamento da AWS têm um tipo de ANY.

AWS CLI

Para criar um endpoint personalizado com a AWS CLI, execute o comando create-db-cluster-endpoint.

O comando a seguir cria um endpoint personalizado anexado a um cluster específico. Inicialmente, o endpoint é associado a todas as instâncias de réplica do Aurora no cluster. Um comando subsequente o associa a um conjunto específico de instâncias de banco de dados no cluster.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample \
  --endpoint-type reader \
  --db-cluster-identifier cluster_id

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample \
  --static-members instance_name_1 instance_name_2
```

Para Windows:

```
aws rds create-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample ^
  --endpoint-type reader ^
  --db-cluster-identifier cluster_id

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample ^
  --static-members instance_name_1 instance_name_2
```

API do RDS

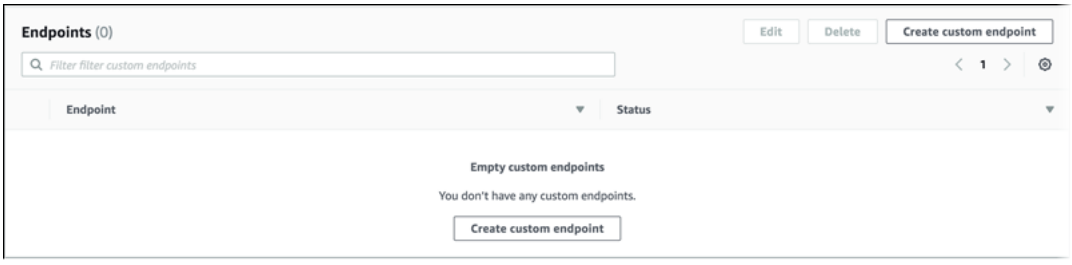
Para criar um endpoint personalizado com a API do RDS, execute a operação CreateDBClusterEndpoint.

Visualizar endpoints personalizados

Console

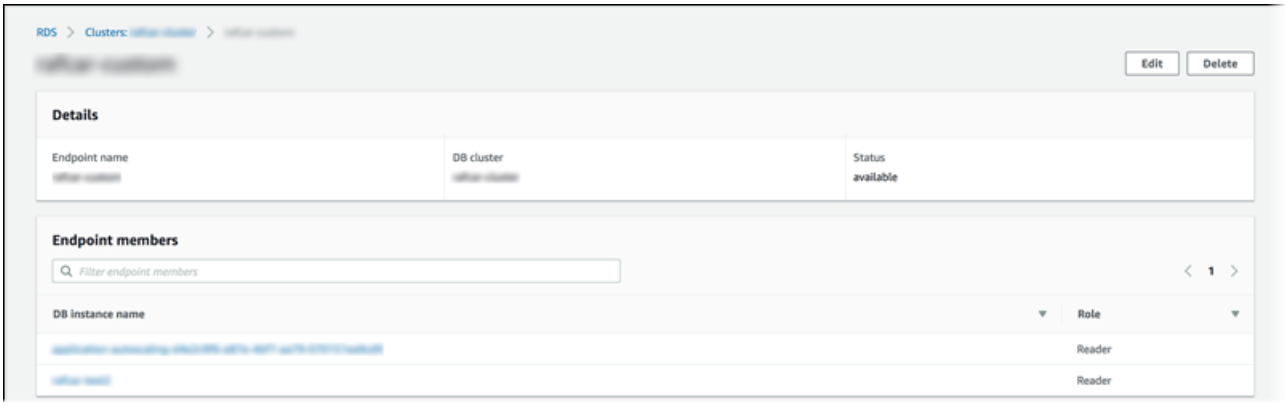
Para visualizar endpoints personalizados com o Console de gerenciamento da AWS, vá até a página de detalhes do cluster e veja a seção Endpoints. Esta seção contém informações apenas sobre endpoints personalizados. Os detalhes dos endpoints integrados estão listados na seção Details (Detalhes) principal. Para ver os detalhes de um endpoint personalizado específico, selecione o nome para abrir a página de detalhes desse endpoint.

A captura de tela a seguir mostra como a lista de endpoints personalizados de um cluster do Aurora está inicialmente vazia.



Depois que você cria alguns endpoints personalizados para esse cluster, eles são mostrados na seção Endpoints.

Clicar na página de detalhes mostra a quais instâncias de banco de dados o endpoint está associado no momento.



Para ver os outros detalhes sobre se novas instâncias de banco de dados adicionadas ao cluster são incluídas automaticamente no endpoint, abra a página Edit (Editar) do endpoint.

AWS CLI

Para visualizar endpoints personalizados com a AWS CLI, execute o comando describe-db-cluster-endpoints.

O comando a seguir mostra os endpoints personalizados associados a um cluster especificado em uma região especificada. A saída inclui os endpoints integrados e todos os endpoints personalizados.

Para Linux, macOS ou Unix:

```
aws rds describe-db-cluster-endpoints --region region_name \
  --db-cluster-identifier cluster_id
```

Para Windows:

```
aws rds describe-db-cluster-endpoints --region region_name ^
  --db-cluster-identifier cluster_id
```

Isto mostra algumas saídas de exemplo de um comando `describe-db-cluster-endpoints`. O `EndpointType` de `WRITER` ou `READER` denota os endpoints de leitura/gravação e somente leitura integrados do cluster. O `EndpointType` de `CUSTOM` denota endpoints criados e escolha as instâncias de banco de dados associadas. Um dos endpoints tem um campo `StaticMembers` não vazio, denotando que ele está associado a um conjunto preciso de instâncias de banco de dados. O outro endpoint tem um campo `ExcludedMembers` não vazio, denotando que o endpoint está associado a todas as instâncias de banco de dados diferentes das listadas em `ExcludedMembers`. Esse segundo tipo de endpoint personalizado cresce para acomodar novas instâncias à medida que você as adiciona ao cluster.

```
{
 "DBClusterEndpoints": [
  {
   "Endpoint": "custom-endpoint-demo.cluster-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "Status": "available",
   "DBClusterIdentifier": "custom-endpoint-demo",
   "EndpointType": "WRITER"
  },
  {
   "Endpoint": "custom-endpoint-demo.cluster-ro-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "Status": "available",
   "DBClusterIdentifier": "custom-endpoint-demo",
   "EndpointType": "READER"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "powers-of-2",
   "ExcludedMembers": [],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "powers-of-2.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-08",
     "custom-endpoint-demo-01",
     "custom-endpoint-demo-02"
   ],
```

```
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:powers-of-2"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "eight-and-higher",
   "ExcludedMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-02",
     "custom-endpoint-demo-07",
     "custom-endpoint-demo-05",
     "custom-endpoint-demo-03",
     "custom-endpoint-demo-06",
     "custom-endpoint-demo-01"
   ],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "eight-and-higher.cluster-custom-123456789012.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHYQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:eight-and-higher"
  }
 ]
}
```


API do RDS

Para visualizar endpoints personalizados com a API do RDS, execute a operação DescribeDBClusterEndpoints.html.

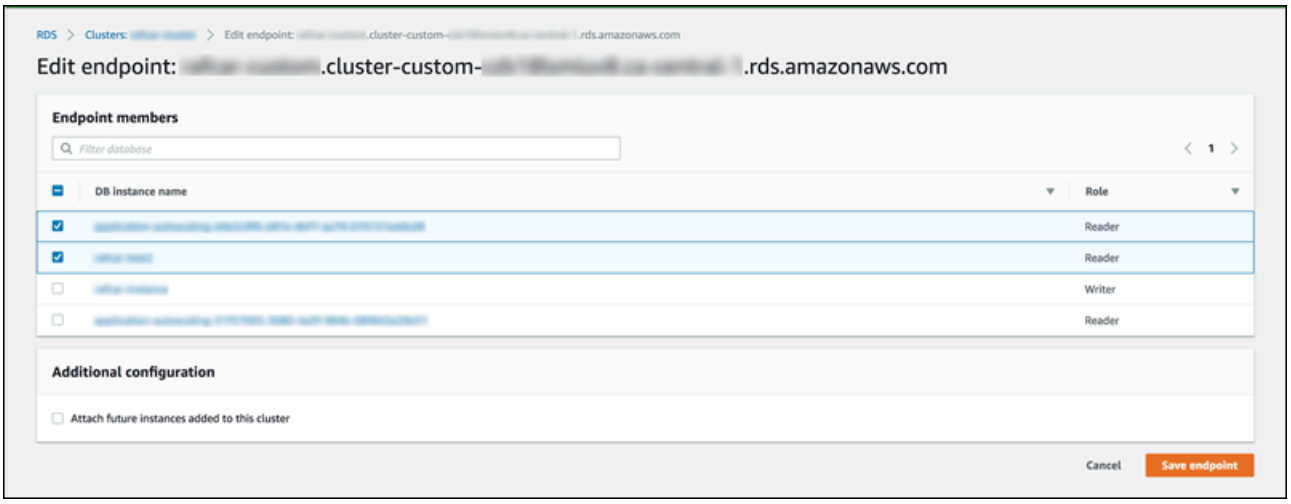
Editar um endpoint personalizado

Edite as propriedades de um endpoint personalizado para alterar quais instâncias de banco de dados estão associadas ao endpoint. Também altere um endpoint entre uma lista estática e uma lista de exclusões. Se você precisar de mais detalhes sobre essas propriedades de endpoint, consulte Regras de associação para endpoints personalizados.

Você pode continuar se conectando e usando um endpoint personalizado enquanto as alterações em uma ação de edição estiverem em andamento.

Console

Para editar um endpoint personalizado com o Console de gerenciamento da AWS, selecione o endpoint na página de detalhes do cluster ou abra a página de detalhes do endpoint e escolha a ação Edit (Editar).



AWS CLI

Para editar um endpoint personalizado com a AWS CLI, execute o comando modify-db-cluster-endpoint.

Os comandos a seguir alteram o conjunto de instâncias de banco de dados que se aplicam a um endpoint personalizado e podem alternar o comportamento de uma lista estática ou de exclusões. Os parâmetros `--static-members` e `--excluded-members` utilizam uma lista separada por espaços de identificadores de instância de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 \
  --static-members db-instance-id-1 db-instance-id-2 db-instance-id-3 \
  --region region_name

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 \
  --excluded-members db-instance-id-4 db-instance-id-5 \
  --region region_name
```

Para Windows:

```
aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 ^
  --static-members db-instance-id-1 db-instance-id-2 db-instance-id-3 ^
  --region region_name

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 ^
  --excluded-members db-instance-id-4 db-instance-id-5 ^
  --region region_name
```

API do RDS

Para editar um endpoint personalizado com a API do RDS, execute a operação ModifyDBClusterEndpoint.html.

Excluir um endpoint personalizado

Exclua um endpoint personalizado usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

Console

Para excluir um endpoint personalizado com o Console de gerenciamento da AWS, vá até a página de detalhes do cluster, selecione o endpoint personalizado apropriado e a ação Delete (Excluir).

AWS CLI

Para excluir um endpoint personalizado com a AWS CLI, execute o comando delete-db-cluster-endpoint.

O comando a seguir exclui um endpoint personalizado. Você não precisa especificar o cluster associado, mas deve especificar a região.

Para Linux, macOS ou Unix:

```
aws rds delete-db-cluster-endpoint --db-cluster-endpoint-identifier custom-end-point-id
 \
  --region region_name
```

Para Windows:

```
aws rds delete-db-cluster-endpoint --db-cluster-endpoint-identifier custom-end-point-id
 ^
  --region region_name
```

API do RDS

Para excluir um endpoint personalizado com a API do RDS, execute a operação DeleteDBClusterEndpoint.

Exemplos da AWS CLI de endpoints personalizados para o Amazon Aurora

O tutorial a seguir usa exemplos da AWS CLI com sintaxe do shell do Unix para mostrar como é possível definir um cluster com várias instâncias de banco de dados “pequenas” e algumas instâncias de banco de dados “grandes” e criar endpoints personalizados estabelecer conexão com cada conjunto de instâncias de banco de dados. Para executar comandos semelhantes no próprio sistema, você deve estar familiarizado com os conceitos básicos de trabalhar com clusters do Aurora e o uso da AWS CLI para fornecer os valores próprios de parâmetros, como região, grupo de sub-redes e grupo de segurança da VPC.

Este exemplo demonstra as etapas de configuração iniciais: criar um cluster do Aurora e adicionar instâncias de banco de dados a ele. Este é um cluster heterogêneo, o que significa que nem todas as instâncias de banco de dados têm a mesma capacidade. A maioria das instâncias usa a classe AWS da instância da `db.r4.4xlarge`, mas as últimas duas instâncias de banco de dados usam `db.r4.16xlarge`. Cada um desses comandos `create-db-instance` de exemplo imprime a saída na tela e salva uma cópia do JSON em um arquivo para inspeção posterior.

```
aws rds create-db-cluster --db-cluster-identifier custom-endpoint-demo --engine aurora-
mysql \
     --engine-version 8.0.mysql_aurora.3.04.0 --master-username $MASTER_USER --manage-
master-user-password \
     --db-subnet-group-name $SUBNET_GROUP  --vpc-security-group-ids $VPC_SECURITY_GROUP
 \
     --region $REGION
```

```
for i in 01 02 03 04 05 06 07 08
do
  aws rds create-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
     --engine aurora --db-cluster-identifier custom-endpoint-demo --db-instance-class
 db.r4.4xlarge \
     --region $REGION \
     | tee custom-endpoint-demo-${i}.json
done

for i in 09 10
do
  aws rds create-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
     --engine aurora --db-cluster-identifier custom-endpoint-demo --db-instance-class
 db.r4.16xlarge \
     --region $REGION \
     | tee custom-endpoint-demo-${i}.json
done
```

As instâncias maiores são reservadas para tipos especializados de consultas de relatórios. Para a impossibilidade de que elas sejam promovidas para a instância primária, o exemplo a seguir altera o nível de promoção para a menor prioridade. Este exemplo especifica a opção `--manage-master-user-password` para gerar a senha mestra do usuário e gerenciá-la no Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

```
for i in 09 10
do
  aws rds modify-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
    --region $REGION --promotion-tier 15
done
```

Vamos supor que você queira usar as duas instâncias “maiores” apenas para as consultas que exijam mais recursos. Para fazer isso, primeiro é possível criar um endpoint somente leitura personalizado. Depois disso, é possível adicionar uma lista estática de membros para que o endpoint se conecte somente a essas instâncias de banco de dados. Essas instâncias já estão no nível de promoção mais baixo, tornando improvável que elas sequer sejam promovidas para a instância primária. Se uma delas for promovida para a instância primária, não será alcançável por meio desse endpoint porque especificamos o tipo `READER`, em vez do tipo `ANY`.

O exemplo a seguir demonstra os comandos de criação e modificação do endpoint, além da saída JSON de exemplo, mostrando o estado inicial e modificado do endpoint personalizado.

```
$ aws rds create-db-cluster-endpoint --region $REGION \
    --db-cluster-identifier custom-endpoint-demo \
    --db-cluster-endpoint-identifier big-instances --endpoint-type reader
{
    "EndpointType": "CUSTOM",
    "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "DBClusterEndpointIdentifier": "big-instances",
    "DBClusterIdentifier": "custom-endpoint-demo",
    "StaticMembers": [],
    "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
    "ExcludedMembers": [],
    "CustomEndpointType": "READER",
    "Status": "creating",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances"
}

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier big-instances \
    --static-members custom-endpoint-demo-09 custom-endpoint-demo-10 --region $REGION
{
    "EndpointType": "CUSTOM",
    "ExcludedMembers": [],
    "DBClusterEndpointIdentifier": "big-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
    "CustomEndpointType": "READER",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances",
    "StaticMembers": [
        "custom-endpoint-demo-10",
        "custom-endpoint-demo-09"
    ],
    "Status": "modifying",
    "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "DBClusterIdentifier": "custom-endpoint-demo"
}
```

O endpoint READER padrão do cluster pode se conectar a instâncias de banco de dados “pequenas” ou “grandes”, tornando impraticável prever a performance da consulta e a escalabilidade quando o cluster fica ocupado. Para dividir a workload claramente entre os conjuntos de instâncias de banco de dados, ignore o endpoint READER padrão e crie um segundo endpoint personalizado que se conecte a todas as outras instâncias de banco de dados. O exemplo a seguir faz isso criando um endpoint personalizado e adicionando uma lista de exclusões. Todas as outras instâncias de banco de dados adicionadas ao cluster depois serão adicionadas a esse endpoint automaticamente. O tipo ANY significa que esse endpoint está associado a oito instâncias no total: a instância primária e outras sete réplicas do Aurora. Se o exemplo tivesse usado o tipo READER, o endpoint personalizado só estaria associado às sete réplicas do Aurora.

```
$ aws rds create-db-cluster-endpoint --region $REGION --db-cluster-identifier custom-
endpoint-demo \
    --db-cluster-endpoint-identifier small-instances --endpoint-type any
{
    "Status": "creating",
    "DBClusterEndpointIdentifier": "small-instances",
    "CustomEndpointType": "ANY",
    "EndpointType": "CUSTOM",
    "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "StaticMembers": [],
    "ExcludedMembers": [],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:small-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY"
}

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier small-instances \
    --excluded-members custom-endpoint-demo-09 custom-endpoint-demo-10 --region $REGION
{
    "DBClusterEndpointIdentifier": "small-instances",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:c7tj4example:cluster-
endpoint:small-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY",
    "CustomEndpointType": "ANY",
    "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "EndpointType": "CUSTOM",
```

```
    "ExcludedMembers": [
        "custom-endpoint-demo-09",
        "custom-endpoint-demo-10"
    ],
    "StaticMembers": [],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "Status": "modifying"
}
```

O exemplo a seguir verifica o estado dos endpoints desse cluster. O cluster ainda tem o endpoint cluster original, com `EndPointType` de `WRITER`, que você continuaria usando nas operações de administração, ETL e outras de gravação. Ele ainda tem o endpoint `READER` original, que você não usaria porque cada conexão com ele pode ser direcionada para uma instância de banco de dados “pequena” ou “grande”. Os endpoints personalizados tornam esse comportamento previsível, com conexões garantidas para usar uma das instâncias de banco de dados “pequena” ou “grande” com base no endpoint especificado.

```
$ aws rds describe-db-cluster-endpoints --region $REGION
{
    "DBClusterEndpoints": [
        {
            "EndpointType": "WRITER",
            "Endpoint": "custom-endpoint-demo.cluster-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "Status": "available",
            "DBClusterIdentifier": "custom-endpoint-demo"
        },
        {
            "EndpointType": "READER",
            "Endpoint": "custom-endpoint-demo.cluster-ro-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "Status": "available",
            "DBClusterIdentifier": "custom-endpoint-demo"
        },
        {
            "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "CustomEndpointType": "ANY",
            "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:small-instances",
            "ExcludedMembers": [
                "custom-endpoint-demo-09",
```

```
                "custom-endpoint-demo-10"
            ],
            "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY",
            "DBClusterIdentifier": "custom-endpoint-demo",
            "StaticMembers": [],
            "EndpointType": "CUSTOM",
            "DBClusterEndpointIdentifier": "small-instances",
            "Status": "modifying"
        },
        {
            "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "CustomEndpointType": "READER",
            "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances",
            "ExcludedMembers": [],
            "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
            "DBClusterIdentifier": "custom-endpoint-demo",
            "StaticMembers": [
                "custom-endpoint-demo-10",
                "custom-endpoint-demo-09"
            ],
            "EndpointType": "CUSTOM",
            "DBClusterEndpointIdentifier": "big-instances",
            "Status": "available"
        }
    ]
}
```

Os exemplos finais demonstram como conexões de banco de dados sucessivas com os endpoints personalizados se ligam às várias instâncias de banco de dados no cluster do Aurora. O endpoint `small-instances` sempre se conecta às instâncias de banco de dados `db.r4.4xlarge`, que são hosts de números mais baixos nesse cluster.

```
$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+--------------------------+
| @@aurora_server_id       |
+--------------------------+
| custom-endpoint-demo-02 |
```

```
+-------------------------+

$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-07 |
+-------------------------+

$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-01 |
+-------------------------+
```

O endpoint `big-instances` sempre se conecta às instâncias de banco de dados `db.r4.16xlarge`, que são os dois hosts de números mais altos nesse cluster.

```
$ mysql -h big-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -u
 $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-10 |
+-------------------------+

$ mysql -h big-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -u
 $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-09 |
+-------------------------+
```

Classes de instâncias de banco de dados do Amazon Aurora

A classe de instância de banco de dados determina a capacidade computacional e de memória de uma instância de banco de dados do Amazon Aurora. A classe de instância de banco de dados da qual você precisa depende dos requisitos de memória e potência de processamento.

Uma classe de instância de banco de dados consiste no tipo de classe e no tamanho de instância de banco de dados. Por exemplo, db.r6g é um tipo de classe de instância de banco de dados otimizado para memória por processadores AWS Graviton2. No tipo de classe instância db.r6g, db.r6g.2xlarge é uma classe de instância de banco de dados. O tamanho dessa classe é 2xlarge.

Para ter mais informações sobre a definição de preço da classe de instância, consulte Definição de preço do Amazon RDS.

Para ter mais informações sobre tipos de classe de instância de banco de dados, mecanismos de banco de dados aceitos, Regiões da AWS compatíveis, ou especificações de hardware referentes a classes de instância de banco de dados, consulte as seções a seguir.

Tópicos

- Tipos de classe de instância de banco de dados
- Mecanismos de banco de dados compatíveis para classes de instância de banco de dados
- Determinar o suporte para classes de instância de bancos de dados nas Regiões da AWS
- Especificações de hardware para classes de instância de banco de dadospara o Aurora

Tipos de classe de instância de banco de dados

O Amazon Aurora é compatível com as classes de instância de banco de dados para os seguintes casos de uso:

- Aurora Serverless v2
- Otimizada para memória
- Performance expansível
- Optimized Reads

Para ter mais informações sobre os tipos de instância do Amazon EC2, consulte Tipos de instância na documentação do Amazon EC2.

Tipo de classe de instância do Aurora Serverless v2

O seguinte tipo de Aurora Serverless v2 está disponível:

- db.serverless: um tipo de classe de instância de banco de dados especial usada pelo Aurora Serverless v2. O Aurora ajusta dinamicamente os recursos de computação, memória e rede à medida que a workload muda. Para obter mais detalhes sobre uso, consulte Usar o Aurora Serverless v2.

Tipos de classe de instância otimizada para memória

A família X otimizada para memória comporta as seguintes classes de instância:

- db.x2g: classes de instância otimizada para aplicações com uso intensivo de memória e baseadas em processadores Graviton2 da AWS. Essas classes de instância oferecem custo baixo por GiB de memória.

  Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Graviton2 da AWS. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados.

A família R otimizada para memória é compatível com os seguintes tipos de classe de instância:

- db.r8g: classes de instância com processadores Graviton4 da AWS. Essas classes de instância são ideais para executar workloads com uso intensivo de memória. Essas instâncias oferecem tamanhos de instância maiores com até 3 vezes mais vCPUs e memória do que as instâncias db.r7g da sétima geração baseadas em Graviton3 da AWS. Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve.
- Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Graviton3 da AWS. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados.
- db.r7g: classes de instância com processadores Graviton3 da AWS. Essas classes de instância são ideais para executar workloads com uso intensivo de memória.

  Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Graviton3 da AWS. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados. Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve.

- db.r7i: classes de instância com processadores Intel Xeon escaláveis de quarta geração. Essas classes de instância são certificadas pela SAP e ideais para executar workloads com uso intenso de memória. Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Intel Xeon escaláveis de quarta geração. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados. Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve.
- db.r6g: classes de instância com processadores Graviton2 da AWS. Essas classes de instância são ideais para executar workloads com uso intensivo de memória Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve.
- Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Graviton2 da AWS. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados.
- db.r6id: classes de instância com processadores Intel Xeon escaláveis de terceira geração. Essas classes de instância são certificadas pela SAP e ideais para executar workloads com uso intenso de memória.
- db.r5: classes de instância otimizada para aplicações com uso intenso de memória. Essas classes de instância oferecem performance aprimorada de rede e do Amazon Elastic Block Store (Amazon EBS) . Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve.
- db.r4: essas classes de instância são compatíveis apenas com o Aurora MySQL 2.x e o Aurora PostgreSQL versões 11 e 12. Em relação a todos os clusters de banco de dados do Aurora que utilizam classes de instância de banco de dados db.r4, recomendamos fazer a atualização para uma classe de instância de geração posterior o quanto antes.

  As classes de instância db.r4 não estão disponíveis para a configuração de armazenamento do cluster do Aurora I/O-Optimized.

Tipos de classe de instância de performance expansível

Os seguintes tipos de classe de instância de banco de dados de performance expansível estão disponíveis:

- db.t4g: classes de instância de uso geral com processadores Graviton2 da AWS baseados em ARM. Essas classes de instância oferecem uma relação de performance e preço melhor do que as classes de instância de banco de dados de intermitência anteriores para um amplo conjunto

de workloads de uso geral. As instâncias db.t4g do Amazon RDS são configuradas para o modo Ilimitado. Isso significa que elas podem se expandir além da linha de base em uma janela de 24 horas por um custo adicional.

Você pode modificar uma instância de bancos de dados para usar uma das classes de instância de banco de dados com processadores Graviton2 da AWS. Para fazer isso, conclua as mesmas etapas de qualquer outra modificação da instância de banco de dados.

- db.t3: classes de instância que fornecem um nível básico de performance, com capacidade de intermitência para o uso total da CPU. As instâncias db.t3 são configuradas para o modo Ilimitado. Essas classes fornecem mais capacidade de computação do que as classes de instância anteriores, db.t2. Elas são desenvolvidas pelo Nitro System da AWS, uma combinação de hardware dedicado e hipervisor leve. Recomendamos só usar essas classes de instância para servidores de desenvolvimento e de teste ou outros servidores que não sejam de produção.
- db.t2 – classes de instância que fornecem um nível básico de performance, com capacidade de intermitência para o uso total da CPU. As instâncias db.t2 são configuradas para o modo Ilimitado. Recomendamos só usar essas classes de instância para servidores de desenvolvimento e de teste ou outros servidores que não sejam de produção.

  As classes de instância db.t2 não estão disponíveis para a configuração de armazenamento do cluster do Aurora I/O-Optimized.

> **Note**
>
> Recomendamos utilizar as classes de instância de banco de dados T somente para servidores de desenvolvimento, teste ou outros servidores que não sejam de produção. Para obter mais recomendações detalhadas sobre as classes de instância T, consulte Uso de classes de instância T para desenvolvimento e testes.

Para especificações de hardware para classes de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

Tipos de classe de instância do Optimized Reads

Os seguintes tipos de classe de instância do Optimized Reads disponíveis são:

- db.r8gd: classes de instância com processadores Graviton4. Essas classes de instância são ideais para executar workloads que fazem uso intenso de memória e oferecem armazenamento ao nível

do bloco SSD baseado em NVME local para aplicações que precisam de armazenamento local de alta velocidade e baixa latência. Elas oferecem memória máxima de 1,5 TiB e até 11,4 TB de armazenamento de SSD baseado em NVMe de conexão direta.

- db.r6gd: classes de instância com processadores Graviton2 da AWS. Essas classes de instância são ideais para executar workloads que fazem uso intenso de memória e oferecem armazenamento ao nível do bloco SSD baseado em NVME local para aplicações que precisam de armazenamento local de alta velocidade e baixa latência.
- db.r6id: classes de instância com processadores Intel Xeon escaláveis de terceira geração. Essas classes de instância são certificadas pela SAP e ideais para executar workloads com uso intenso de memória. Elas oferecem memória máxima de 1 TiB e até 7,6 TB de armazenamento SSD baseado em NVMe de conexão direta.

Mecanismos de banco de dados compatíveis para classes de instância de banco de dados

As tabelas a seguir mostram as classes de instâncias de banco de dados compatíveis com os mecanismos de banco de dados do Amazon Aurora.

db.serverless: classe de instância do Aurora Serverless v2 com ajuste de escala automático de capacidade.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.serverless	Consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.	Consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

db.x2g: classes de instância otimizada para memória desenvolvidas por processadores Graviton2 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.x2g.16xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.12xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.8xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.4xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.2xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e

Classe de instância	Aurora MySQL	Aurora PostgreSQL
		posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.x2g.large	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores

db.r6gd: classes de instâncias de Leituras otimizadas com processadores Graviton2 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r6gd.16xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6gd.12xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6gd.8xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6gd.4xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6gd.2xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6gd.xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores

db.r6i: classes de instância do Optimized Reads

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r6id.32xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores
db.r6id.32xlarge	Não	17.4 e posteriores, 16.1 e posteriores, 15.4 e posteriores e 14.9 e posteriores

db.r8g: classes de instância otimizada para memória com processadores Graviton4 da AWS

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r8g.48xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.24xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.16xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.12xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.8xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.4xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r8g.2xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r8g.large	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores

db.r8gd: classes de instâncias de leituras otimizadas com processadores Graviton4 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r8gd.48xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8gd.24xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8gd.16xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8gd.12xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r8gd.8xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8gd.4xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8gd.2xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores
db.r8g.xlarge	Não	17.4 e posteriores, 16.3 e posteriores, 15.7 e posteriores e 14.12 e posteriores

db.r7g: classes de instância otimizada para memória com processadores Graviton3 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r7g.16xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores
db.r7g.12xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores
db.r7g.8xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r7g.4xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores
db.r7g.2xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores
db.r7g.xlarge	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores
db.r7g.large	Versões 2.12.0 e posteriores, 3.03.1 e posteriores	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.7 e posteriores e 13.10 e posteriores

db.r7i: classes de instância otimizada para memória otimizadas para memória

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r7i.48xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.24xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.16xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r7i.12xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.8xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.4xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.2xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.xlarge	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores
db.r7i.large	3.08.0 e posterior	17.4 posteriores, 16.3 posteriores, 15.7 posteriores, 14.12 posteriores e 13.15 posteriores

db.r6g: classes de instância otimizada para memória com processadores Graviton2 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r6g.16xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r6g.12xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.r6g.8xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.r6g.4xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.r6g.2xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.r6g.xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores
db.r6g.large	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.8 e posteriores, 11.9 e 11.12 e posteriores

db.r6i: classes de instância otimizada para memória

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r6i.32xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.24xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.16xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.12xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.8xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.4xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.2xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e

Classe de instância	Aurora MySQL	Aurora PostgreSQL
		posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.xlarge	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores
db.r6i.large	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.5 e posteriores e 12.9 e posteriores

db.r5: classes de instância otimizada para memória

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r5.24xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.16xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.12xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.8xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.4xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.2xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r5.xlarge	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente
db.r5.large	Todas as versões disponíveis atualmente	Todas as versões disponíveis atualmente

db.r4 – classes de instância otimizada para memória

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.r4.16xlarge	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis
db.r4.8xlarge	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis
db.r4.4xlarge	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis
db.r4.2xlarge	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis
db.r4.xlarge	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis
db.r4.large	Todas as versões 2.x; não compatíveis em versões 3.x.	Todas as versões 11 e 12 compatíveis

db.t4g: classes de instância de desempenho expansível com processadores Graviton2 da AWS.

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.t4g.2xlarge	Não	Não
db.t4g.xlarge	Não	Não
db.t4g.large	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.7 e posteriores e 11.12 e posteriores
db.t4g.medium	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.7 e posteriores e 11.12 e posteriores
db.t4g.small	Não	Não

db.t3: classes de instância expansível

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.t3.2xlarge	Não	Não
db.t3.xlarge	Não	Não
db.t3.large	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e posteriores, 13.3 e posteriores, 12.7 e posteriores e 11.12 e posteriores
db.t3.medium	Todas as versões disponíveis atualmente	17.4 e posteriores, 16.1 e posteriores, 15.2 e posteriores, 14.3 e

Classe de instância	Aurora MySQL	Aurora PostgreSQL
		posteriores, 13.3 e posteriores, 12.7 e posteriores e 11.12 e posteriores
db.t3.small	Todas as versões 2.x; não compatíveis em versões 3.x.	Não
db.t3.micro	Não	Não

db.t2: classes de instância expansível

Classe de instância	Aurora MySQL	Aurora PostgreSQL
db.t2.medium	Todas as versões 2.x; não compatíveis em versões 3.x.	Não
db.t2.small	Todas as versões 2.x; não compatíveis em versões 3.x.	Não

Determinar o suporte para classes de instância de bancos de dados nas Regiões da AWS

Para determinar as classes de instância de bancos de dados compatíveis com cada mecanismo de banco de dados em uma Região da AWSespecífica, você pode usar uma das várias abordagens. Você pode usar o Console de gerenciamento da AWS, a página Definição de preço do Amazon RDS ou o comando describe-orderable-db-instance-options da AWS CLI.

> **Note**
>
> Quando você executa operações com a Console de gerenciamento da AWS, ela mostra automaticamente as classes de instância de bancos de dados compatíveis com um mecanismo de banco de dados específico, uma versão do mecanismo de banco de dados e

a Região da AWS. São exemplos de operação que você pode realizar: criação e modificação de uma instância de banco de dados.

Sumário

- Usar a página de definição de preço do Amazon RDS para determinar o suporte para classe de instância de banco de dados em Regiões da AWS
- Usar a AWS CLI para determinar o suporte para classe de instância de banco de dados em Regiões da AWS
  - Listando as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS
  - Listagem das versões do mecanismo de banco de dados que oferecem suporte a uma classe de instância de bancos de dados específica em uma Região da AWS

Usar a página de definição de preço do Amazon RDS para determinar o suporte para classe de instância de banco de dados em Regiões da AWS

É possível usar a página Definição de preço do Amazon Aurora para determinar as classes de instância de bancos de dados compatíveis com cada mecanismo de bancos de dados em uma Região da AWS específica.

Para usar a página de definição de preço para determinar as classes de instância de Bancos de Dados compatíveis com cada mecanismo em uma região

1. Acesse Definição de preço do Amazon Aurora.
2. Escolha um mecanismo do Amazon Aurora na seção Calculadora de Preços da AWS.
3. Em Escolher uma região, escolha uma Região da AWS.
4. Em Opção de configuração de cluster, escolha uma opção de configuração.
5. Use a seção de instâncias compatíveis para ver as classes de instância de banco de dados aceitas.
6. (Opcional) Escolha outras opções na calculadora e selecione Salvar e visualizar resumo ou Salvar e adicionar serviço.

Usar a AWS CLI para determinar o suporte para classe de instância de banco de dados em Regiões da AWS

Você pode usar a AWS CLI para determinar quais classes de instância de bancos de dados são compatíveis com versões específicas de mecanismos de banco de dados em uma Região da AWS.

Para usar os exemplos da AWS CLI a seguir, insira valores válidos para o mecanismo de banco de dados, a versão do mecanismo de banco de dados, a classe de instância de bancos de dados e a Região da AWS. A tabela a seguir mostra os valores válidos do mecanismo de banco de dados.

Nome do mecanismo	Valor do mecanismo nos comandos CLI	Mais informações sobre as versões
Aurora compatível com o MySQL 5.7 e 8.0	`aurora-mysql`	Atualizações feitas no mecanismo de banco de dados da versão 2 do Amazon Aurora MySQL e Atualizações feitas no mecanismo de banco de dados da versão 3 do Amazon Aurora MySQL nas Notas de lançamento do Aurora MySQL
Aurora PostgreSQL	`aurora-postgresql`	Notas de lançamento do Aurora PostgreSQL

Para obter informações sobre nomes de Região da AWS, consulte AWSRegiões de.

Os exemplos a seguir demonstram como determinar o suporte de classe de instância de bancos de dados em uma Região da AWS usando o comando da AWS CLI describe-orderable-db-instance-options.

Tópicos

- Listando as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS
- Listagem das versões do mecanismo de banco de dados que oferecem suporte a uma classe de instância de bancos de dados específica em uma Região da AWS

Listando as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS

Para listar as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS, execute o comando a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 \
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region region
```

Para Windows:

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 ^
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region region
```

A saída também mostra os modos de mecanismo compatíveis com cada classe de instância de banco de dados.

Por exemplo, o comando a seguir lista as classes de instância de banco de dados compatíveis com a versão 13.6 do mecanismo de banco de dados do Aurora PostgreSQL no Leste dos EUA (N. da Virgínia).

Para Linux, macOS ou Unix:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 \
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region us-east-1
```

Para Windows:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 ^
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}"  ^
    --output table ^
    --region us-east-1
```

Listagem das versões do mecanismo de banco de dados que oferecem suporte a uma classe de instância de bancos de dados específica em uma Região da AWS

Para listar as versões do mecanismo de banco de dados compatíveis com uma classe de instância de bancos de dados específica em uma Região da AWS, execute o comando a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class \
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region region
```

Para Windows:

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class ^
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region region
```

A saída também mostra os modos de mecanismo compatíveis com cada versão do mecanismo de banco de dados.

Por exemplo, o comando a seguir lista as versões do mecanismo de banco de dados do mecanismo de banco de dados do Aurora PostgreSQL que oferecem suporte para a classe de instância de bancos de dados db.r5.large no Leste dos EUA (N. da Virgínia).

Para Linux, macOS ou Unix:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.r7g.large \
```

```
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region us-east-1
```

Para Windows:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.r7g.large ^
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region us-east-1
```

Especificações de hardware para classes de instância de banco de dadospara o Aurora

Na tabela desta seção, é possível encontrar detalhes de hardware sobre as classes de instância de banco de dados do Amazon RDS para o Aurora.

Para obter informações sobre o suporte a mecanismos de banco de dados do Aurora para cada classe de instância de banco de dados, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.

Tópicos

- Terminologia de hardware para classes de instância de banco de dadospara o Aurora
- Especificações de hardware para as classes de instância otimizada para memória
- Especificações de hardware para as classes de instância de desempenho expansível

Terminologia de hardware para classes de instância de banco de dadospara o Aurora

A terminologia a seguir é usada para descrever especificações de hardware para classes de instâncias de banco de dados:

vCPU

O número de unidades de processamento central (CPUs) virtuais. Uma CPU virtual é uma unidade de capacidade que pode ser usada para comparar classes de instância de banco de

dados. Em vez de comprar ou alugar um determinado processador para usar durante vários meses ou anos, você está alugando a capacidade de acordo com a hora. Nosso objetivo é tornar uma quantidade consistente e específica da capacidade da CPU disponível, dentro dos limites de hardware subjacente real.

ECU

A medida relativa da potência de todo o processamento de uma instância do Amazon EC2. Para que os desenvolvedores tenham facilidade ao comparar a capacidade da CPU entre diferentes classes de instância, definimos uma unidade computacional do Amazon EC2. A quantidade de CPU que é alocada para determinada instância é expressa em termos dessas Unidades computacionais do EC2. No momento, uma ECU fornece a capacidade de CPU equivalente a de um processador 2007 Xeon ou 2007 Opteron de 1,0–1,2 GHz.

Memória (GiB)

A memória RAM, em gibibytes, alocada à instância de banco de dados. Geralmente, há uma proporção consistente entre a memória e a vCPU. Por exemplo, considere a classe de instância db.r4, que possui uma proporção entre memória e vCPU semelhante à da classe de instância db.r5. No entanto, para a maioria dos casos de uso, a classe de instância db.r5 fornece uma performance melhor e mais consistente do que a classe de instância db.r4.

Máx. Largura de banda EBS (Mbps)

A largura de banda EBS máxima em megabits por segundo. Divida em oito para obter a taxa de transferência esperada em megabytes por segundo.

> **Note**
>
> Essa figura refere-se à largura de banda de E/S para armazenamento local na instância de banco de dados. Ela não se aplica à comunicação com o volume do cluster do Aurora.

Largura de banda de rede

A velocidade da rede em relação a outras classes de instância de banco de dados.

Para obter informações sobre como usar as métricas do Amazon CloudWatch para monitorar o throughput de sua instância de banco de dados do Aurora, consulte Avaliação do uso de instâncias de banco de dados para Aurora MySQL com métricas do Amazon CloudWatch e Avaliação do uso de instâncias de banco de dados para Aurora PostgreSQL com métricas do CloudWatch.

Especificações de hardware para as classes de instância otimizada para memória

As tabelas a seguir mostram as especificações de computação, memória, armazenamento e largura de banda para as classes de instância otimizada para memória.

db.x2g: classes de instância otimizada para memória com processadores Graviton2 da AWS.

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.x2g.16xlarge	64	—	1024	Otimizada somente para EBS	19.000	25
db.x2g.12xlarge	48	—	768	Otimizada somente para EBS	14.250	20
db.x2g.8xlarge	32	—	512	Otimizada somente para EBS	9.500	12
db.x2g.4xlarge	16	—	256	Otimizada somente para EBS	4.750	Até 10
db.x2g.2xlarge	8	—	128	Otimizada somente para EBS	Até 4.750	Até 10
db.x2g.xlarge	4	—	64	Otimizada somente para EBS	Até 4.750	Até 10
db.x2g.large	2	—	32	Otimizada somente para EBS	Até 4.750	Até 10

db.r8gd: classes de instância otimizada para memória com processadores Graviton2 da AWS e armazenamento de SSD.

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r8gd.48xlarge	192	—	1536	SSD de NVMe de 6 x 1.900	40.000	50
db.r8gd.24xlarge	96	—	768	SSD de NVMe de 3 x 1.900	30.000	40
db.r8gd.16xlarge	64	—	512	SSD de NVMe de 2 x 1.900	20.000	30
db.r8gd.12xlarge	48	—	384	SSD de NVMe de 3 x 950	15.000	22,5
db.r8gd.8xlarge	32	—	256	SSD de NVMe de 1 x 1.900	10.000	15
db.r8gd.4xlarge	16	—	128	SSD de NVMe de 1 x 950	Até 10 mil	Até 15
db.r8gd.2xlarge	8	—	64	SSD de NVMe de 1 x 474	Até 10 mil	Até 15
db.r8gd.xlarge	4	—	32	SSD de NVMe de 1 x 237	Até 10 mil	Até 12,5
db.r8gd.large	2	—	16	SSD de NVMe de 1 x 118	Até 10 mil	Até 12,5

db.r8g: classes de instância otimizada para memória com processadores Graviton4 da AWS

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r8g.48xlarge	192	—	1536	Otimizada somente para EBS	40.000	50
db.r8g.24xlarge	96	—	768	Otimizada somente para EBS	30.000	40
db.r8g.16xlarge	64	—	512	Otimizada somente para EBS	20.000	30
db.r8g.12xlarge	48	—	384	Otimizada somente para EBS	15.000	22,5
db.r8g.8xlarge	32	—	256	Otimizada somente para EBS	10.000	15
db.r8g.4xlarge	16	—	128	Otimizada somente para EBS	Até 10 mil	Até 15
db.r8g.2xlarge	8	—	64	Otimizada somente para EBS	Até 10 mil	Até 15
db.r8g.xlarge	4	—	32	Otimizada somente para EBS	Até 10 mil	Até 12,5

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r8g.large	2	—	16	Otimizada somente para EBS	Até 10 mil	Até 12,5

db.r7i: classes de instância otimizada para memória com processadores Intel Xeon escaláveis de quarta geração

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r7i.48xlarge	192	—	1536	Otimizada somente para EBS	40.000	50
db.r7i.24xlarge	96	—	768	Otimizada somente para EBS	30.000	37.5
db.r7i.16xlarge	64	—	512	Otimizada somente para EBS	20.000	25
db.r7i.12xlarge	48	—	384	Otimizada somente para EBS	15.000	18,75
db.r7i.8xlarge	32	—	256	Otimizada somente para EBS	10.000	12,5

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r7i.4xlarge	16	—	128	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r7i.2xlarge	8	—	64	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r7i.xlarge	4	—	32	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r7i.large	2	—	16	Otimizada somente para EBS	Até 10 mil	Até 12,5

db.r7g: classes de instância otimizada para memória com processadores Graviton3 da AWS

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r7g.16xlarge	64	—	512	Otimizada somente para EBS	20.000	30
db.r7g.12xlarge	48	—	384	Otimizada somente para EBS	15.000	22,5

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r7g.8xlarge	32	—	256	Otimizada somente para EBS	10.000	15
db.r7g.4xlarge	16	—	128	Otimizada somente para EBS	Até 10 mil	Até 15
db.r7g.2xlarge	8	—	64	Otimizada somente para EBS	Até 10 mil	Até 15
db.r7g.xlarge	4	—	32	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r7g.large	2	—	16	Otimizada somente para EBS	Até 10 mil	Até 12,5

db.r6id: classes de instância otimizada para memória com processadores escaláveis Intel Xeon de 3ª geração

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6id.32xlarge	128	—	1,024	SSD de NVMe de 4 x 1.900	40.000	50

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6id.32xlarge	96	—	768	SSD de NVMe de 4 x 1.425	30.000	37.5

db.r6gd: classes de instância otimizada para memória com processadores Graviton2 da AWS e armazenamento de memória SSD

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6gd.16xlarge	64	—	512	SSD de NVMe de 2 x 1.900	19.000	25
db.r6gd.12xlarge	48	—	384	SSD de NVMe de 2 x 1.425	13.500	20
db.r6gd.8xlarge	32	—	256	SSD de NVMe de 1 x 1.900	9.000	12
db.r6gd.4xlarge	16	—	128	SSD de NVMe de 1 x 950	4.750	Até 10
db.r6gd.2xlarge	8	—	64	SSD de NVMe de 1 x 474	Até 4.750	Até 10
db.r6gd.xlarge	4	—	32	SSD de NVMe de 1 x 237	Até 4.750	Até 10

db.r6g: classes de instância otimizada para memória com processadores Graviton2 da AWS

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6g.16xlarge	64	—	512	Otimizada somente para EBS	19.000	25
db.r6g.12xlarge	48	—	384	Otimizada somente para EBS	13.500	20
db.r6g.8xlarge	32	—	256	Otimizada somente para EBS	9.000	12
db.r6g.4xlarge	16	—	128	Otimizada somente para EBS	4.750	Até 10
db.r6g.2xlarge	8	—	64	Otimizada somente para EBS	Até 4.750	Até 10
db.r6g.xlarge	4	—	32	Otimizada somente para EBS	Até 4.750	Até 10
db.r6g.large	2	—	16	Otimizada somente para EBS	Até 4.750	Até 10

db.r6i: classes de instância otimizada para memória com processadores Intel Xeon escaláveis de terceira geração

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6i.32xlarge	128	—	1,024	Otimizada somente para EBS	40.000	50
db.r6i.24xlarge	96	—	768	Otimizada somente para EBS	30.000	37.5
db.r6i.16xlarge	64	—	512	Otimizada somente para EBS	20.000	25
db.r6i.12xlarge	48	—	384	Otimizada somente para EBS	15.000	18,75
db.r6i.8xlarge	32	—	256	Otimizada somente para EBS	10.000	12,5
db.r6i.4xlarge	16	—	128	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r6i.2xlarge	8	—	64	Otimizada somente para EBS	Até 10 mil	Até 12,5
db.r6i.xlarge	4	—	32	Otimizada somente para EBS	Até 10 mil	Até 12,5

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r6i.large	2	—	16	Otimizada somente para EBS	Até 10 mil	Até 12,5

db.r5: classes de instância otimizada para memória

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r5.24xlarge	96	347	768	Otimizada somente para EBS	19.000	25
db.r5.16xlarge	64	264	512	Otimizada somente para EBS	13.600	20
db.r5.12xlarge	48	173	384	Otimizada somente para EBS	9.500	12
db.r5.8xlarge	32	132	256	Otimizada somente para EBS	6.800	10
db.r5.4xlarge	16	71	128	Otimizada somente para EBS	4.750	Até 10

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r5.2xlarge	8	38	64	Otimizada somente para EBS	Até 4.750	Até 10
db.r5.xlarge	4	19	32	Otimizada somente para EBS	Até 4.750	Até 10
db.r5.large	2	10	16	Otimizada somente para EBS	Até 4.750	Até 10

db.r4: classes de instância otimizada para memória com processadores Intel Xeon Platinum e armazenamento SSD

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r4.16xlarge	64	195	488	Otimizada somente para EBS	14.000	25
db.r4.8xlarge	32	99	244	Otimizada somente para EBS	7.000	10
db.r4.4xlarge	16	53	122	Otimizada somente para EBS	3.500	Até 10

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.r4.2xlarge	8	27	61	Otimizada somente para EBS	1.700	Até 10
db.r4.xlarge	4	13,5	30,5	Otimizada somente para EBS	850	Até 10
db.r4.large	2	7	15,25	Otimizada somente para EBS	425	Até 10

Especificações de hardware para as classes de instância de desempenho expansível

As tabelas a seguir mostram as especificações de computação, memória, armazenamento e largura de banda para as classes de instância de desempenho expansível.

db.t4g: classes de instância de desempenho expansível com processadores Graviton2 da AWS

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.t4g.large	2	—	8	Otimizada somente para EBS	Até 2.780	Até 5
db.t4g.medium	2	—	4	Otimizada somente para EBS	Até 2.085	Até 5

db.t3: classes de instância expansível

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.t3.large	2	Variáv	8	Otimizada somente para EBS	Até 2.048	Até 5
db.t3.medium	2	Variáv	4	Otimizada somente para EBS	Até 1.536	Até 5
db.t3.small	2	Variáv	2	Otimizada somente para EBS	Até 1.536	Até 5

db.t2: classes de instância expansível

Classe de instância	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GiB)	Máx. Largura de banda EBS (Mbps)	Largura de banda da rede (Gbps)
db.t2.medium	2	Variáv	4	Somente EBS	—	Moderada
db.t2.small	1	Variáv	2	Somente EBS	—	Baixo

Armazenamento do Amazon Aurora

A seguir, saiba mais sobre o subsistema de armazenamento do Aurora. O Aurora usa uma arquitetura de armazenamento distribuído e compartilhado que é um fator importante em termos de performance, escalabilidade e confiabilidade para clusters do Aurora.

Tópicos

- Visão geral do armazenamento do Amazon Aurora
- O que o volume de cluster contém
- Configurações de armazenamento para clusters de banco de dados do Amazon Aurora
- Como o armazenamento do Aurora redimensiona automaticamente
- Como o armazenamento de dados do Aurora é faturado

Visão geral do armazenamento do Amazon Aurora

Os dados do Aurora são armazenados no volume de cluster, que é um volume virtual único que usa Solid State Drives (SSDs – Unidades de estado sólido). Um volume de cluster consiste em cópias dos dados em várias zonas de disponibilidade em uma única região da AWS. Como os dados são replicados automaticamente nas Zonas de disponibilidade, seus dados são resilientes, havendo menos possibilidade de perda de dados. Essa replicação também garante que o banco de dados esteja disponível durante um failover. Ele faz isso porque as cópias de dados já existem nas outras zonas de disponibilidade e continuam atendendo a solicitações de dados para as instâncias de banco de dados no cluster de banco de dados. A quantidade de replicação independe do número de instâncias de banco de dados no cluster.

O Aurora usa armazenamento local separado para arquivos temporários não persistentes. Isso inclui arquivos que são usados para fins como classificar grandes conjuntos de dados durante o processamento de consultas e criação de índices. Para obter mais informações, consulte Limites de armazenamento temporário para o Aurora MySQL e Limites de armazenamento temporário para o Aurora PostgreSQL.

O que o volume de cluster contém

O volume de cluster do Aurora contém todos os dados do usuário, os objetos de esquema e os metadados internos como as tabelas do sistema e o log binário. Por exemplo, o Aurora armazena todas as tabelas, os índices, os Binary Large Objects (BLOBs – Grandes objetos binários), os procedimentos armazenados etc. para um cluster do Aurora no volume do cluster.

A arquitetura de armazenamento compartilhado do Aurora torna os dados independentes das instâncias de banco de dados no cluster. Por exemplo, adicione uma instância de banco de dados rapidamente porque o Aurora não cria uma nova cópia dos dados da tabela. Em vez disso, a instância de banco de dados se conecta ao volume compartilhado que já contém todos os dados. Remova uma instância de banco de dados de um cluster sem remover dados subjacentes do cluster. O Aurora só remove os dados quando você exclui todo o cluster.

Configurações de armazenamento para clusters de banco de dados do Amazon Aurora

O Amazon Aurora tem duas configurações de armazenamento do cluster de banco de dados:

- Aurora I/O-Optimized: melhor preço/performance e previsibilidade para aplicações com uso intenso de E/S. Você paga somente pelo uso e armazenamento de seus clusters de banco de dados, sem custos adicionais pelas operações de E/S de leitura e gravação.

  O Aurora I/O-Optimized é a melhor opção quando seus gastos com E/S são 25% ou mais do gasto total com o banco de dados do Aurora.

  Você pode escolher Aurora I/O-Optimized ao criar ou modificar um cluster de banco de dados com uma versão de mecanismo de banco de dados compatível com a configuração do cluster do Aurora I/O-Optimized. É possível mudar do Aurora I/O-Optimized para o Aurora Standard a qualquer momento.
- Aurora Standard: preços econômicos para muitas aplicações com uso moderado de E/S. Além do uso e do armazenamento dos clusters de banco de dados, você também paga uma taxa padrão por um milhão de solicitações para operações de E/S.

  O Aurora Standard é a melhor opção quando seus gastos com E/S são inferiores a 25% dos gastos totais com o banco de dados do Aurora.

  Você pode mudar de Aurora Standard para Aurora I/O-Optimized uma vez a cada 30 dias. Quando você alterna entre as opções de armazenamento Aurora Standard e Aurora I/O-Optimized para instâncias de banco de dados não baseadas em NVMe, não há tempo de inatividade. No entanto, para instâncias de banco de dados baseadas em NVMe, alternar entre as opções de armazenamento Aurora I/O-Optimized e Aurora Standard requer a reinicialização do mecanismo de banco de dados, o que pode causar um breve período de inatividade.

Para obter mais informações sobre Região da AWS e compatibilidade de versão, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com configurações de armazenamento em cluster.

Para obter mais informações sobre o preço referente às configurações de armazenamento do Amazon Aurora, consulte Preços do Amazon Aurora.

Para receber informações sobre como escolher a configuração de armazenamento ao criar um cluster de banco de dados, consulte Criar um cluster de banco de dados. Para receber informações sobre como modificar a configuração de armazenamento de um cluster de banco de dados, consulte Configurações do Amazon Aurora.

Como o armazenamento do Aurora redimensiona automaticamente

Os volumes de cluster do Aurora crescem automaticamente à medida em que aumenta a quantidade de dados em seu banco de dados. Para ter informações sobre os tamanhos máximos de volume do cluster do Aurora para cada versão de mecanismo, consulte Limites de tamanho do Amazon Aurora. Essa escalabilidade automática de armazenamento é combinada com um subsistema de armazenamento altamente distribuído e de alta performance. Isso faz do Aurora uma boa escolha para seus dados corporativos importantes quando seus principais objetivos são confiabilidade e alta disponibilidade.

Para exibir o status do volume, consulte Exibir o status do volume para um cluster de banco de dados Aurora MySQL ou Exibir o status do volume para um cluster de bancos de dados Aurora PostgreSQL. Para formas de equilibrar os custos de armazenamento com outras prioridades, Escalabilidade de armazenamento descreve como monitorar as métricas do Amazon Aurora `AuroraVolumeBytesLeftTotal` e `VolumeBytesUsed` em CloudWatch.

Quando os dados do Aurora são removidos, o espaço alocado para esses dados é liberado. Exemplos de remoção de dados incluem descartar ou truncar uma tabela. Essa redução automática no uso do armazenamento ajuda a minimizar as cobranças de armazenamento.

> **Note**
>
> Os limites de armazenamento e o comportamento de redimensionamento dinâmico discutidos aqui se aplicam a tabelas persistentes e outros dados armazenados no volume do cluster.
>
> Para Aurora PostgreSQL, os dados das tabelas temporárias são armazenados na instância de banco de dados local.
>
> Para Aurora MySQL versão 2, os dados da tabela temporária são armazenados por padrão no volume do cluster para instâncias gravadoras e no armazenamento local para instâncias leitoras. Para obter mais informações, consulte Mecanismo de armazenamento para tabelas temporárias em disco.
>
> Para Aurora MySQL versão 3, os dados da tabela temporária são armazenados na instância de banco de dados local ou no volume do cluster. Para obter mais informações, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3.

O tamanho máximo das tabelas temporárias que residem no armazenamento local é limitado pelo tamanho máximo do armazenamento local da instância de banco de dados. O tamanho do armazenamento local depende da classe de instância usada. Para obter mais informações, consulte Limites de armazenamento temporário para o Aurora MySQL e Limites de armazenamento temporário para o Aurora PostgreSQL.

Alguns recursos de armazenamento, como o tamanho máximo de um volume de cluster e o redimensionamento automático quando os dados são removidos, dependem da versão do Aurora do cluster. Para obter mais informações, consulte Escalabilidade de armazenamento. Também é possível aprender como evitar problemas de armazenamento e como monitorar o armazenamento alocado e o espaço livre no cluster.

Como o armazenamento de dados do Aurora é faturado

Embora um volume de cluster do Aurora possa aumentar até 256 tebibytes (TiB) em versões de mecanismo específicas, você só receberá cobrança pelo espaço que usar em um volume de cluster do Aurora. Em versões anteriores do Aurora, o volume do cluster podia reutilizar o espaço liberado quando você removia dados, mas o espaço de armazenamento alocado nunca diminuía. Agora, quando os dados do Aurora são removidos, como ao excluir uma tabela ou banco de dados, o espaço total alocado diminui em uma quantidade equivalente. Assim, é possível reduzir as cobranças de armazenamento excluindo tabelas, índices, bancos de dados etc., que não são mais necessários.

Tip

Para versões anteriores sem o recurso de redimensionamento dinâmico, redefinir o uso de armazenamento para um cluster envolvia fazer um despejo lógico e restaurar para um novo cluster. Essa operação pode levar muito tempo para um volume substancial de dados. Se você encontrar essa situação, considere atualizar seu cluster para uma versão que ofereça suporte ao redimensionamento dinâmico de volume.

Para obter informações sobre quais versões do Aurora oferecem suporte ao redimensionamento dinâmico e sobre como minimizar as cobranças de armazenamento monitorando o uso do armazenamento para seu cluster, consulte Escalabilidade de armazenamento. Para obter informações sobre faturamento de armazenamento de backup do Aurora, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora. Para obter informações da definição de preço sobre o armazenamento de dados do Aurora, consulte Preços do Amazon RDS para Aurora.

Confiabilidade do Amazon Aurora

O Aurora foi projetado para ser confiável, durável e tolerante a falhas. É possível arquitetar o cluster de banco de dados do Aurora para aumentar a disponibilidade por meio de ações, como adicionar réplicas do Aurora e instalá-las em zonas de disponibilidade diferentes. Além disso, o Aurora inclui vários recursos automáticos que fazem dele uma solução de banco de dados confiável.

Tópicos

- Reparo automático de armazenamento
- Cache de página perdurável
- Recuperação após reinicializações não planejadas

Reparo automático de armazenamento

Como o Aurora mantém várias cópias de seus dados em três zonas de disponibilidade, a chance de perder dados devido a uma falha no disco é muito baixa. O Aurora detecta automaticamente as falhas nos volumes de disco que compõem o volume do cluster. Quando um segmento de um volume de disco falha, o Aurora repara imediatamente o segmento. Quando o Aurora repara o segmento do disco, ele usa os dados nos outros volumes que compõem o volume do cluster para garantir que os dados no segmento reparado sejam atuais. Como resultado, o Aurora evita a perda de dados e reduz a necessidade de executar uma restauração pontual para recuperação de uma falha de disco.

Cache de página perdurável

No Aurora, o cache de página de cada instância de banco de dados é gerenciado em um processo separado do banco de dados, permitindo que o cache de página perdure independentemente do banco de dados. (O cache de página também é chamado de grupo de buffer do InnoDB no Aurora MySQL e cache de buffer no Aurora PostgreSQL.)

No evento improvável de uma falha no banco de dados, o cache de página permanece na memória, o que mantém as páginas de dados atuais “em atividade” no cache de página quando o banco de dados é reiniciado. Isso gera um ganho de performance, evitando a necessidade de executar operações de E/S de leitura para “ativar” o cache de página.

No caso do Aurora MySQL, o comportamento do cache de página na reinicialização e no failover é o seguinte:

- É possível reinicializar a instância do gravador sem reinicializar as instâncias do leitor.
  - Se as instâncias do leitor não forem reinicializadas simultaneamente com a instância do gravador, elas não perderão os caches de página.
  - Se as instâncias do leitor não forem reinicializadas simultaneamente com a instância do gravador, elas perderão os caches de página.
- Quando uma instância do leitor é reinicializada, os caches de página no gravador e as instâncias do leitor perduram.
- Quando ocorre uma falha no cluster de banco de dados, o efeito é semelhante a quando uma instância do gravador é reinicializada. Na nova instância do gravador (anteriormente, a instância do leitor), o cache de página perdura, mas não ocorre o mesmo na instância do leitor (anteriormente, a instância do gravador).

No caso do Aurora PostgreSQL, é possível usar o gerenciamento de cache de cluster para preservar o cache de página de uma instância de leitor designada que se torna a instância do gravador após o failover. Para ter mais informações, consulte Recuperação rápida após failover com o gerenciamento de cache do cluster para o Aurora PostgreSQL.

Recuperação após reinicializações não planejadas

O Aurora é projetado para se recuperar de uma reinicialização não planejada quase instantaneamente e continuar a fornecer os dados de sua aplicação sem o log binário. O Aurora se recupera de forma assíncrona em threads paralelos, de maneira que o banco de dados seja aberto e fique disponível imediatamente após a reinicialização não planejada.

Para ter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora e Otimizações para reduzir o tempo de reinicialização do banco de dados.

Veja a seguir, considerações para log binário e recuperação após reinicializações não planejadas no Aurora MySQL:

- Ativar o log binário no Aurora afeta diretamente o tempo de recuperação após uma reinicialização não planejada, porque isso força a instância de banco de dados a executar uma recuperação de log binário.
- O tipo de log binário usado afeta o tamanho e a eficiência dele. Para a mesma quantidade de atividade do banco de dados, alguns formatos oferecem mais informações que outros nos logs binários. As seguintes configurações para o parâmetro `binlog_format` resultam em diferentes quantidades de dados de log:

- `ROW` – a maior quantidade de dados de log
- `STATEMENT` – a menor quantidade de dados de log
- `MIXED` – uma quantidade moderada de dados de log, que geralmente fornece a melhor combinação de integridade e performance dos dados

A quantidade de dados do log binário afeta o tempo de recuperação. Se houver mais dados registrados nos logs binários, a instância de banco de dados processará mais dados durante a recuperação, aumentando o tempo necessário.

- Para reduzir a sobrecarga computacional e melhorar os tempos de recuperação com o registro em log binário, é possível usar o log binário aprimorado. O log binário aprimorado melhora o tempo de recuperação do banco de dados em até 99%. Para ter mais informações, consulte Configurar o log binário avançado para Aurora MySQL.
- O Aurora não precisa dos logs binários para replicar dados dentro de um cluster de banco de dados nem para executar a restauração de point-in-time (PITR).
- Se o log binário não for necessário para replicação externa (ou um fluxo de log binário externo), recomendamos que você defina o parâmetro `binlog_format` como `OFF` para desativá-lo. Isso reduz o tempo de recuperação.

Para obter mais informações sobre o registro em log binário e a replicação no Aurora, consulte Replicação com o Amazon Aurora. Para obter mais informações sobre as implicações dos diferentes tipos de replicação do MySQL, consulte Vantagens e desvantagens da replicação baseada em instrução e baseada em linha na documentação do MySQL.

Segurança do Amazon Aurora

A segurança do Amazon Aurora é gerenciada em três níveis:

- Para controlar quem pode realizar ações de gerenciamento do Amazon RDS em clusters de banco de dados e instâncias de banco de dados do Aurora, você usa o AWS Identity and Access Management (IAM). Ao se conectar à AWS usando credenciais do IAM, sua conta da AWSdeve ter políticas do IAM que concedam as permissões necessárias para executar operações de gerenciamento do Amazon RDS. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

Se estiver usando o IAM para acessar o console do Amazon RDS, será necessário primeiramente fazer login no Console de gerenciamento da AWS com as credenciais de usuário, depois acessar o console do Amazon RDS em https://console.aws.amazon.com/rds.

- Os clusters de banco de dados do Aurora devem ser criados em uma nuvem privada virtual (VPC) com base no serviço da Amazon VPC. Para controlar quais dispositivos e instâncias do Amazon EC2 podem abrir conexões para o endpoint e a porta da instância de banco de dados dos clusters de banco de dados do Aurora em uma VPC, use um grupo de segurança da VPC. É possível estabelecer essas conexões de endpoint e porta usando Transport Layer Security (TLS)/Secure Sockets Layer (SSL). Além disso, as regras de firewall em sua empresa podem controlar se dispositivos sendo executados nela podem abrir conexões em uma instância de banco de dados. Para obter mais informações sobre VPCs, consulte Amazon VPC e Amazon Aurora.
- Para autenticar os logins e as permissões de um cluster de banco de dados do Amazon Aurora, siga uma das seguintes abordagens ou uma combinação delas.
  - Você pode adotar a mesma abordagem de uma instância de banco de dados autônoma do MySQL ou do PostgreSQL.

    As técnicas de autenticação de logins e permissões para instâncias de bancos de dados autônomas do MySQL ou do PostgreSQL, como usar comandos SQL ou modificar tabelas do esquema de banco de dados, também funcionam com o Aurora. Para obter mais informações, consulte Segurança com o Amazon Aurora MySQL ou Segurança com o Amazon Aurora PostgreSQL.
  - Você pode usar a autenticação de banco de dados do IAM.

    Com a autenticação de banco de dados do IAM, é possível autenticar o cluster de banco de dados do Aurora utilizando um usuário ou um perfil do IAM e um token de autenticação. Um token de autenticação é um valor exclusivo, gerado usando o processo de assinatura Signature Version 4. Ao usar a autenticação de banco de dados do IAM, você pode usar as mesmas credenciais para controlar o acesso aos seus recursos e bancos de dados da AWS. Para ter mais informações, consulte Autenticação do banco de dados do IAM.
  - É possível utilizar a autenticação Kerberos para o Aurora PostgreSQL e o Aurora MySQL.

    É possível usar o Kerberos para autenticar usuários quando eles se conectam ao seu cluster de banco de dados do Aurora PostgreSQL e Aurora MySQL. Nesse caso, o cluster de banco de dados funciona com o AWS Directory Service for Microsoft Active Directory para habilitar a autenticação Kerberos. O AWS Directory Service for Microsoft Active Directory também é chamado de AWS Managed Microsoft AD. Manter todas as suas credenciais no mesmo diretório

pode economizar tempo e esforço. Há um lugar centralizado para armazenar e gerenciar credenciais para vários clusters de banco de dados. O uso de um diretório também pode melhorar o perfil de segurança geral. Para ter mais informações, consulte Usar a autenticação Kerberos com o Aurora PostgreSQL e Usar a autenticação Kerberos para Aurora MySQL.

Para obter informações sobre a configuração de segurança, consulte Segurança no Amazon Aurora.

Uso do SSL com clusters de banco de dados Aurora

Os clusters de banco de dados do Amazon Aurora oferecem suporte a conexões Secure Sockets Layer (SSL) de aplicações que usam o mesmo processo e a mesma chave pública que as instâncias de banco de dados do Amazon RDS. Para obter mais informações, consulte Segurança com o Amazon Aurora MySQL, Segurança com o Amazon Aurora PostgreSQL ou Usar TLS/SSL com o Aurora Serverless v1.

Alta disponibilidade do Amazon Aurora

A arquitetura do Amazon Aurora envolve separação entre armazenamento e computação. O Aurora contém alguns recursos de alta disponibilidade que se aplicam aos dados no cluster de banco de dados. Os dados permanecem seguros, mesmo que algumas ou todas as instâncias de banco de dados no cluster fiquem indisponíveis. Outros recursos de alta disponibilidade se aplicam às instâncias de banco de dados. Esses recursos ajudam a garantir que uma ou mais instâncias de banco de dados estejam prontas para lidar com solicitações de banco de dados do aplicativo.

Tópicos

- Alta disponibilidade de dados do Aurora
- Alta disponibilidade de instâncias de banco de dados do Aurora
- Alta disponibilidade entre as regiões da AWS com bancos de dados globais do Aurora
- Tolerância a falhas para um cluster de banco de dados do Aurora
- Alta disponibilidade com o Amazon RDS Proxy

Alta disponibilidade de dados do Aurora

O Aurora armazena cópias dos dados em um cluster de banco de dados em várias zonas de disponibilidade em uma única Região da AWS. O Aurora armazena essas cópia independentemente

de as instâncias no cluster de banco de dados abrangerem várias zonas de disponibilidade. Para ter mais informações sobre o Aurora, consulte Como gerenciar um cluster de banco de dados do Amazon Aurora.

Quando os dados são gravados na instância principal de banco de dados, o Aurora replica os dados de forma síncrona nas zonas de disponibilidade para seis nós de armazenamento associados ao volume do cluster. Isso fornece redundância de dados, elimina congelamentos de E/S e minimiza picos de latência durante backups do sistema. Executar uma instância de banco de dados com alta disponibilidade pode aumentar a disponibilidade durante a manutenção planejada do sistema e ajudar a proteger os bancos de dados contra falhas e interrupções da zona de disponibilidade. Para ter mais informações sobre zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.

Alta disponibilidade de instâncias de banco de dados do Aurora

Depois de criar a instância primária (gravador), você poderá criar até 15 réplicas somente leitura do Aurora. As réplicas do Aurora também são conhecidas como instâncias de leitor. As réplicas do Aurora usam replicação assíncrona para permitir alta disponibilidade sem afetar o desempenho da instância primária.

Durante as operações diárias, é possível descarregar parte do trabalho para aplicativos com uso intensivo de leitura usando as instâncias de leitor para processar consultas `SELECT`. Quando um problema afeta a instância primária, uma dessas instâncias de leitor assume como instância primária. Esse mecanismo é conhecido como failover. Muitos recursos do Aurora se aplicam ao mecanismo de failover. Por exemplo, o Aurora detecta problemas de banco de dados e ativa o mecanismo de failover automaticamente, quando necessário. O Aurora também tem recursos que reduzem o tempo de conclusão do failover. Isso minimiza o tempo em que o banco de dados não está disponível para gravação durante um failover.

O Aurora foi projetado para se recuperar o mais rápido possível, e o caminho mais rápido para a recuperação geralmente é reiniciar ou fazer failover para a mesma instância de banco de dados. A reinicialização é mais rápida e envolve menos sobrecarga do que o failover.

Para usar uma string de conexão que permanece a mesma mesmo quando um failover promove uma nova instância primária, conecte-se ao endpoint do cluster. O endpoint do cluster sempre representa a instância primária atual no cluster. Para ter mais informações sobre o endpoint do cluster, consulte Conexões de endpoints do Amazon Aurora.

> **Tip**
>
> Em cada Região da AWS, as zonas de disponibilidade representam locais distintos entre si para fornecer isolamento em caso de interrupções. Recomendamos distribuir a instância primária e as instâncias de leitor no cluster de banco de dados em várias AZs para melhorar a disponibilidade do cluster de banco de dados. Dessa forma, um problema que afeta uma AZ inteira não causa uma interrupção para o cluster.
> É possível configurar um cluster de banco de dados multi-AZ fazendo uma escolha simples ao criar o cluster. Você pode usar o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS. Também é possível converter um cluster de banco de dados existente do Aurora em um cluster de banco de dados multi-AZ adicionando uma nova instância de banco de dados de leitor e especificando uma AZ diferente.

Alta disponibilidade entre as regiões da AWS com bancos de dados globais do Aurora

Para obter alta disponibilidade em várias Regiões da AWS, é possível configurar bancos de dados globais do Aurora. Cada banco de dados global do Aurora abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e recuperação de desastres decorrentes de interrupções em uma Região da AWS. O Aurora replica de forma assíncrona todos os dados e atualizações da Região da AWS primária para cada uma das regiões secundárias. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.

Tolerância a falhas para um cluster de banco de dados do Aurora

Um cluster de banco de dados do Aurora é tolerante a falhas por concepção. O volume do cluster abrange várias zonas de disponibilidade em uma única Região da AWS e cada zona de disponibilidade contém uma cópia dos dados de volume do cluster. Esta funcionalidade significa que seu cluster de banco de dados pode tolerar falhas de uma Zona de disponibilidade sem perder dados, apenas uma breve interrupção do serviço.

Se a instância primária em um cluster de banco de dados falhar, o Aurora fará failover automaticamente para uma nova instância primária de uma das duas seguintes maneiras:

- Ao promover uma réplica do Aurora existente para a nova instância primária
- Ao criar uma nova instância primária

Se o cluster de banco de dados tiver uma ou mais réplicas do Aurora, uma réplica do Aurora será promovida à instância primária durante um evento de falha. Um evento de falha resulta em uma breve interrupção, durante a qual as operações de leitura e gravação falham com uma exceção. No entanto, o serviço é restaurado normalmente em menos de 60 segundos e muitas vezes em menos de 30 segundos. Para aumentar a disponibilidade do seu cluster de banco de dados, recomendamos que você crie pelo menos uma ou mais réplicas do Aurora em duas ou mais Zonas de disponibilidade diferentes.

> **Tip**
>
> No Aurora MySQL, você pode melhorar a disponibilidade durante um failover com mais de uma instância de banco de dados de leitor em um cluster. No Aurora MySQL, o Aurora reinicia somente a instância de banco de dados do gravador e a instância do leitor que é destino de failover. Outras instâncias do leitor no cluster permanecem disponíveis durante um failover para continuar processando consultas por meio de conexões com o endpoint do leitor.
>
> Você também pode melhorar a disponibilidade durante um failover usando o proxy do RDS com o seu cluster de banco de dados do Aurora. Para ter mais informações, consulte Alta disponibilidade com o Amazon RDS Proxy.

Você pode personalizar a ordem em que suas réplicas do Aurora são promovidas à instância primária após uma falha, atribuindo uma prioridade a cada réplica. As prioridades variam de 0, para a prioridade mais alta, a 15, para a prioridade mais baixa. Se a instância primária falhar, o Amazon RDS promoverá a réplica do Aurora com a maior prioridade à nova instância primária. É possível modificar a prioridade de uma réplica do Aurora a qualquer momento. Modificar a prioridade não desencadeia um failover.

A mesma prioridade pode ser compartilhada por mais de uma réplica do Aurora, resultando em níveis de promoção. Se duas ou mais réplicas do Aurora compartilharem a mesma prioridade, o Amazon RDS promoverá a réplica que for maior. Se duas ou mais réplicas do Aurora compartilharem a mesma prioridade e o mesmo tamanho, o Amazon RDS promoverá uma réplica arbitrária no mesmo nível de promoção.

> **Note**
>
> Vários fatores estão envolvidos na identificação de um destino de failover. Após cinco tentativas de failover com falha, os níveis de promoção não são mais considerados.

Se o cluster de banco de dados não contiver quaisquer réplicas do Aurora, a instância primária será recriada na mesma AZ durante um evento de falha. Um evento de falha resulta em uma interrupção durante a qual as operações de leitura e gravação falham com uma exceção. O serviço é reestabelecido quando a nova instância primária é criada, o que normalmente leva menos de 10 minutos. Promover uma réplica do Aurora à instância primária é muito mais rápido do que criar uma nova instância primária.

Suponha que a instância principal no cluster não esteja disponível devido a uma interrupção que afeta toda uma AZ. Nesse caso, a maneira de colocar uma nova instância principal online depende de o cluster usar ou não uma configuração multi-AZ:

- Se o cluster provisionado ou do Aurora Serverless v2 contiver instâncias de leitor em outras AZs, o Aurora usará o mecanismo de failover para promover uma dessas instâncias de leitor para ser a nova instância principal.
- Se o cluster provisionado ou do Aurora Serverless v2 contiver apenas uma única instância de banco de dados ou se a instância principal e todas as instâncias de leitor estiverem na mesma AZ, você deverá criar manualmente uma ou mais instâncias de banco de dados em outra AZ.
- Se o cluster usar Aurora Serverless v1, Aurora criará automaticamente uma nova instância de banco de dados em outra AZ. No entanto, esse processo envolve uma substituição de host e, portanto, leva mais tempo do que um failover.

> **Note**
>
> O Amazon Aurora também oferece suporte a replicação com um banco de dados MySQL externo ou uma instância de banco de dados MySQL do RDS. Para ter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Alta disponibilidade com o Amazon RDS Proxy

Com o RDS Proxy, você pode criar aplicações que podem tolerar falhas de banco de dados de forma transparente sem precisar escrever códigos complexos de tratamento de falhas. O proxy direciona automaticamente o tráfego para uma nova instância de banco de dados, preservando as conexões da aplicação. Ele também ignora os caches do Sistema de Nomes de Domínio (DNS) para reduzir os tempos de failover em até 66% para bancos de dados multi-AZ do Aurora. Para ter mais informações, consulte Amazon RDS Proxypara Aurora.

Replicação com o Amazon Aurora

Existem várias opções de replicação no Aurora. Cada cluster de banco de dados do Aurora tem replicação interna entre várias instâncias de banco de dados no mesmo cluster. Você também pode configurar a replicação com o cluster Aurora como a origem ou o destino. Quando você replica dados para ou fora de um cluster de Aurora cluster, você pode escolher entre recursos internos, como bancos de dados Aurora globais ou os mecanismos tradicionais de replicação para os mecanismos de banco de dados MySQL ou PostgreSQL. Você pode escolher as opções apropriadas com base em qual fornece a combinação certa de alta disponibilidade, conveniência e performance para suas necessidades. As seções a seguir explicam como e quando escolher cada técnica.

Tópicos

- Réplicas do Aurora
- Replicação com o Aurora MySQL
- Replicação com o Aurora PostgreSQL

Réplicas do Aurora

Quando você cria uma segunda, terceira e outras instâncias de banco de dados e assim por diante em um cluster de banco de dados Aurora provisionado, Aurora configura automaticamente a replicação da instância de banco de dados do gravador para todas as outras instâncias de banco de dados. Essas outras instâncias de banco de dados são somente leitura e são conhecidas como réplicas de Aurora. Também nos referimos a elas como instâncias de leitor ao discutir as maneiras de combinar instâncias de banco de dados de escritor e leitor dentro de um cluster.

As réplicasAurora têm dois propósitos principais. Você pode emitir consultas para eles para dimensionar as operações de leitura da aplicação. Normalmente, você faz isso conectando-se ao

endpoint do leitor do cluster. Dessa forma, Aurora pode espalhar a carga para conexões somente leitura na quantidade de réplicas de Aurora que você tem no cluster. As réplicas de Aurora também ajudam a aumentar a disponibilidade. Se a instância do gravador em um cluster ficar indisponível, Aurora promove automaticamente uma das instâncias do leitor para ocupar seu lugar como o novo gravador.

Um cluster de banco de dados do Aurora pode conter até 15 Aurora réplicas. As réplicas do Aurora podem ser distribuídas entre as zonas de disponibilidade abrangidas por um cluster de banco de dados em uma região da AWS.

Os dados em seu cluster de banco de dados têm seus próprios recursos de alta disponibilidade e confiabilidade, independentemente das instâncias de banco de dados no cluster. Se você não estiver familiarizado com os recursos de armazenamento do Aurora, consulte Visão geral do armazenamento do Amazon Aurora. O volume do cluster de banco de dados é composto fisicamente por várias cópias dos dados do cluster de banco de dados. A instância primária e as réplicas de Aurora no cluster do banco de dados visualizam os dados no volume de cluster como um volume lógico único.

Como resultado, todas as réplicas do Aurora retornam os mesmos dados para os resultados da consulta com o mínimo de atraso da réplica. Esse atraso é geralmente muito inferior a 100 milissegundos, depois que a instância primária grava uma atualização. O atraso da réplica varia de acordo com a taxa de mudança no banco de dados. Ou seja, durante os períodos em que uma grande quantidade de operações de gravação ocorre para o banco de dados, você pode ver um aumento no atraso da réplica.

> **Note**
>
> A réplica do Aurora é reiniciada automaticamente quando perde a comunicação com a instância de banco de dados do gravador por mais de 60 segundos nas seguintes versões do Aurora PostgreSQL:
>
> - 14.6 e versões posteriores
> - 13.9 e versões posteriores
> - 12.13 e versões posteriores
> - Todas as versões do Aurora PostgreSQL 11
>
> Com o recurso de disponibilidade de leitura, as réplicas do Aurora não são reiniciadas automaticamente. Para obter mais informações sobre o recurso de disponibilidade de leitura

e as versões em que ele é aplicável, consulte Melhorar a disponibilidade de leitura das réplicas do Aurora.

As réplicas do Aurora funcionam bem para a escalabilidade de leitura porque são totalmente dedicadas a operações de leitura no seu volume de cluster. As operações de gravação são gerenciadas pela instância principal. Como o volume do cluster é compartilhado entre todas as instâncias de banco de dados no cluster de banco de dados, um trabalho adicional mínimo é necessário para replicar uma cópia dos dados para cada réplica do Aurora.

Para aumentar a disponibilidade, você pode usar as réplicas do Aurora como destinos de failover. Em outras palavras, se a instância primária falhar, uma réplica do Aurora será promovida à instância primária. Há uma breve interrupção durante a qual as solicitações de leitura e gravação feitas na instância principal falharão com uma exceção.

Promover uma réplica do Aurora por failover é muito mais rápido do que recriar a instância primária. Se o cluster de banco de dados Aurora não incluir todas as réplicas do Aurora, o cluster de banco de dados não estará disponível durante o período necessário para a instância de banco de dados se recuperar do evento de falha.

Quando acontece um failover, algumas das réplicas do Aurora podem ser reinicializadas, dependendo da versão do mecanismo de banco de dados. Por exemplo, no Aurora MySQL, o Aurora reinicia somente a instância de banco de dados do gravador e o destino de failover durante um failover. Para obter mais informações sobre o comportamento de reinicialização de diferentes versões do mecanismo de banco de dados Aurora, consulte Reinicializar um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora. Para obter informações sobre o que acontece com os caches de páginas durante a reinicialização ou o failover, consulte Cache de página perdurável.

Para cenários de alta disponibilidade, recomendamos criar uma ou mais réplicas do Aurora. Eles devem ser iguais à mesma classe da instância de banco de dados da instância primária em zonas de disponibilidade diferentes para o cluster de banco de dados do Aurora. Para obter mais informações sobre réplicas do Aurora como destinos de failover, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

Você não pode criar uma réplica do Aurora criptografada para um cluster de banco de dados do Aurora não criptografado. Você não pode criar uma réplica do Aurora não criptografada para um cluster de banco de dados do Aurora criptografado.

> **Tip**
>
> Você pode usar réplicas do Aurora em um cluster do Aurora como sua única forma de replicação para manter seus dados altamente disponíveis. Você também pode combinar a replicação interna de Aurora com os outros tipos de replicação. Desta forma, pode ajudar a fornecer um nível extra de alta disponibilidade e distribuição geográfica de seus dados.

Para obter detalhes sobre como criar uma réplica do Aurora, consulte Adicionar réplicas do Aurora a um cluster de banco de dados.

Replicação com o Aurora MySQL

Além das réplicas do Aurora, você tem as seguintes opções de replicação com o Aurora MySQL:

- Clusters de banco de dados do Aurora MySQL em diferentes regiões da AWS.
  - Você pode replicar dados em várias regiões usando um banco de dados Aurora global. Para obter mais detalhes, consulte Alta disponibilidade entre as regiões da AWS com bancos de dados globais do Aurora
  - É possível criar uma réplica de leitura do Aurora de um cluster de banco de dados Aurora MySQL em uma região da AWS diferente com a replicação de log binário (binlog) do MySQL. Cada cluster pode ter até cinco réplicas de leitura criadas dessa maneira, cada uma em uma região diferente.
- Dois clusters de banco de dados do Aurora MySQL na mesma região da usando a replicação do log binário do MySQL (binlog).
- Uma instância de banco de dados RDS para MySQL como a fonte de um cluster de banco de dados Aurora MySQL, criando uma réplica de leitura do Aurora de uma instância de banco de dados RDS para MySQL. Normalmente, essa abordagem é usada para migrar para o Aurora MySQL em vez de uma replicação contínua.

Para obter mais informações sobre a replicação com o Aurora MySQL, consulte Replicação com o Amazon Aurora MySQL.

Replicação com o Aurora PostgreSQL

Além das réplicas do Aurora, você tem as seguintes opções de replicação com o Aurora PostgreSQL:

- Um cluster de banco de dados primário do Aurora em uma região e até dez clusters de banco de dados secundários apenas para leitura em diferentes regiões usando um Aurora Global Database. O Aurora PostgreSQL não é compatível com réplicas do Aurora entre regiões. No entanto, você pode usar o banco de dados global Aurora para escalar os recursos de leitura do cluster de banco de dados Aurora PostgreSQL para mais de uma região da AWS e para atingir as metas de disponibilidade. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.
- Dois clusters de banco de dados de Aurora PostgreSQL na mesma região, usando o recurso de replicação lógica do PostgreSQL.
- Uma instância de banco de dados RDS para PostgreSQL como fonte de dados e um cluster de banco de dados Aurora PostgreSQL, criando uma réplica de leitura do Aurora de uma instância de banco de dados RDS para PostgreSQL. Normalmente, essa abordagem é usada para migrar para o Aurora MySQL em vez de uma replicação contínua.

Para obter mais informações sobre a replicação com o Aurora PostgreSQL, consulte Replicação com Amazon Aurora PostgreSQL.

Faturamento da instância de banco de dados para Aurora

As instâncias provisionadas do Amazon RDS em um cluster do Amazon Aurora são faturadas com base nos seguintes componentes:

- Horas de instância de banco de dados (por hora) – com base na classe da instância de banco de dados (por exemplo, db.t2.small ou db.m4.large). A definição de preço está listada em uma base por hora, mas é calculada em segundos e mostra o tempo no formato decimal. O uso do RDS é cobrado em incrementos de um segundo, com um mínimo de 10 minutos. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.
- Armazenamento (por GiB por mês) – a capacidade de armazenamento provisionado para a sua instância de banco de dados. Se você dimensionar sua capacidade de armazenamento provisionada dentro do mês, sua fatura será rateada. Para obter mais informações, consulte Armazenamento do Amazon Aurora.
- Solicitações de entrada/saída (E/S) (por 1 milhão de solicitações): o número total de solicitações de E/S de armazenamento que você fez em um ciclo de faturamento, para somente a configuração de cluster de banco de dados do Aurora Standard.

  Para obter mais informações sobre o faturamento de E/S do Amazon Aurora, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.

- Armazenamento de backup (por GiB por mês) – o armazenamento de backup é o armazenamento associado a backups automatizados de banco de dados e a qualquer DB snapshot ativo que você tenha feito. Aumentar seu período de retenção de backup ou fazer snapshots de bancos de dados adicionais aumenta o armazenamento de backup utilizado por seu banco de dados. A cobrança por segundo não se aplica ao armazenamento de backup (medido em GB por mês).

  Para obter mais informações, consulte Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora.
- Transferência de dados (por GB): transferência de dados para dentro e para fora de sua instância de banco de dados de ou para a Internet e outras regiões da AWS. Para ter exemplos úteis, consulte a postagem do blog da AWS Exploring Data Transfer Costs for AWS Managed Databases.

O Amazon RDS fornece as seguintes opções de compra para permitir otimizar os custos com base em suas necessidades:

- On-Demand Instances (Instâncias sob demanda): pague por hora pelas horas de instância de banco de dados que você usar. A definição de preço está listada em uma base por hora, mas é calculada em segundos e mostra o tempo no formato decimal. O uso do RDS agora é cobrado em incrementos de um segundo, com um mínimo de 10 minutos.
- Reserved Instances (Instâncias reservadas): reserve uma instância de banco de dados por um período de vigência de um ou três anos e receba um desconto significativo com relação à definição de preço das instâncias de banco de dados sob demanda. Com o uso de instâncias reservadas, você pode iniciar, excluir, iniciar ou parar várias instâncias dentro de uma hora e obter o benefício da instância reservada para todas as instâncias.
- Aurora Serverless v2: o Aurora Serverless v2 fornece capacidade sob demanda em que a unidade de faturamento é a unidade de capacidade do Aurora (ACU) em horas, em vez de horas de instância de banco de dados. A capacidade do Aurora Serverless v2 aumenta e diminui, em um intervalo que você especificar, dependendo da carga em seu banco de dados. Você pode configurar um cluster onde toda a capacidade é o Aurora Serverless v2. Você pode configurar uma combinação do Aurora Serverless v2 e de instâncias provisionadas sob demanda ou reservadas. Para obter mais informações sobre como funcionam as ACUs do Aurora Serverless v2, consulte Como funciona o Aurora Serverless v2.
- Aurora PostgreSQL Limitless Database: Aurora PostgreSQL Limitless Database é um recurso de escalabilidade horizontal automatizado que escala além do throughput de gravação e dos limites de armazenamento de uma única instância de banco de dados. O Limitless Database distribui a

workload em várias instâncias de gravação do Aurora, mantendo a facilidade de operar como um único banco de dados. O Limitless Database fornece capacidade sob demanda, onde a unidade de cobrança é horas de unidade de capacidade do Aurora (ACU) em um grupo de fragmentos de banco de dados. Para obter informações sobre como funcionam as ACUs do Limitless Database, consulte Criação de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database.

Para obter informações sobre a definição de preço do Aurora, consulte a página de definição de preço do Aurora.

Tópicos

- Instâncias de banco de dados sob demanda do Aurora
- Instâncias de banco de dados reservadas para o Amazon Aurora

Instâncias de banco de dados sob demanda do Aurora

As Instâncias de banco de dados sob demanda do Amazon RDS são cobradas com base na classe da instância de banco de dados (por exemplo, db.t3.small ou db.m5.large). Para obter informações sobre a definição de preço do Amazon RDS, consulte a página de produto do Amazon RDS.

O faturamento é iniciado para uma instância de banco de dados assim que a instância de banco de dados se torna disponível. A definição de preço está listada em uma base por hora, mas é calculada em segundos e mostra o tempo no formato decimal. O uso do Amazon RDS é faturado em incrementos de um segundo, com um mínimo de dez minutos. No caso de uma alteração de configuração faturável, como a expansão da capacidade de computação ou armazenamento, você paga um mínimo de 10 minutos. O faturamento continua até que a instância de banco de dados é encerrada, o que ocorre quando você exclui a instância de banco de dados ou se ocorre uma falha na instância de banco de dados.

Se não desejar ser faturado por sua instância de banco de dados, você deverá encerrá-la ou excluí-la para evitar ser cobrado por horas adicionais da instância de banco de dados. Para obter mais informações sobre os estados de instâncias de banco de dados dos quais você é faturado, consulte Visualizar o status de uma instância de banco de dados em um cluster do Aurora.

Instâncias de banco de dados interrompidas

Quando sua instância de banco de dados for interrompida, você será cobrado pelo armazenamento provisionado, incluindo IOPS provisionadas. Você também é faturado pelo armazenamento de backup, incluindo o armazenamento de snapshots manuais e backups automatizados durante a janela de retenção especificada. Não há cobrança por horas de instância de banco de dados.

Instâncias de banco de dados multi-AZ

Uma configuração multi-AZ aprimora a durabilidade e disponibilidade dos dados ao provisionar e manter automaticamente uma réplica em espera síncrona em outra zona de disponibilidade. Devido aos recursos adicionais e ao aumento da disponibilidade, as implantações multi-AZ têm um preço mais alto do que as implantações single-AZ e podem custar aproximadamente o dobro devido à instância em espera adicional e aos recursos associados.

Considere os seguintes detalhes importantes sobre o preço de multi-AZ:

- Custos de computação: cobrados por hora de instância de banco de dados para as instâncias primária e em espera.

- Custos de armazenamento: cobrados por GB-mês pelo armazenamento provisionado para as instâncias primária e em espera.
- Custos de transferência de dados: a replicação entre as instâncias primária e em espera está incluída no custo, mas outras taxas de transferência de dados podem ser aplicadas com base no uso.

Para estimar com precisão seus custos mensais com base em seu caso de uso específico e na Região da AWS, você pode usar a AWS Calculadora de Preços. Nessa ferramenta, você insere os detalhes da sua configuração e recebe um detalhamento abrangente dos custos.

> **Note**
>
> Os preços estão sujeitos a alterações. Consulte as informações mais atualizadas na página Preço do Amazon RDS.

Instâncias de banco de dados reservadas para o Amazon Aurora

Usando instâncias de banco de dados reservadas, você pode reservar uma instância de banco de dados por um período de um ou três anos. As instâncias de banco de dados reservadas fornecem um desconto significativo em comparação com os preços de instâncias de banco de dados sob demanda. As instâncias de banco de dados reservadas não são instâncias físicas, mas um desconto na fatura aplicado na sua conta pelo uso de determinadas instâncias de banco de dados sob demanda. Os descontos para instâncias de bancos de dados reservadas são vinculados ao tipo de instância e à Região da AWS.

O processo geral de trabalho com instâncias de banco de dados reservadas é: primeiro, obtenha informações sobre ofertas de instâncias de banco de dados reservadas disponíveis; em seguida, compre uma oferta de instância de banco de dados reservada; e, por fim, obtenha informações sobre suas instâncias de banco de dados reservadas existentes.

Para ter informações sobre a compra de instâncias de banco de dados reservadas e a visualização de cobranças de instâncias de banco de dados reservadas, consulte as seções a seguir.

- Comprar instâncias de banco de dados reservadas para o Amazon Aurora
- Visualizar a cobrança de instâncias de banco de dados reservadas para o Amazon Aurora

Visão geral de instâncias de banco de dados reservadas

Ao comprar uma instância de banco de dados reservada no Amazon RDS, você adquire um compromisso de obter uma taxa com desconto sobre um tipo específico de instância de banco de dados pela duração dela. Para usar uma instância de banco de dados reservada do Amazon RDS, crie uma instância de banco de dados como faria para uma instância sob demanda.

A nova instância de banco de dados que você criar deve ter as mesmas especificações da instância de banco de dados reservada quanto ao seguinte:

- Região da AWS
- Mecanismo de banco de dados (não é preciso haver correspondência com o número da versão do mecanismo de banco de dados).
- Tipo de instância do banco de dados

Se as especificações da nova instância de banco de dados corresponderem às de uma instância de banco de dados reservada existente para a sua conta, será cobrada a taxa de desconto oferecida

para a instância reservada. Caso contrário, uma taxa sob demanda será cobrada para a instância de banco de dados.

Você pode modificar uma instância de banco de dados que está usando como instância de banco de dados reservada. Se a modificação estiver dentro das especificações da instância de banco de dados reservada, parte ou todo o desconto ainda se aplicará à instância de banco de dados modificada. Se a modificação estiver fora das especificações, como alterar a classe de instância, o desconto não será mais aplicado. Para ter mais informações, consulte Instâncias de banco de dados reservadas de tamanho flexível.

Tópicos

- Tipos de oferta
- Flexibilidade de configuração do cluster de banco de dados do Aurora
- Instâncias de banco de dados reservadas de tamanho flexível
- Exemplos de faturamento de instância de banco de dados reservada do Aurora
- Excluir uma instância de banco de dados reservada

Para ter mais informações sobre instâncias de banco de dados reservadas, inclusive definição de preço, consulte Instâncias reservadas do Amazon RDS.

Tipos de oferta

Instâncias de banco de dados reservadas estão disponíveis em três variedades: sem pagamento adiantado, com pagamento adiantado parcial e com pagamento adiantado integral. Esses tipos permitem otimizar os custos do Amazon RDS com base no uso esperado.

> **Note**
>
> Nem todas as classes de instância do RDS são compatíveis com todos os tipos de oferta de instância reservada. Por exemplo, algumas classes de instância podem não oferecer a opção Sem pagamento adiantado. Para confirmar a disponibilidade, analise as ofertas de instância reservada no Console de gerenciamento da AWS ou use o comando `describe-reserved-db-instances-offerings` da AWS CLI.

Sem taxas iniciais

Essa opção fornece acesso a uma instância de banco de dados reservado sem a necessidade de pagamento adiantado. Sua instância de banco de dados reservada sem pagamento adiantado será cobrada de acordo com uma taxa horária com desconto por cada hora dentro do período de vigência, independentemente do uso, e nenhum pagamento adiantado é obrigatório. Essa opção só está disponível como uma reserva de um ano.

Adiantado parcial

Essa opção requer que uma parte da instância de banco de dados reservada seja paga antecipadamente. As horas restantes do período de vigência serão cobradas com base em uma taxa horária com desconto, independentemente do uso. Essa opção é a substituição da opção de Utilização pesada anterior.

Adiantado integral

O pagamento integral é feito no início do período de vigência, sem outros custos ou cobranças por hora incorridos pelo restante do período, independentemente do número de horas usadas.

Se você estiver usando faturamento consolidado, todas as contas da organização serão tratadas como se fossem uma só. Isso significa que todas as contas na organização podem receber o custo-benefício por hora das instâncias de banco de dados reservadas que são compradas por qualquer outra conta. Para ter mais informações sobre o faturamento consolidado, consulte Instâncias de bancos de dados reservadas do Amazon RDS no Guia do usuário do Gerenciamento de Faturamento e Custos da AWS.

Flexibilidade de configuração do cluster de banco de dados do Aurora

É possível usar instâncias de banco de dados reservadas do Aurora com as duas configurações do cluster de banco de dados:

- Aurora I/O-Optimized: você paga somente pelo uso e armazenamento dos clusters de banco de dados, sem custos adicionais pelas operações de E/S de leitura e gravação.
- Aurora Standard: além do uso e do armazenamento dos clusters de banco de dados, você também paga uma taxa padrão por um milhão de solicitações para operações de E/S.

O Aurora contabiliza automaticamente a diferença de preço entre essas configurações. O Aurora I/O-Optimized consome 30% mais unidades normalizadas por hora do que o Aurora Standard.

Para obter mais informações sobre as configurações de armazenamento do cluster de bancos de dados do Aurora, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora. Para obter mais informações sobre o preço referente às configurações de armazenamento de cluster do Aurora, consulte Preços do Amazon Aurora.

Instâncias de banco de dados reservadas de tamanho flexível

Ao adquirir uma instância de banco de dados reservada, uma das especificações feitas é a classe da instância, por exemplo, db.r5.large. Para ter mais informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Se você tiver uma instância de banco de dados e precisar escalá-la para uma capacidade maior, sua instância de banco de dados reservada será automaticamente aplicada à sua instância de banco de dados escalada. Ou seja, suas instâncias de banco de dados reservadas são aplicadas automaticamente em todos os tamanhos de classe de instância de banco de dados. As instâncias de banco de dados reservadas de tamanho flexível estão disponíveis para instâncias de bancos de dados com a mesma Região da AWS e mecanismo de banco de dados. As instâncias de banco de dados reservadas de tamanho flexível só reduzem o tipo de classe de instância. Por exemplo, uma instância de banco de dados reservada de db.r6i.large pode se aplicar a uma db.r6i.xlarge, mas não a uma db.r6id.large ou db.r7g.large, porque db.r6id.large e db.r7g.large são tipos de classe de instância diferentes.

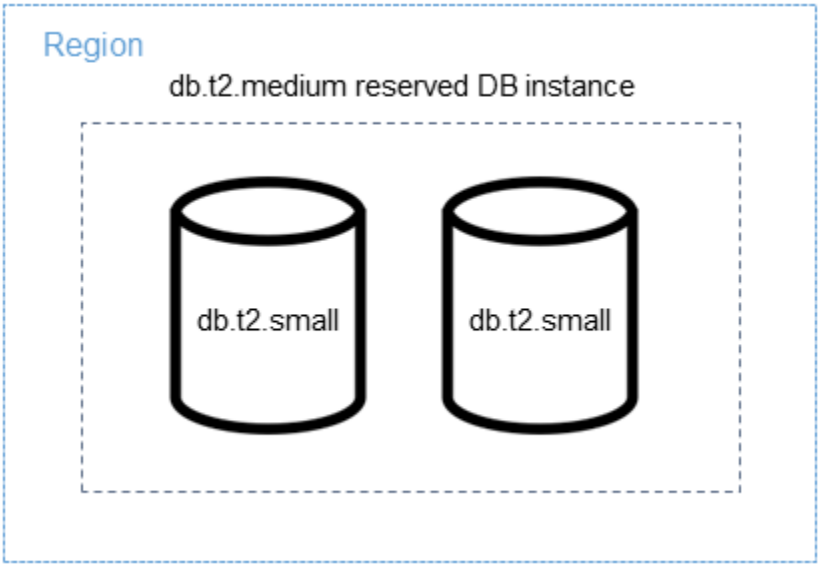
As instâncias de banco de dados reservadas de tamanho flexível estão disponíveis para os seguintes mecanismos de banco de dados Aurora:

- Aurora MySQL
- Aurora PostgreSQL

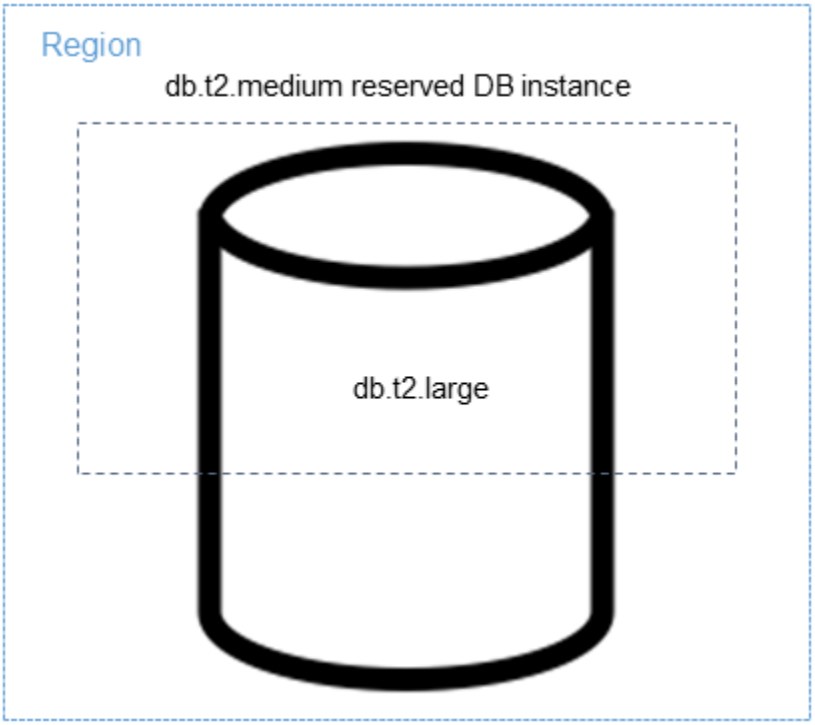
Você pode comparar o uso de tipos diferentes de instância de banco de dados reservada usando unidades normalizadas por hora. Por exemplo, uma unidade de uso em duas instâncias de banco de dados db.r3.large é equivalente a oito unidades normalizadas por hora de uso em uma db.r3.small. A tabela a seguir mostra o número de unidades normalizadas por hora para cada tamanho de instância de banco de dados.

Tamanho da instância	Unidades normalizadas por hora para uma instância de banco de dados, Aurora Standard	Unidades normalizadas por hora para uma instância de banco de dados, Aurora I/O-Optimized	Unidades normalizadas por hora para três instâncias de banco de dados (gravador e dois leitores), Aurora Standard	Unidades normalizadas por hora para três instâncias de banco de dados (gravador e dois leitores), Aurora I/O-Optimized
pequeno	1	1,3	3	3,9
médio	2	2,6	6	7,8
grande	4	5,2	12	15,6
xlarge	8	10,4	24	31,2
2xlarge	16	20,8	48	62,4
4xlarge	32	41,6	96	124,8
8xlarge	64	83,2	192	249,6
12xlarge	96	124,8	288	374,4
16xlarge	128	166,4	384	499,2
24xlarge	192	249,6	576	748,8
32xlarge	256	332,8	768	998,4

Por exemplo, suponhamos que você compre uma instância de bancos de dados `db.t2.medium` reservada e tenha duas instâncias de banco de dados `db.t2.small` em execução na conta na mesma Região da AWS. Nesse caso, o benefício de faturamento é aplicado integralmente a ambas as instâncias.



Ou, se você tiver uma instância `db.t2.large` em execução em sua conta na mesma Região da AWS, o benefício de faturamento será aplicado a 50% do uso da instância de banco de dados.



> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter

mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.

Exemplos de faturamento de instância de banco de dados reservada do Aurora

Os exemplos a seguir ilustram os preços das instâncias de banco de dados reservadas para clusters de banco de dados do Aurora usando as configurações de cluster de banco de dados do Aurora Standard e Aurora I/O-Optimized.

Exemplo usando Aurora Standard

O preço de uma instância de banco de dados reservada não fornece um desconto para os custos associados a armazenamento, backups e E/S. O exemplo a seguir ilustra o custo mensal total de uma instância de banco de dados reservada:

- Uma classe de instância de banco de dados single-AZ db.r5.large reservada do MySQL Aurora em Leste dos EUA (N. da Virgínia) por um custo de USD 0,19 por hora ou USD 138,70 por mês
- Armazenamento do Aurora a um custo de US$ 0,10 por GiB por mês (suponhamos US$ 45,60 por mês neste exemplo)
- E/S do Aurora a um custo de US$ 0,20 por 1 milhão de solicitações (suponhamos US$ 20 por mês neste exemplo)
- Armazenamento de backup do Aurora a US$ 0,021 por GiB por mês (suponhamos US$ 30 por mês neste exemplo)

Adicione todas essas opções (US$ 138,70 + US$ 45,60 + US$ 20 + US$ 30) com a instância de banco de dados reservada e o custo total por mês US$ 234,30.

Se você optar por usar uma instância de banco de dados sob demanda em vez de uma instância de banco de dados reservada, uma classe de instância de banco de dados single-AZ db.r5.large do Aurora MySQL no Leste dos EUA (N. da Virgínia) custa USD 0,29 por hora, ou USD 217,50 por mês. Dessa maneira, para uma instância de banco de dados sob demanda, adicione todas essas opções (US$ 217,50 + US$ 45,60 + US$ 20 + US$ 30), e o custo total por mês é US$ 313,10. Você economiza quase USD 79 por mês utilizando a instância de banco de dados reservada.

Exemplo de uso de um cluster de banco de dados do Aurora Standard com duas instâncias de leitor

Para usar instâncias reservadas para clusters de banco de dados do Aurora, basta comprar uma instância reservada para cada instância de banco de dados no cluster.

Estendendo o primeiro exemplo, você tem um cluster de banco de dados do Aurora MySQL com uma instância de banco de dados de gravador e duas réplicas do Aurora, totalizando três instâncias de banco de dados no cluster. As duas réplicas do Aurora não incorrem em cobranças extras de armazenamento ou backup. Se você comprar três instâncias de banco de dados db.r5.large reservadas do Aurora MySQL, seu custo será de USD 234,30 (para a instância de banco de dados do gravador) + 2 * (USD 138,70 + USD 20 E/D por réplica do Aurora), por um total de USD 551,70 por mês.

O custo sob demanda correspondente para um cluster de banco de dados do Aurora MySQL com uma instância de banco de dados de gravador e duas réplicas do Aurora é de USD 313,10 + 2 * (USD 217,50 + USD 20 de E/S por instância), totalizando USD 788,10 por mês. Você economiza quase USD 236,40 por mês utilizando as instâncias de banco de dados reservadas.

Exemplo usando Aurora I/O-Optimized

É possível reutilizar as instâncias de banco de dados reservadas existentes do Aurora Standard com o Aurora I/O-Optimized. Para aproveitar ao máximo os benefícios de seus descontos em instâncias reservadas com Aurora I/O-Optimized, você pode comprar 30% de instâncias reservadas adicionais semelhantes às suas instâncias reservadas atuais.

A tabela a seguir mostra exemplos de como estimar as instâncias reservadas adicionais ao usar Aurora I/O-Optimized. Se as instâncias reservadas necessárias forem uma fração, você poderá aproveitar a flexibilidade de tamanho disponível com as instâncias reservadas para chegar a um número inteiro. Nesses exemplos, “atual” se refere às instâncias reservadas Aurora Standard que você tem agora. Instâncias reservadas adicionais são o número de instâncias reservadas Aurora Standard que você deve comprar para manter seus descontos atuais em instâncias reservadas durante o uso do Aurora I/O-Optimized.

DB instance class	Instâncias reservadas atuais Aurora Standard	Instâncias reservadas necessárias para Aurora I/O-Optimized	Instâncias reservadas adicionais necessárias	Instâncias reservadas adicionais necessárias, usando flexibilidade de tamanho
db.r6g.large	10	10 * 1,3 = 13	3 * db.r6g.large	3 * db.r6g.large

DB instance class	Instâncias reservadas atuais Aurora Standard	Instâncias reservadas necessárias para Aurora I/O-Optimized	Instâncias reservadas adicionais necessárias	Instâncias reservadas adicionais necessárias, usando flexibilidade de tamanho
db.r6g.4xlarge	20	20 * 1,3 = 26	6 * db.r6g.4xlarge	6 * db.r6g.4xlarge
db.r6g.12xlarge	5	5 * 1,3 = 6,5	1,5 * db.r6g.12xlarge	Um de cada de db.r6g.12xlarge, r6g.4xlarge e r6g.2xlarge (0,5 * db.r6g.12xlarge = 1 * db.r6g.4xlarge + 1 * db.r6g.2xlarge)
db.r6i.24xlarge	15	15 * 1,3 = 19,5	4,5 * db.r6i.24xlarge	4 * db.r6i.24xlarge + 1 * db.r6i.12xlarge (0,5 * db.r6i.24xlarge = 1 * db.r6i.12xlarge)

Exemplo de uso de um cluster de banco de dados do Aurora I/O-Optimized com duas instâncias de leitor

Você tem um cluster de banco de dados do Aurora MySQL com uma instância de banco de dados de gravador e duas réplicas do Aurora, totalizando três instâncias de banco de dados no cluster. Eles usam a configuração do cluster de banco de dados do Aurora I/O-Optimized. Para usar instâncias de banco de dados reservadas para esse cluster, você precisaria comprar quatro instâncias de

banco de dados reservadas da mesma classe de instância de banco de dados. Três instâncias de banco de dados usando o Aurora I/O-Optimized consomem 3,9 unidades normalizadas por hora, em comparação com 3 unidades normalizadas por hora para três instâncias de banco de dados usando o Aurora Standard. No entanto, você economiza os custos mensais de E/S para cada instância de banco de dados.

> **Note**
>
> Os preços nestes exemplos são preços de exemplo e talvez não correspondam aos preços reais. Para obter informações sobre a definição de preço do Aurora, consulte Definição de preço do Amazon Aurora.

Excluir uma instância de banco de dados reservada

Os períodos de vigência de uma instância de banco de dados reservada envolvem um compromisso de um ou três anos. Não é possível cancelar uma instância de banco de dados reservada. No entanto, você pode excluir uma instância de banco de dados coberta por um desconto de instância de banco de dados reservada. O processo de exclusão de uma instância de banco de dados coberta por um desconto de instância de banco de dados reservada é o mesmo que o de qualquer outra instância de banco de dados.

Você receberá uma cobrança pelos custos adiantados, independentemente do uso dos recursos.

Se você excluir uma instância de banco de dados coberta por um desconto de instância de banco de dados reservada, poderá iniciar outra instância de banco de dados com especificações compatíveis. Neste caso, você continua recebendo a taxa com desconto durante o período de vigência da reserva (um ou três anos).

Comprar instâncias de banco de dados reservadas para o Amazon Aurora

Você pode usar o Console de gerenciamento da AWS, a AWS CLI e a API do RDS para trabalhar com instâncias de banco de dados reservadas.

Console

Você pode usar o Console de gerenciamento da AWS para trabalhar com instâncias de banco de dados reservadas conforme exibido nos procedimentos a seguir.

Para obter informações sobre preços e ofertas de instâncias de bancos de dados reservadas disponíveis

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Reserved instances (Instâncias reservadas).
3. Escolha Purchase Reserved DB Instance (Comprar instância de banco de dados reservada).
4. Em Product description (Descrição do produto), escolha o tipo de licenciamento e mecanismo de banco de dados.
5. Para DB instance class (Classe de instância do banco de dados), escolha a classe da instância de banco de dados.
6. Em Opção de implantação, selecione se deseja uma implantação de instância de banco de dados single-AZ ou multi-AZ.

> **Note**
>
> As instâncias reservadas do Amazon Aurora sempre têm a opção de implantação definida como Instância de banco de dados single-AZ. No entanto, quando você cria um cluster de banco de dados do Aurora, a opção de implantação padrão é Criar uma réplica ou um leitor do Aurora em uma AZ diferente (multi-AZ).
> Você deve comprar uma instância de banco de dados reservada para cada instância que planeja usar, inclusive réplicas do Aurora. Portanto, para implantações multi-AZ no Aurora, você deve comprar instâncias de banco de dados reservadas complementares.

7. Em Prazo, selecione por quanto tempo a instância de banco de dados deve ser reservada.
8. Em Offering type (Tipo de oferta), escolha o tipo de oferta.

   Após selecionar o tipo de oferta, você pode visualizar as informações de preço.

> **Important**
>
> Escolha Cancel (Cancelar) para evitar a compra da instância de banco de dados reservada e gerar quaisquer cobranças.

Assim que tiver informações sobre as ofertas de instâncias de banco de dados reservadas disponíveis, você poderá usá-las para comprar uma oferta, conforme mostrado no procedimento a seguir.

Para comprar uma instância de banco de dados reservada

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Reserved instances (Instâncias reservadas).
3.
> **Important**
>
> Antes de continuar, verifique se você está na Região da AWS correta. As instâncias de banco de dados reservadas são específicas à região e não podem ser transferidas entre regiões. Verifique o seletor de região no canto superior direito do console para garantir que você está comprando a instância reservada na região desejada.

4. Selecione Purchase Reserved DB Instance (Comprar instância de banco de dados reservada).
5. Em Product description (Descrição do produto), escolha o tipo de licenciamento e mecanismo de banco de dados.
6. Para DB instance class (Classe de instância do banco de dados), escolha a classe da instância de banco de dados.
7. Em Implantação multi-AZ, selecione se deseja uma implantação de instância de banco de dados single-AZ ou multi-AZ.

> **Note**
>
> As instâncias reservadas do Amazon Aurora sempre têm a opção de implantação definida como Instância de banco de dados Single-AZ. Quando você cria um cluster de banco de dados do Amazon Aurora por meio de sua instância de banco de dados reservada, ele é criado automaticamente como multi-AZ. Você deve comprar uma instância de banco de dados reservada para cada instância do banco de dados que planeja usar, inclusive réplicas do Aurora.

8. Em Term (Prazo), escolha quanto tempo você deseja que a instância de banco de dados seja reservada.
9. Em Offering type (Tipo de oferta), escolha o tipo de oferta.

Após selecionar o tipo de oferta, você pode visualizar as informações de definição de preço. Console Purchase reserved DB instance (Comprar instância de banco de dados reservada)

10. (Opcional) Você pode atribuir seu próprio identificador às instâncias de banco de dados reservadas adquiridas, a fim de ajudar a manter o controle delas. Em Reserved Id (ID da instância reservada), digite um identificador para a instância de banco de dados reservada.
11. Selecione Enviar.

    Sua instância de banco de dados reservada é comprada e, depois, exibida na lista Reserved instances (Instâncias reservadas).

Depois de adquirir instâncias de banco de dados reservadas, você poderá obter informações sobre elas, conforme mostrado no procedimento a seguir.

Para obter informações sobre instâncias de Bancos de Dados reservadas para a sua conta da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel Navigation (Navegação), escolha Reserved instances (Instâncias reservadas).

   As instâncias de banco de dados reservadas para sua conta são exibidas. Para ver informações detalhadas sobre uma instância de banco de dados reservada específica, escolha essa instância na lista. Você pode acabar vendo informações detalhadas sobre essa instância no painel de detalhes na parte inferior do console.

AWS CLI

Você pode usar a AWS CLI para trabalhar com instâncias de banco de dados reservadas, conforme mostrado nos exemplos a seguir.

Example obtenção de ofertas de instâncias de banco de dados reservadas disponíveis

Para obter informações sobre as ofertas disponíveis de instâncias de banco de dados reservadas, chame o comando AWS CLI da `describe-reserved-db-instances-offerings`.

```
aws rds describe-reserved-db-instances-offerings
```

Essa chamada retorna uma saída semelhante à seguinte:

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

Assim que tiver informações sobre as ofertas de instâncias de banco de dados reservadas disponíveis, você poderá usá-las para comprar uma oferta, conforme mostrado no exemplo a seguir.

Para comprar uma instância de banco de dados reservada, use o comando AWS CLI da `purchase-reserved-db-instances-offering` com os seguintes parâmetros:

- `--reserved-db-instances-offering-id` – o ID da oferta que você deseja comprar. Consulte o exemplo anterior para obter o ID da oferta.
- `--reserved-db-instance-id` – você pode atribuir seu próprio identificador às instâncias de banco de dados reservadas adquiridas, a fim de ajudar a manter o controle delas.

Example compra de uma instância de banco de dados reservada

O exemplo a seguir compra a oferta de instância de banco de dados reservada com o ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f* e atribui o identificador *MyReservation*.

Para Linux, macOS ou Unix:

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Para Windows:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

Esse comando retorna uma saída semelhante à seguinte:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

Depois de adquirir instâncias de banco de dados reservadas, você poderá obter informações sobre elas, conforme mostrado no exemplo a seguir.

Para obter informações sobre instâncias de Bancos de Dados reservadas para sua conta da AWS, chame o comando da AWS CLI `describe-reserved-db-instances`, como mostrado no exemplo a seguir.

Example obtenção de suas instâncias de banco de dados reservadas

```
aws rds describe-reserved-db-instances
```

Esse comando retorna uma saída semelhante à seguinte:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

API do RDS

Você pode usar a API do RDS para trabalhar com instâncias de banco de dados reservadas:

- Para obter informações sobre as ofertas de instâncias de bancos de dados reservadas disponíveis, chame a operação da API do Amazon RDS `DescribeReservedDBInstancesOfferings`.
- Assim que tiver informações sobre as ofertas de instâncias de banco de dados reservadas disponíveis, você poderá usá-las para comprar uma oferta, conforme mostrado no exemplo a seguir. Chame a operação da API do RDS `PurchaseReservedDBInstancesOffering` com os seguintes parâmetros:
  - `--reserved-db-instances-offering-id` – o ID da oferta que você deseja comprar.

- `--reserved-db-instance-id` – você pode atribuir seu próprio identificador às instâncias de banco de dados reservadas adquiridas, a fim de ajudar a manter o controle delas.
- Depois de adquirir instâncias de banco de dados reservadas, você poderá obter informações sobre elas, conforme mostrado no exemplo a seguir. Chame a operação da API do RDS `DescribeReservedDBInstances`.

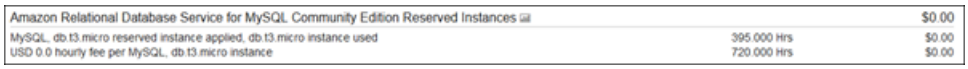
Visualizar a cobrança de instâncias de banco de dados reservadas para o Amazon Aurora

É possível visualizar o faturamento das suas instâncias de banco de dados reservadas no Painel de cobrança do Console de gerenciamento da AWS.

Para visualizar a cobrança de instâncias de banco de dados reservadas

1. Faça login no Console de gerenciamento da AWS.
2. No menu da conta no canto superior direito, escolha Billing Dashboard (Painel de faturamento).
3. Escolha Bill Details (Detalhes da fatura) no canto superior direito do painel.
4. Em AWS Service Charges (Cobranças de serviços da), expanda Relational Database Service (Serviço de banco de dados relacional).
5. Expanda a Região da AWS na qual as suas instâncias de banco de dados reservadas se encontram, por exemplo Oeste dos EUA (Oregon).

   Suas instâncias de banco de dados reservadas e suas cobranças por hora no mês atual são mostradas em Instâncias reservadas do Amazon Relational Database Service for ***Mecanismo de banco de dados***.

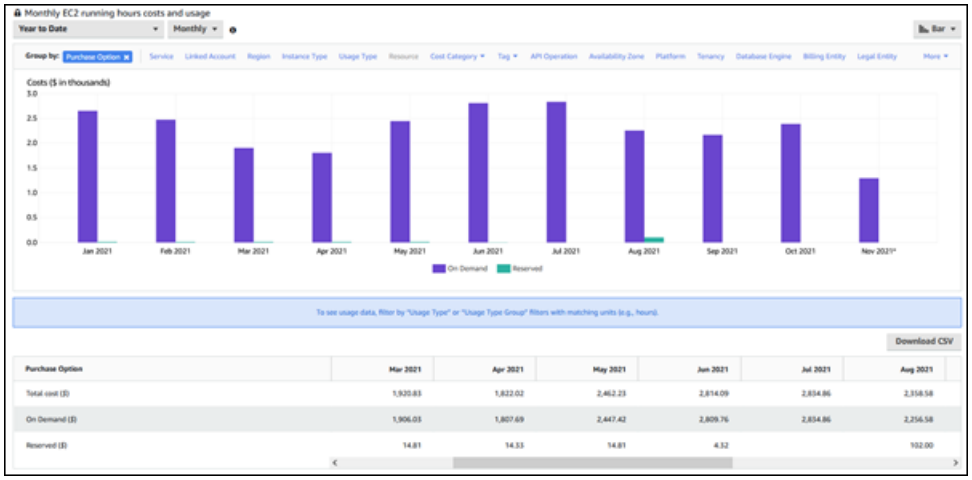
   

   A instância de banco de dados reservada neste exemplo foi comprada com Adiantamento integral e, portanto, não há cobranças por hora.
6. Escolha o ícone do Cost Explorer (gráfico de barras) ao lado do título Reserved Instances (Instâncias reservadas).

   O Cost Explorer exibe o gráfico Monthly EC2 running hours costs and usage (Custos e uso por hora mensais de execução do EC2).
7. Limpe o filtro Usage Type Group (Grupo de tipos de uso) à direita do gráfico.

8. Escolha o período e a unidade de tempo para os quais você deseja examinar custos de uso.

   O seguinte exemplo mostra os custos mensais acumulados no ano referentes ao uso de instâncias de banco de dados sob demanda e reservadas.



   Os custos da instância de banco de dados reservada de janeiro até junho de 2021 referem-se a cobranças mensais para uma instância com Pagamento adiantado parcial, enquanto os custos em agosto de 2021 referem-se a uma cobrança única para uma instância com Adiantamento integral.

   O desconto de instâncias reservadas para a instância com Pagamento adiantado parcial expirou em junho de 2021, mas a instância de banco de dados não foi excluída. Após a data de validade, ela foi simplesmente cobrada com base na taxa sob demanda.

Amazon Aurora no nível gratuito da AWS

O Aurora PostgreSQL também está disponível no nível gratuito da AWS tal como descrito em Criar com a configuração expressa. Além das limitações da configuração expressa, as seguintes restrições se aplicam ao usar o Aurora PostgreSQL com o nível gratuito da AWS:

- Somente clusters criados com configuração expressa podem ser usados (a configuração completa não está disponível).
- Até quatro unidades de capacidade do Aurora (ACUs) por cluster.
- Armazenamento de até 1 GB por cluster.
- No máximo dois clusters e duas instâncias por conta.
- Não há suporte a Integrações ETL zero do Aurora Global Database.

Configuração de seu ambiente para Amazon Aurora

Antes de usar o Amazon Aurora pela primeira vez, execute as seguintes tarefas.

Tópicos

- Inscrever-se para uma Conta da AWS
- Criar um usuário com acesso administrativo
- Conceder acesso programático
- Determinar requisitos
- Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança

Caso você já tenha uma Conta da AWS, conheça os requisitos do Aurora e prefira usar os padrões para grupos de segurança do IAM e da VPC, vá para Conceitos básicos do Amazon Aurora .

Inscrever-se para uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas abaixo para criar uma.

Como cadastrar uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve para uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

A AWS envia um e-mail de confirmação depois que o processo de inscrição é concluído. A qualquer momento, é possível exibir as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se cadastrar em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o Centro de Identidade do AWS IAM e crie um usuário administrativo para não usar o usuário-raiz em tarefas cotidianas.

Proteger o Usuário raiz da conta da AWS

1. Faça login no Console de gerenciamento da AWS como o proprietário da conta ao escolher a opção Usuário-raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Fazer login como usuário-raiz no Guia do usuário do Início de Sessão da AWS.
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo MFA virtual para sua Conta da AWS de usuário-raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

   Para obter instruções, consulte Habilitar o Centro de Identidade do AWS IAM no Guia do usuário do Centro de Identidade do AWS IAM.
2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

   Para obter um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como a fonte de identidade, consulte Configurar o acesso dos usuários com o Diretório do Centro de Identidade do IAM padrão no Guia do usuário do Centro de Identidade do AWS IAM.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

  Para obter ajuda para fazer login usando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM.

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de logon único ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do Centro de Identidade do AWS IAM.

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com a AWS de fora do Console de gerenciamento da AWS. A forma de conceder acesso programático depende do tipo de usuário que está acessando a AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Recomendado) Use credenciais do console como credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os AWS SDKs ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Login para desenvolvimento local da AWS no Guia do usuário do AWS Command Line Interface. • Para AWS SDKs, consulte Login para desenvolvimento local da AWS no Guia de referência de AWS SDKs e ferramentas.

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o Centro de Identidade do AWS IAM no Guia do usuário da AWS Command Line Interface. • Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Determinar requisitos

O bloco de construção básico do Aurora é o cluster do banco de dados. Uma ou mais instâncias de banco de dados podem pertencer a um cluster de banco de dados. Um cluster de banco de dados fornece um endereço de rede chamado de endpoint de cluster. Seus aplicativos se conectam ao endpoint de cluster exposto pelo cluster de banco de dados sempre que precisam acessar os bancos de dados criados nesse cluster de banco de dados. As informações que você especifica ao criar o cluster de banco de dados controla elementos de configuração, como memória, mecanismo de banco de dados e versão, configuração de rede, segurança e períodos de manutenção.

Antes de criar um cluster de banco de dados e um grupo de segurança, você deve conhecer suas necessidades de rede e cluster de banco de dados. Veja aqui alguns fatores importantes a considerar:

- Requisitos de recurso – quais são os requisitos de memória e processador de seu aplicativo ou serviço? Você usará essas configurações ao determinar que classe de instância de banco de dados você usará quando criar seu cluster de banco de dados. Para conhecer especificações sobre as classes de instâncias de bancos de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.
- VPC, sub-rede e grupo de segurança – Seu cluster de banco de dados estará em uma nuvem privada virtual (VPC). As regras do grupo de segurança deve ser configuradas na conexão com um cluster de banco de dados. A lista a seguir descreve as regras para cada opção da VPC:
  - VPC padrão: se sua conta da AWS tiver uma VPC padrão na região da AWS, essa VPC estará configurada para oferecer suporte a clusters de bancos de dados. Se você especificar a VPC padrão ao criar o cluster de banco de dados:
    - Você deve criar um grupo de segurança da VPC que autorize conexões da aplicação ou serviço com o cluster de banco de dados do Aurora. Use a opção Security Group (Grupo de segurança) no console da VPC ou na AWS CLI para criar grupos de segurança da VPC. Para obter mais informações, consulte Etapa 3: Criar um grupo de segurança da VPC.
    - Você deve especificar o grupo de sub-redes de banco de dados padrão. Se este for o primeiro cluster de banco de dados que você criou na região da AWS, o Amazon RDS criará o grupo de sub-redes de banco de dados padrão quando criar o cluster de banco de dados.
  - VPC definida pelo usuário — Se quiser especificar uma VPC definida pelo usuário ao criar um cluster de banco de dados:
    - Você deve criar um grupo de segurança da VPC que autorize conexões da aplicação ou serviço com o cluster de banco de dados do Aurora. Use a opção Security Group (Grupo de segurança) no console da VPC ou na AWS CLI para criar grupos de segurança da VPC. Para obter mais informações, consulte Etapa 3: Criar um grupo de segurança da VPC.
    - A VPC deve atender a certos requisitos para hospedar clusters de bancos de dados, como ter pelo menos duas sub-redes, cada uma em uma zona de disponibilidade separada. Para obter mais informações, consulte Amazon VPC e Amazon Aurora.
    - Você deve especificar um grupo de sub-redes de banco de dados que defina quais sub-redes nessa VPC podem ser usadas pelo cluster de banco de dados. Para obter informações, consulte a seção sobre grupos de sub-redes de banco de dados, em Trabalhar com um cluster de banco de dados em uma VPC.

- Alta disponibilidade: você precisa de suporte a failover? No Aurora, uma implantação multi-AZ cria uma instância primária e réplicas do Aurora. Você pode configurar a instância primária e réplicas do Aurora para estarem em zonas de disponibilidade diferentes para suporte de failover. Para manter a alta disponibilidade, recomendamos as implantações multi-AZ para cargas de trabalho de produção. Para fins de desenvolvimento e teste, você pode usar uma implantação que não seja multi-AZ. Para obter mais informações, consulte Alta disponibilidade do Amazon Aurora.
- Políticas do IAM: sua conta da AWS tem políticas que concedem as permissões necessárias para executar operações do Amazon RDS? Quando você se conecta à AWS usando credenciais do IAM, sua conta do IAM deve ter políticas do IAM que concedam as permissões necessárias para realizar operações do Amazon RDS. Para obter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.
- Portas abertas: Em que porta TCP/IP seu banco de dados ouvirá? O firewall de algumas empresas pode bloquear conexões com a porta padrão para o seu mecanismo de banco de dados. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o novo cluster de banco de dados. Observe que, depois de criar um cluster de banco de dados que escuta em uma porta específica, você pode alterar essa porta modificando o cluster de banco de dados.
- Região da AWS: Em qual região da AWS você quer seu banco de dados? Ter o banco de dados próximo do aplicativo ou do serviço Web pode reduzir a latência da rede. Para obter mais informações, consulte Regiões e zonas de disponibilidade.

Quando você tiver as informações necessárias para criar o grupo de segurança e o cluster de banco de dados, continue na próxima etapa.

Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança

Seu cluster de banco de dados será criado em uma VPC. Grupos de segurança fornecem acesso ao cluster de banco de dados na VPC. Eles atuam como um firewall para o cluster de banco de dados associada, controlando o tráfego de entrada e de saída no nível do cluster. Os clusters de bancos de dados são criados por padrão com um firewall e um grupo de segurança padrão que impede o acesso ao cluster de banco de dados. Portanto, você deve adicionar regras a um grupo de segurança que permitam que você se conecte ao cluster de banco de dados. Use as informações de rede e configuração que você determinou na etapa anterior para criar regras para permitir o acesso ao seu cluster de banco de dados.

Por exemplo, se você tiver um aplicativo que acessará um banco de dados no seu cluster de banco de dados em uma VPC, deverá adicionar uma regra de TCP personalizada que especifique o alcance da porta e os endereços IP que a aplicação usará para acessar o banco de dados. Se tiver uma aplicação em uma instância do Amazon EC2, você poderá usar o grupo de segurança configurado para a instância do Amazon EC2.

Você pode configurar a conectividade entre uma instância do Amazon EC2 e um cluster de banco de dados ao criar o cluster de banco de dados. Para obter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2.

> **Tip**
>
> Você pode configurar a conectividade de rede entre uma instância do Amazon EC2 e um cluster de banco de dados automaticamente ao criar o cluster de banco de dados. Para obter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2.

Para obter informações sobre como conectar recursos no Amazon Lightsail aos seus clusters de banco de dados, consulte Conectar recursos do Lightsail aos Serviços da AWS usando peering de VPC.

Para obter mais informações sobre como criar uma VPC para uso com o Aurora, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4). Para obter informações sobre cenários comuns para acessar uma instância de banco de dados, consulte Cenários para acessar um cluster de banco de dados em uma VPC.

Para criar um grupo de segurança de VPC

1. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.

   > **Note**
   >
   > Verifique se você está no console da VPC, não no console do RDS.

2. No canto superior direito do Console de gerenciamento da AWS, escolha a região da AWS na qual quer criar o grupo de segurança da VPC e o cluster de banco de dados. Na lista de recursos da Amazon VPC para essa região da AWS, você deve ver pelo menos uma VPC e

várias sub-redes. Caso contrário, significa que não há uma VPC padrão na região da AWS em questão.

3. No painel de navegação, escolha Grupos de segurança.
4. Escolha Create grupo de segurança (Criar grupo de segurança).

   A página Create grupo de segurança (Criar grupo de segurança) é exibida.
5. Em Basic details (Detalhes básicos), insira o Security group name (Nome do grupo de segurança) e a Description (Descrição). Para VPC, escolha a VPC na qual você deseja criar seu cluster de banco de dados.
6. Em Inbound rules (Regras de entrada), escolha Add rule (Adicionar regra).

   a. Em Type (Tipo), escolha Custom TCP (TCP personalizada).

   b. Em Port range (Intervalo de portas), insira o valor da porta a ser usado para o cluster de banco de dados.

   c. Em Source (Origem), selecione um nome de grupo de segurança ou digite o intervalo de endereços IP (valor CIDR) de onde você acessará o cluster de banco de dados. Se você selecionar My IP (Meu IP), isso concederá acesso ao cluster de banco de dados do endereço IP detectado no navegador.
7. Se você precisar adicionar mais endereços IP ou intervalos de portas diferentes, escolha Add rule (Adicionar regra) e insira as informações para a regra.
8. (Opcional) Em Outbound rules (Regras de saída), adicione regras para o tráfego de saída. Por padrão, todo tráfego de saída é permitido.
9. Escolha Create grupo de segurança (Criar grupo de segurança).

Você pode usar o grupo de segurança de VPC que acabou de criar como o grupo de segurança do seu cluster de banco de dados quando você o criar.

> **Note**
>
> Se usar uma VPC padrão, será criada para você um grupo de sub-redes padrão distribuídas por todas as sub-redes da VPC. Ao criar um cluster de banco de dados, você pode selecionar a VPC padrão e usar default (padrão) em DB Subnet Group (Grupo de sub-redes de banco de dados).

Depois de concluir os requisitos de configuração, você pode criar um cluster de banco de dados usando seus requisitos e grupo de segurança seguindo as instruções em Criar um cluster de bancos de dados do Amazon Aurora. Para obter informações sobre como começar criando um cluster de banco de dados que usa um mecanismo de banco de dados específico, consulte Conceitos básicos do Amazon Aurora.

Conceitos básicos do Amazon Aurora

Nesta seção, você aprenderá a criar e se conectar a um cluster de banco de dados Aurora usando o Amazon RDS.

Os procedimentos a seguir são tutoriais que demonstram as noções básicas de introdução ao Aurora. As seções posteriores apresentam conceitos e procedimentos do Aurora mais avançados, como os tipos diferentes de endpoints e como dimensionar clusters do Aurora.

> **Important**
>
> É necessário concluir as tarefas em Configuração de seu ambiente para Amazon Aurora antes de criar ou se conectar a um cluster de banco de dados.

Tópicos

- Criar um cluster de banco de dados do Aurora MySQL e se conectar a ele
- Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele
- Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora

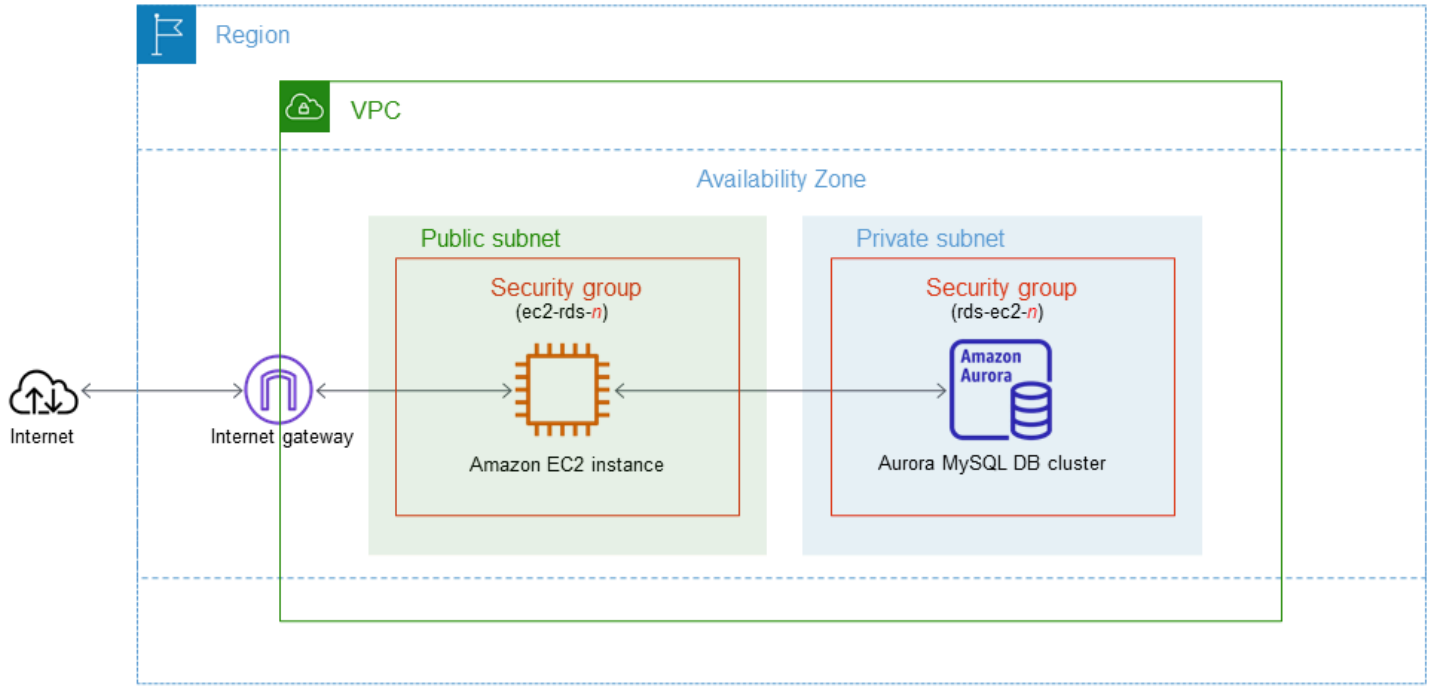
Criar um cluster de banco de dados do Aurora MySQL e se conectar a ele

Este tutorial cria uma instância do EC2 e um cluster de banco de dados do Aurora MySQL. O tutorial mostra como acessar o cluster de banco de dados pela instância do EC2 usando um cliente MySQL padrão. Como prática recomendada, este tutorial cria um cluster de banco de dados privado em uma nuvem privada virtual (VPC). Na maioria dos casos, outros recursos na mesma VPC, como instâncias do EC2, podem acessar o cluster de banco de dados, mas recursos fora da VPC não podem acessá-lo.

Depois de concluir o tutorial, haverá uma sub-rede pública e privada em cada zona de disponibilidade na sua VPC. Em uma zona de disponibilidade, a instância do EC2 está na sub-rede pública e a instância de banco de dados está na sub-rede privada.

> **Important**
>
> Não há cobrança para criar uma conta da AWS. No entanto, ao concluir este tutorial, você poderá incorrer em custos para os recursos da AWS utilizados. Se esses recursos não forem mais necessários após a conclusão do tutorial, você poderá excluí-los.

O diagrama a seguir mostrará a configuração quando o tutorial estiver completo.



Esse tutorial permite criar recursos usando um dos seguintes métodos:

1. Use o Console de gerenciamento da AWS: Etapa 1: Criar uma instância do EC2 e Etapa 2: Criar um cluster de banco de dados do Aurora MySQL
2. Use o CloudFormation para criar a instância de banco de dados e a instância do EC2: (Opcional) Criar VPC, instância do EC2 e cluster do Aurora MySQL usando o CloudFormation

O primeiro método usa Criação fácil para criar um cluster de banco de dados privado do Aurora MySQL com o Console de gerenciamento da AWS. Aqui, especifique apenas o tipo do mecanismo de banco de dados, o tamanho da instância de banco de dados e o identificador do cluster de banco de dados. A opção Easy create (Criação fácil) usa a configuração padrão para as outras opções de configuração.

Ao usar a opção Criação padrão, é possível especificar mais opções de configuração ao criar um cluster de banco de dados. Essas opções incluem configurações de disponibilidade, segurança, backups e manutenção. Para criar um cluster de banco de dados público, você deve usar a Criação padrão. Para mais informações, consulte the section called “Criar um cluster de banco de dados”.

Tópicos

- Pré-requisitos
- Etapa 1: Criar uma instância do EC2
- Etapa 2: Criar um cluster de banco de dados do Aurora MySQL
- (Opcional) Criar VPC, instância do EC2 e cluster do Aurora MySQL usando o CloudFormation
- Etapa 3: Conectar-se a um cluster de bancos de dados do Aurora MySQL
- Etapa 4: Excluir a instância do EC2 e o cluster de banco de dados
- (Opcional) Excluir a instância do EC2 e o cluster de banco de dados criado com o CloudFormation
- (Opcional) Conectar o cluster de banco de dados a uma função do Lambda

Pré-requisitos

Antes de começar, conclua as etapas nas seguintes seções:

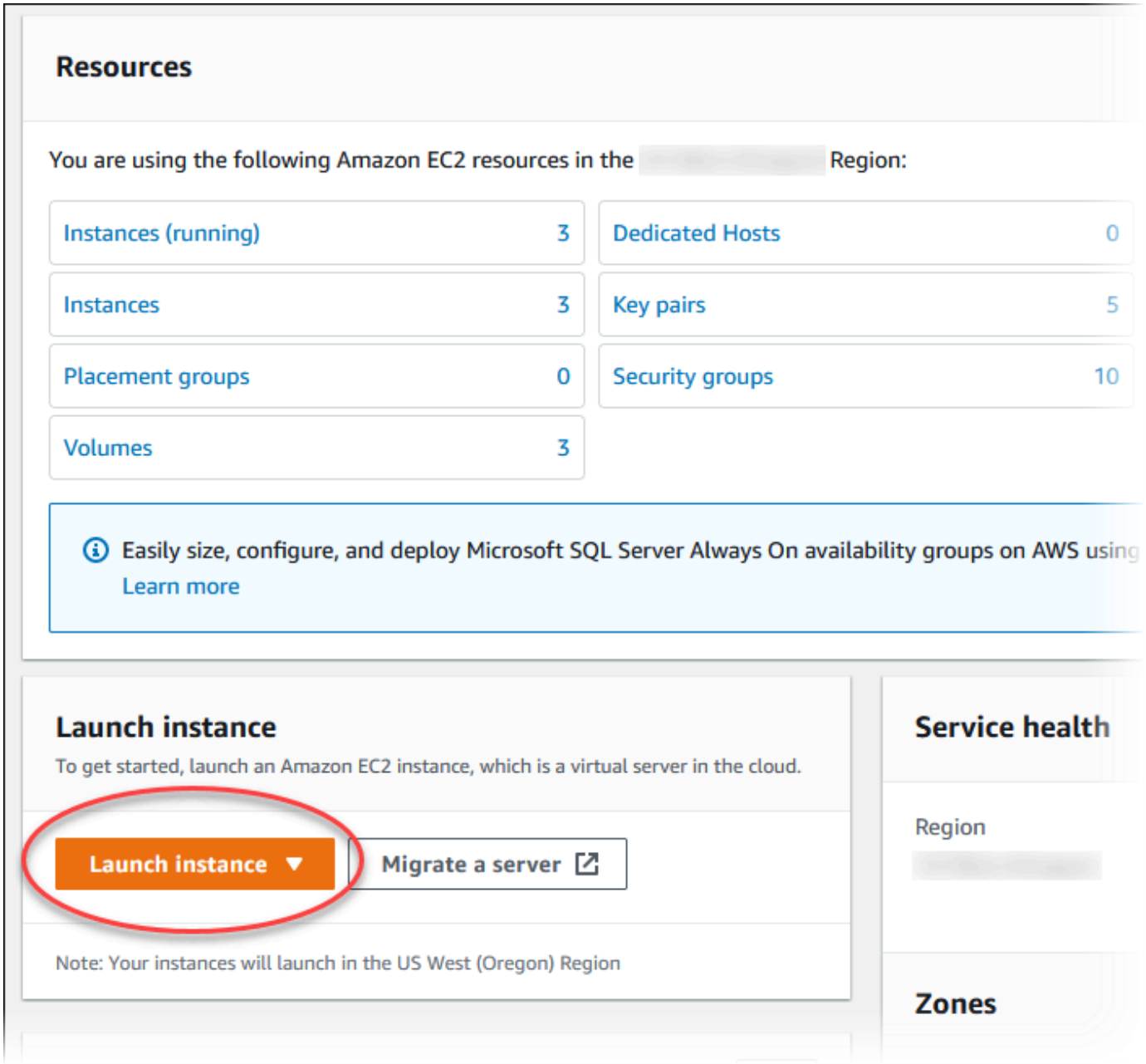
- Inscrever-se para uma Conta da AWS
- Criar um usuário com acesso administrativo

Etapa 1: Criar uma instância do EC2

Crie uma instância do Amazon EC2 que você usará para se conectar ao banco de dados.

Para criar uma instância do EC2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No canto superior direito do Console de gerenciamento da AWS, selecione a Região da AWS em que você deseja criar a instância do EC2.
3. Escolha Painel do EC2 e Executar instância, conforme mostrado na imagem a seguir.

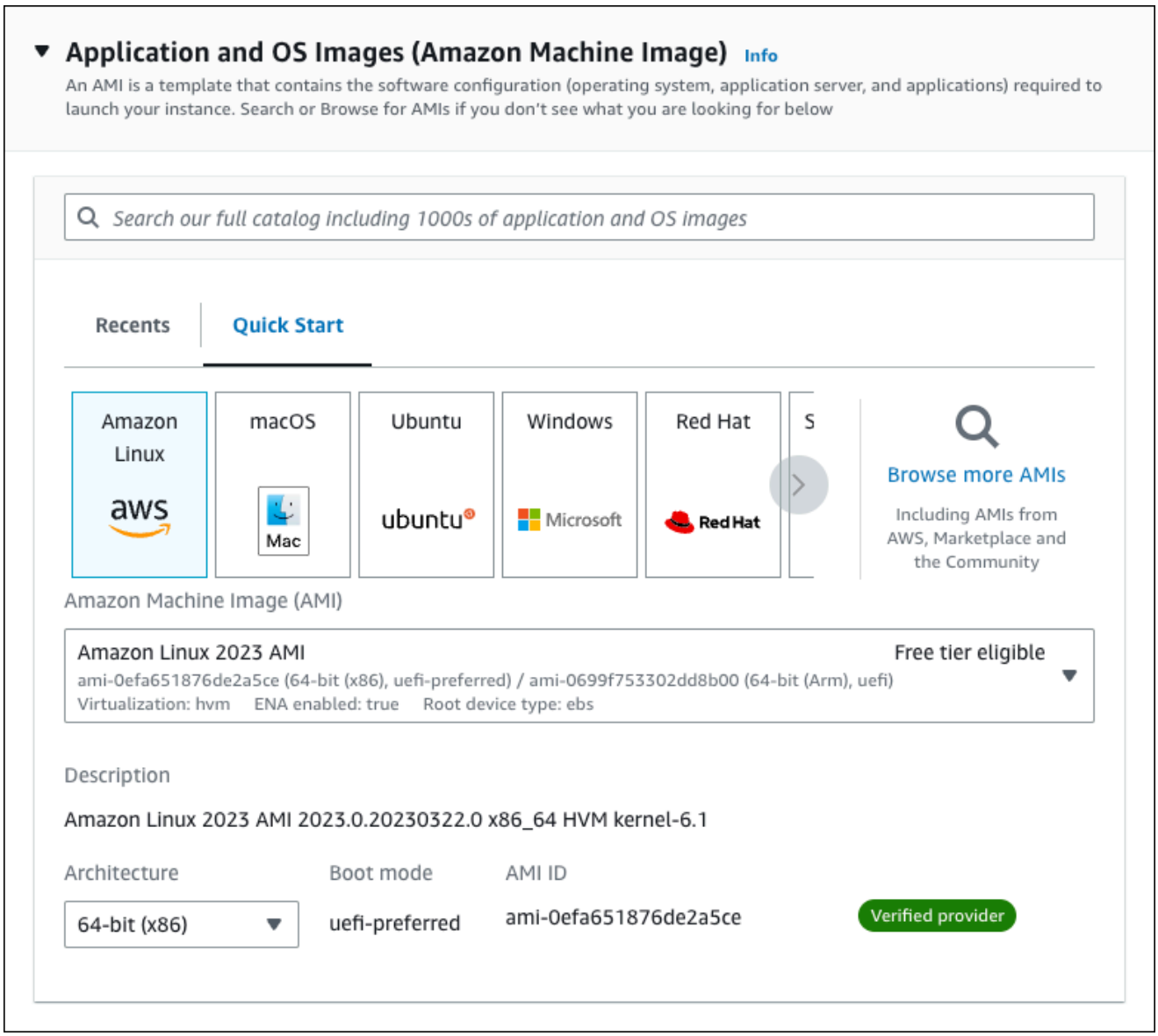


A página Iniciar uma instância é aberta.

4. Escolha as configurações a seguir na página Iniciar uma instância.

   a. Em Name and tags (Nome e etiquetas), em Name (Nome), insira **`ec2-database-connect`**.

   b. Em Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon), selecione Amazon Linux e, depois, AMI do Amazon Linux 2023. Mantenha as seleções padrão nas outras opções.



c. Em Instance type (Tipo de instância), escolha t2.micro.

d. Em Key pair (login) (Par de chaves (login)), escolha um Key pair name (Nome do par de chaves) para usar um par de chaves existente. Para criar um par de chaves para a instância do Amazon EC2, escolha Create new key pair (Criar um novo key pair), depois use a janela Create key pair (Criar par de chaves) para criá-lo.

   Consulte mais informações sobre como criar um par de chaves em Criar um par de chaves no Guia do usuário do Amazon EC2.

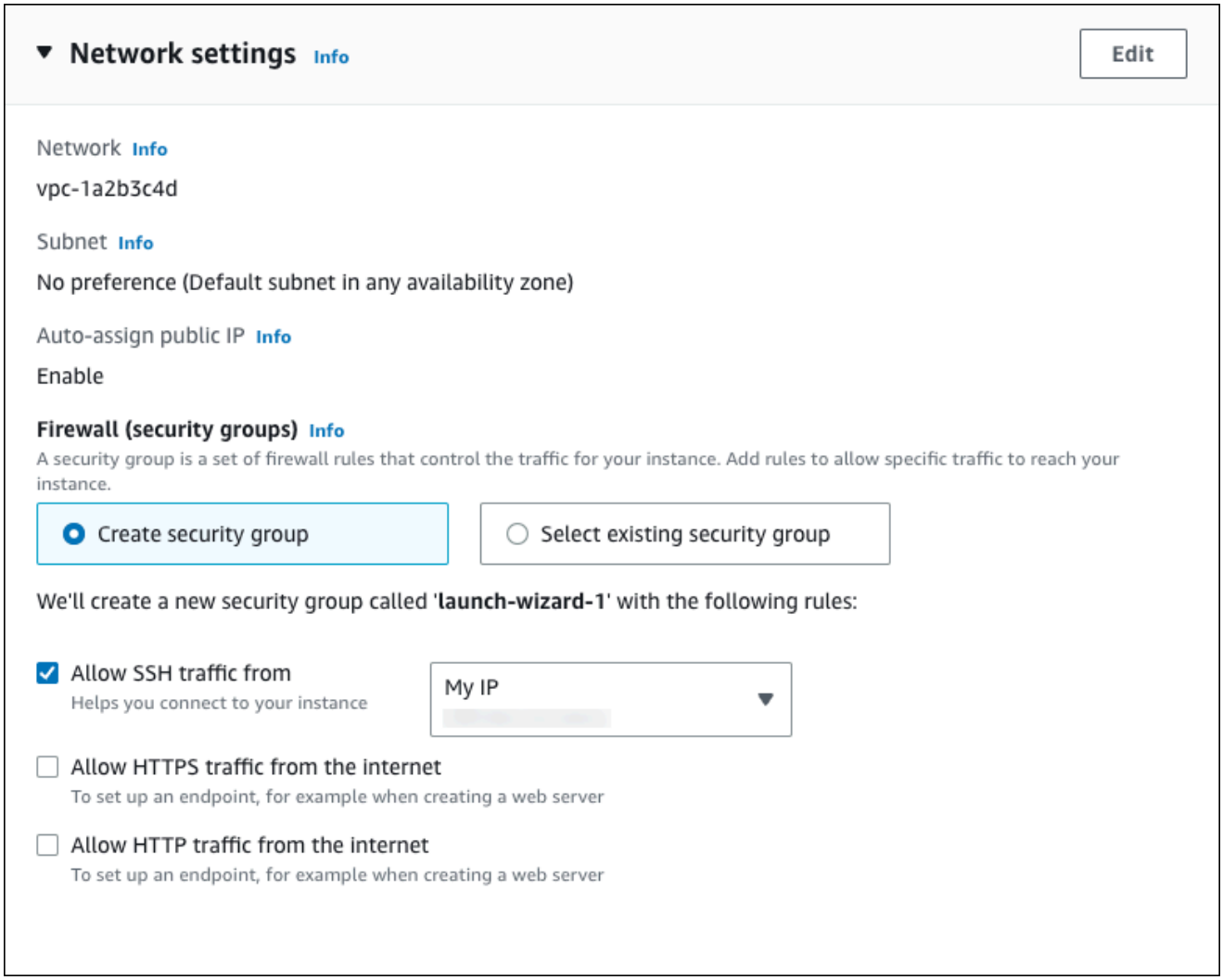
e. Para Permitir tráfego SSH, em Configurações de rede, selecione a origem das conexões SSH com a instância do EC2.

Você pode escolher My IP (Meu IP) se o endereço IP exibido estiver correto para conexões SSH. Caso contrário, determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, é possível usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP é 192.0.2.1/32.

Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Em caso afirmativo, determine o intervalo de endereços IP utilizado por computadores cliente.

> **Warning**
>
> Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas do EC2 usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias do EC2 usando SSH.

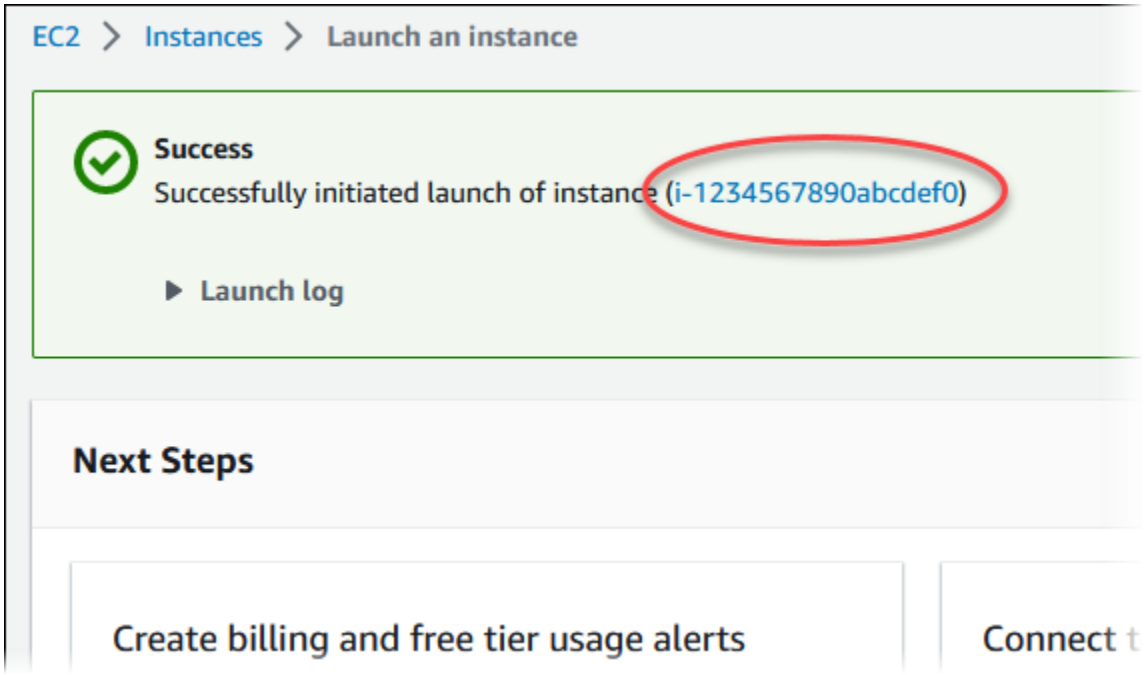
A imagem a seguir mostra um exemplo da seção Configurações de rede.



f. Mantenha os valores padrão para as seções restantes.

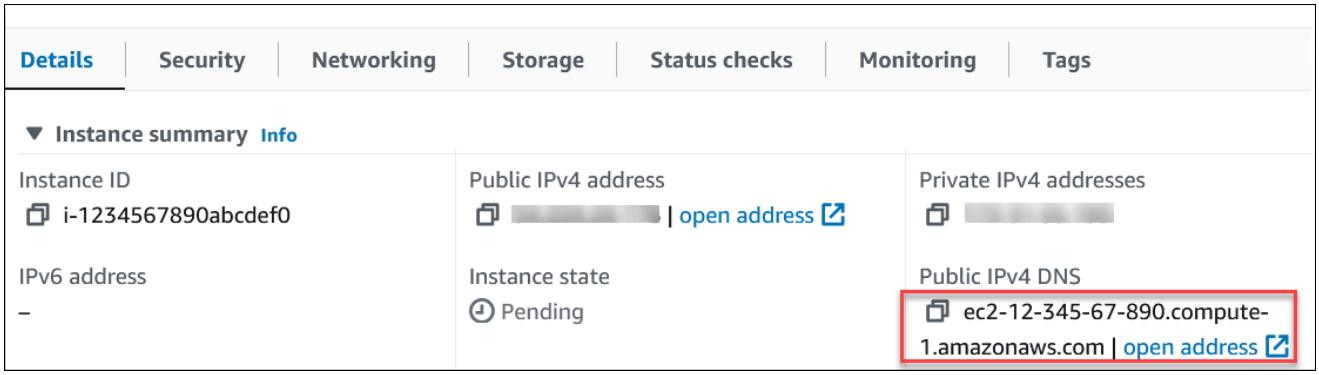
g. Revise um resumo da configuração da instância do EC2 no painel Resumo e, quando estiver com tudo pronto, escolha Executar instância.

5. Na página Status de inicialização, anote o identificador de sua nova instância do EC2, por exemplo: `i-1234567890abcdef0`.

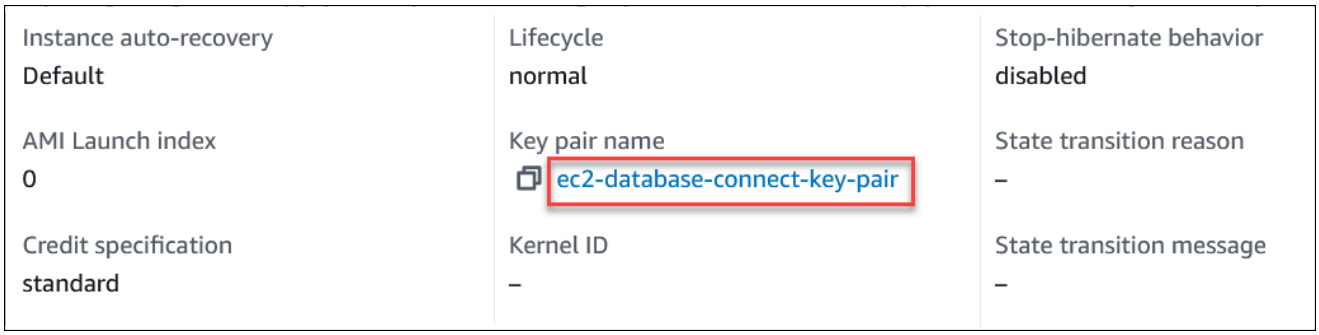


6. Selecione o identificador de instância do EC2 para abrir a lista de instâncias do EC2 e, depois, selecione sua instância do EC2.
7. Na guia Detalhes, observe os seguintes valores, necessários ao se conectar utilizando SSH:

   a. No Resumo da instância, observe o valor do DNS IPv4 público.



   b. Em Detalhes da instância, observe o valor do nome do par de chaves.



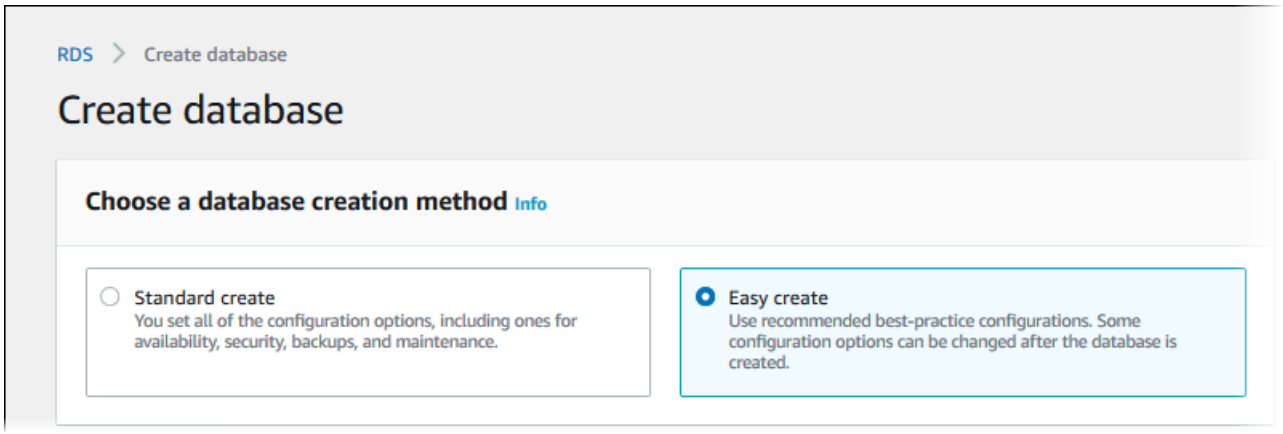
8. Espere até o Estado da instância para a instância do EC2 ficar como Executando antes de continuar.

Etapa 2: Criar um cluster de banco de dados do Aurora MySQL

Neste exemplo, use a opção Criação fácil para criar um cluster de banco de dados do Aurora MySQL com uma classe de instância de banco de dados db.r6g.large.

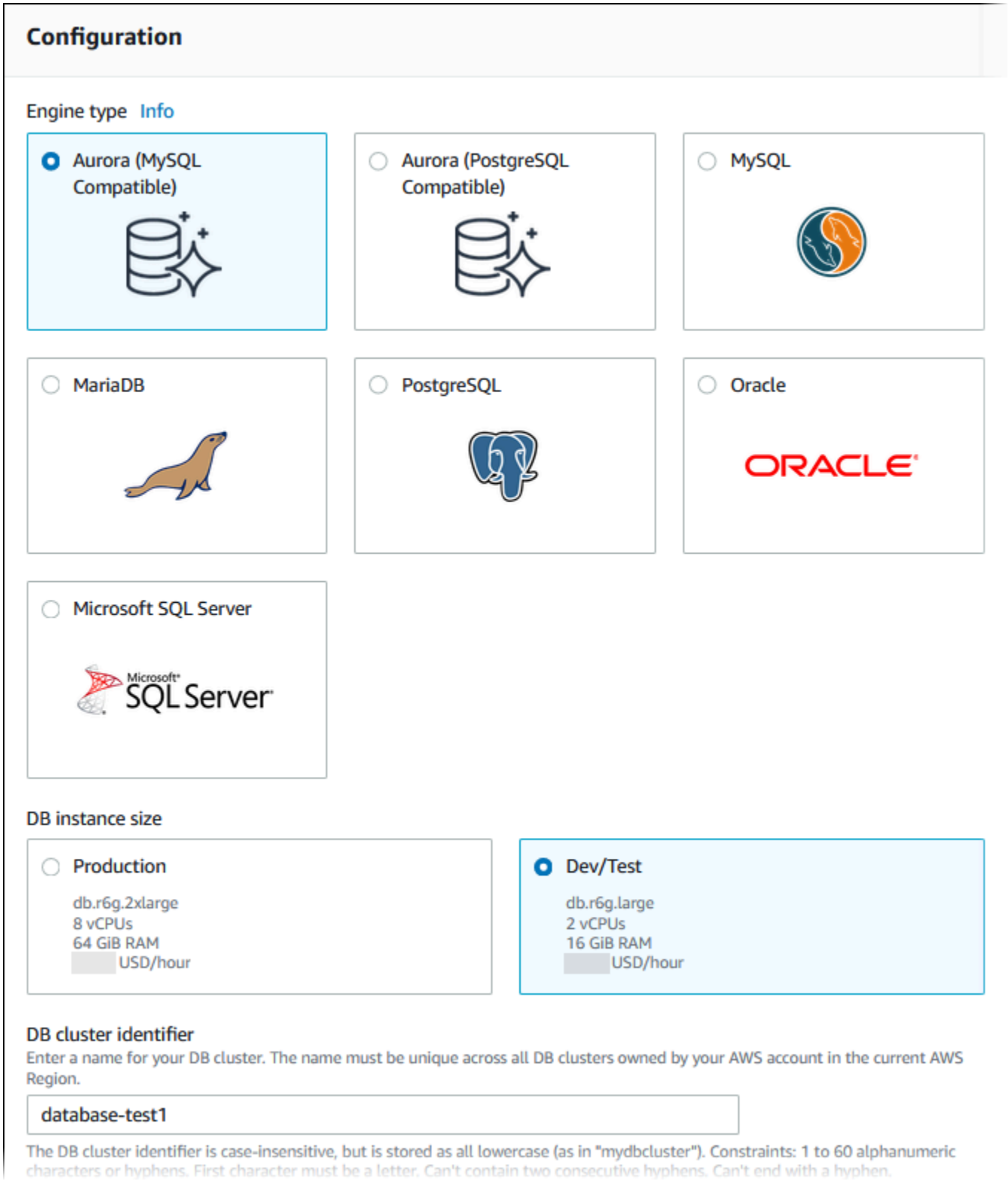
Como criar um cluster de banco de dados do Aurora MySQL com a criação fácil

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do console do Amazon RDS, selecione a Região da AWS na qual você deseja criar o cluster de banco de dados.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Selecione a opção Create database (Criar banco de dados) e verifique se a opção Easy Create (Criação fácil) está habilitada.



5. Em Configuração, selecione Aurora (compatível com MySQL) para Tipo de mecanismo.
6. Em DB instance size (Tamanho da instância de banco de dados), escolha Dev/Test (Desenvolvimento/teste).
7. Em Identificador do cluster de banco de dados, insira **`database-test1`**.

   A página Create database (Criar banco de dados) deve ser semelhante à imagem a seguir.

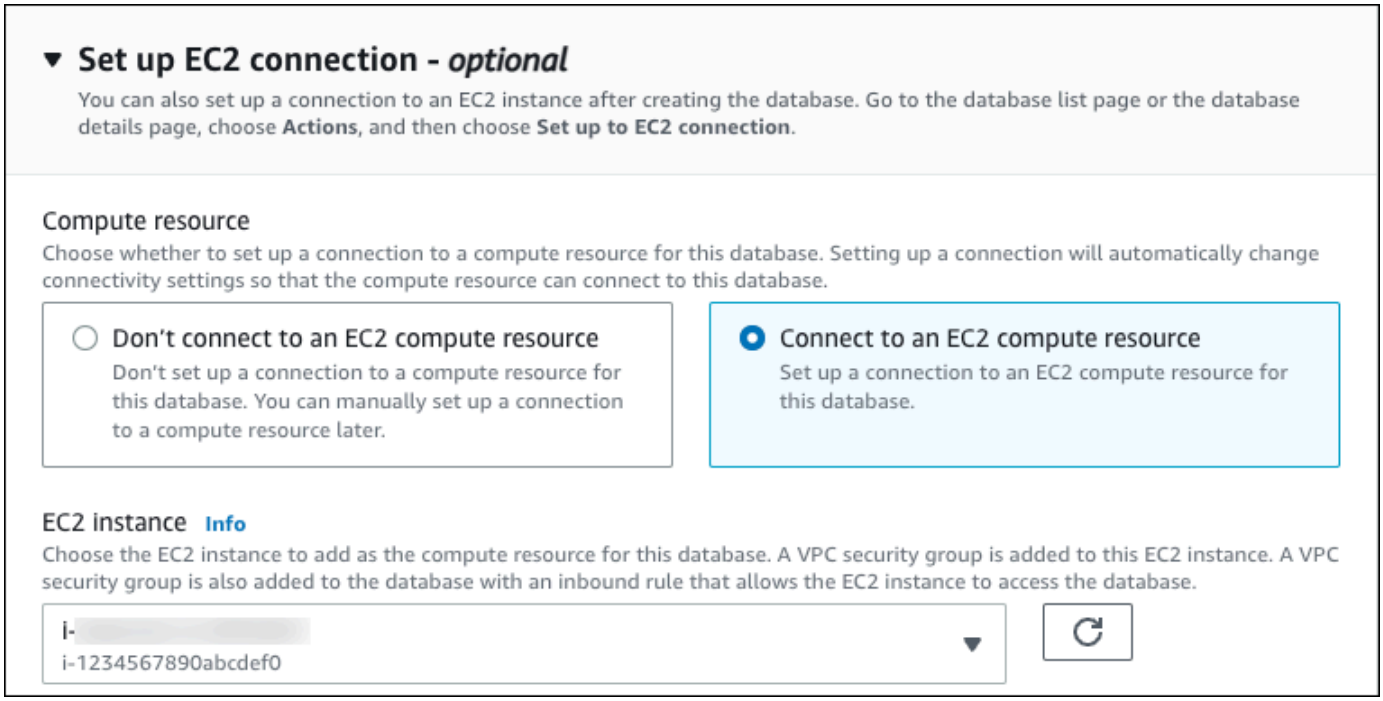


8. Em Nome do usuário principal, insira um nome para o usuário mestre ou deixe o nome padrão.
9. Para usar uma senha primária gerada automaticamente para o cluster de banco de dados, marque a opção Gerar uma senha automaticamente.

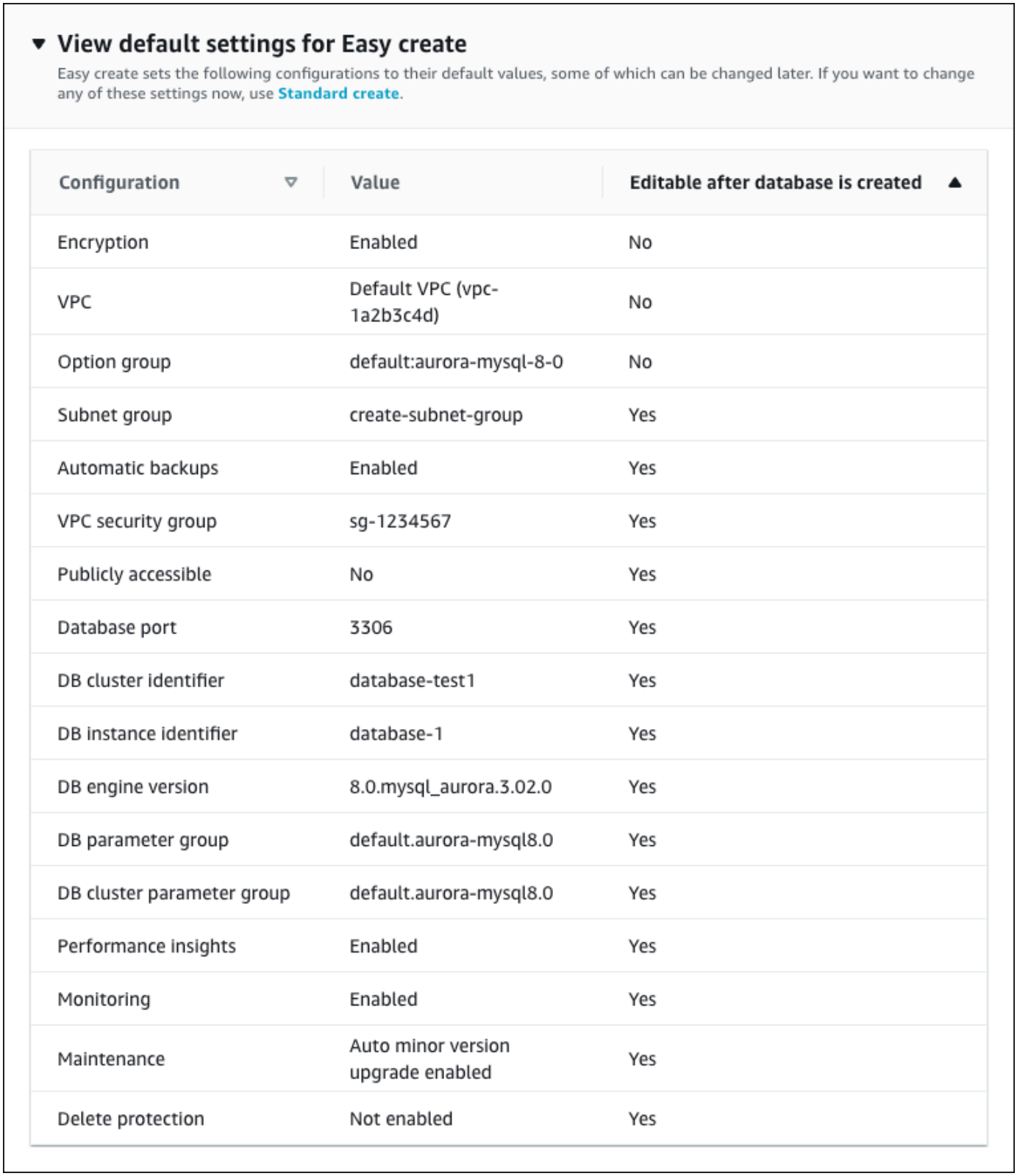
Para inserir sua senha primária, desmarque a opção Gerar uma senha automaticamente, depois insira a mesma senha em Senha primária e em Confirmar senha.

10. Para configurar uma conexão com a instância do EC2 que você criou anteriormente, abra Configurar conexão do EC2: opcional.

    Selecione Conectar-se a um recurso computacional do EC2. Selecione a instância do EC2 que você criou anteriormente.



11. Abra Visualizar as configurações padrão da criação fácil.



▼ **View default settings for Easy create**

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-mysql-8-0	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	8.0.mysql_aurora.3.02.0	Yes
DB parameter group	default.aurora-mysql8.0	Yes
DB cluster parameter group	default.aurora-mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Você pode examinar as configurações padrão usadas com a opção Easy Create (Criação fácil). A coluna Editável após a criação do banco de dados mostra quais opções podem ser alteradas após a criação do banco de dados.

- Se uma configuração tiver Não nessa coluna e você quiser uma configuração diferente, poderá usar a opção Criação padrão para criar o cluster de banco de dados.
- Se uma configuração tiver Sim nessa coluna e você quiser uma configuração diferente, poderá usar a opção Criação padrão para criar o cluster de banco de dados, ou modificar o cluster de banco de dados depois de criá-lo para alterar a configuração.

12. Selecione Criar banco de dados.

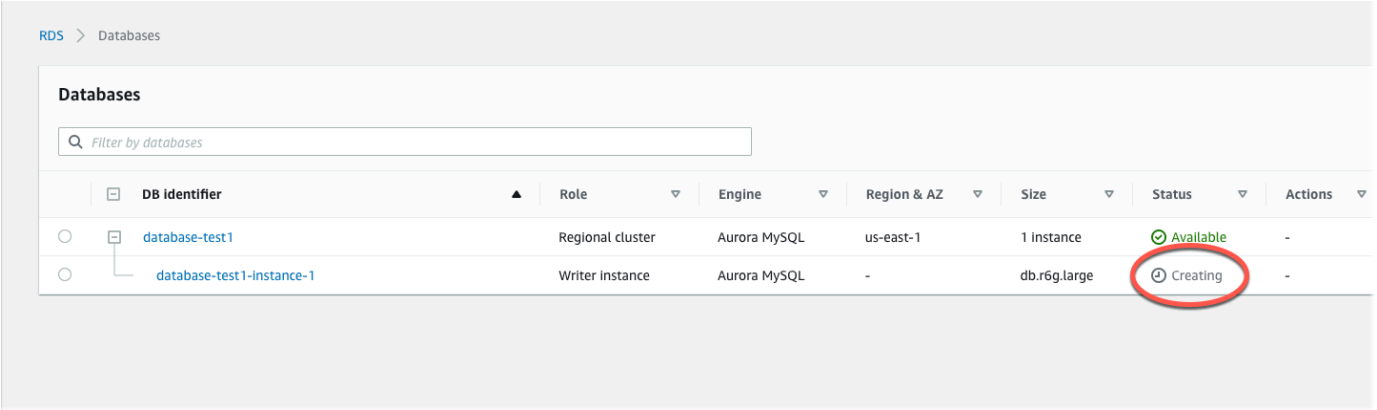
    Para visualizar o nome do usuário principal e a senha do cluster de banco de dados, escolha Visualizar detalhes da credencial.

    Use o nome de usuário e a senha que aparecem para se conectar ao cluster de banco de dados como o usuário principal.

    > **Important**
    >
    > Você não pode visualizar a senha do usuário principal novamente. Caso você não a registre, talvez seja necessário alterá-la.
    > Se for necessário alterar a senha do usuário primário depois que o cluster de banco de dados estiver disponível, será possível modificar o cluster de banco de dados para fazer isso. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

13. Na lista Bancos de dados, selecione o nome do novo cluster de banco de dados MySQL para mostrar seus detalhes.

    A instância do gravador tem o status Criando até o cluster de banco de dados estar pronto para uso.



Quando o status da instância do gravador muda para Disponível, você pode se conectar ao cluster de banco de dados. Dependendo da classe da instância de banco de dados e da quantidade de armazenamento, pode levar até 20 minutos para que o novo cluster de banco de dados esteja disponível.

(Opcional) Criar VPC, instância do EC2 e cluster do Aurora MySQL usando o CloudFormation

Em vez de usar o console para criar a VPC, a instância do EC2 e o cluster de banco de dados do Aurora MySQL, você pode usar o CloudFormation para provisionar recursos da AWS tratando a infraestrutura como código. Para ajudar a organizar os recursos da AWS em unidades menores e mais gerenciáveis, você pode usar a funcionalidade de pilha aninhada do CloudFormation. Consulte mais informações em Criar uma pilha no console do CloudFormation e Trabalhar com pilhas aninhadas.

> **Important**
>
> O CloudFormation é gratuito, mas os recursos que o CloudFormation são ativos. Você incorre nas taxas de uso padrão para esses recursos até que os encerre. Para obter mais informações, consulte Definição de preço do Amazon Aurora.

Para criar recursos usando o console do CloudFormation, conclua as seguintes etapas:

- Etapa 1: baixar o modelo do CloudFormation
- Etapa 2: configurar recursos usando o CloudFormation

Baixar o modelo do CloudFormation

Um modelo do CloudFormation é um arquivo de texto JSON ou YAML que contém as informações da configuração dos recursos que você deseja criar na pilha. Esse modelo também cria uma VPC e um bastion host para você com o cluster do Aurora.

Para baixar o arquivo de modelo, abra o link Aurora MySQL CloudFormation template.

Na página do Github, clique no botão Baixar arquivo bruto para salvar o arquivo YAML do modelo.

Configurar recursos usando o CloudFormation

> **Note**
>
> Antes de iniciar esse processo, verifique se você tem um par de chaves para uma instância do EC2 na Conta da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2 e instâncias do Linux.

Ao usar o modelo do CloudFormation, você deve selecionar os parâmetros certos para garantir que os recursos sejam criados corretamente. Siga as etapas abaixo:

1. Faça login no Console de gerenciamento da AWS e abra o console CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Selecione Create Stack (Criar pilha).
3. Na seção Especificar modelo, selecione Fazer upload de um arquivo de modelo do computador e escolha Próximo.
4. Na página Especificar detalhes da pilha, defina os seguintes parâmetros:
   a. Defina o Nome da pilha como AurMySQLTestStack.
   b. Em Parâmetros, defina Zonas de disponibilidade selecionando duas zonas de disponibilidade.
   c. Em Configuração do bastion host do Linux, em Nome da chave, selecione um par de chaves para fazer login na instância do EC2.
   d. Em Configurações do bastion host do Linux, defina o Intervalo de IP permitido para o endereço IP. Para conectar instâncias do EC2 à VPC usando o Secure Shell (SSH), determine o endereço IP público usando o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP é 192.0.2.1/32.

> **Warning**
>
> Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas do EC2 usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias do EC2 usando SSH.

e. Em Configuração geral do banco de dados, defina a Classe da instância de banco de dados como db.r6g.large.

f. Defina o Nome do banco de dados como **`database-test1`**.

g. Em Nome de usuário principal do banco de dados, insira um nome para o usuário principal.

h. Defina a Gerenciar senha de usuário principal do banco de dados com o Secrets Manager como `false` para esse tutorial.

i. Em Senha do banco de dados, defina uma senha de sua escolha. Lembre-se dessa senha para as etapas seguintes do tutorial.

j. Defina Implantação multi-AZ como `false`.

k. Deixe todas as outras configurações com os valores padrão. Clique em Próximo para continuar.

5. Na página Configurar opções de pilha, mantenha todas as opções padrão. Clique em Próximo para continuar.
6. Na página Revisar pilha, selecione Enviar depois de verificar as opções do banco de dados e do bastion host do Linux.

Depois que o processo de criação da pilha for concluído, visualize as pilhas com os nomes BastionStack e AMSNS para anotar as informações necessárias para se conectar ao banco de dados. Consulte mais informações em Viewing CloudFormation stack data and resources on the Console de gerenciamento da AWS.

Etapa 3: Conectar-se a um cluster de bancos de dados do Aurora MySQL

Você pode usar qualquer aplicação cliente padrão SQL para se conectar ao cluster de banco de dados. Neste exemplo, você se conecta a um cluster de banco de dados do Aurora MySQL usando o cliente da linha de comando mysql.

Como se conectar ao cluster de banco de dados do Aurora MySQL

1. Encontre o endpoint (nome de DNS) e o número da porta da instância do gravador para seu cluster de banco de dados.

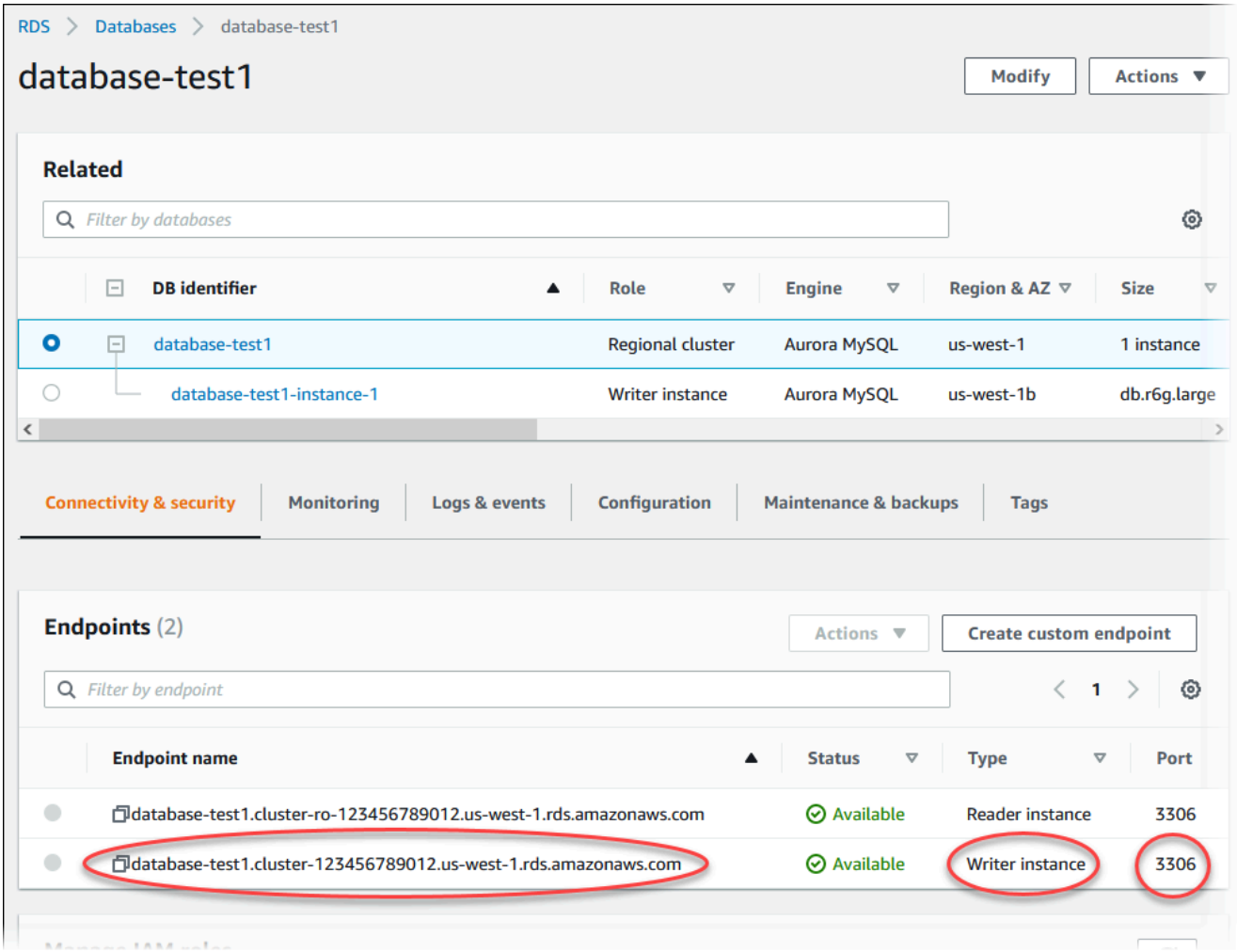
   a. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

   b. No canto superior direito do console do Amazon RDS, escolha a Região da AWS do cluster de banco de dados.

   c. No painel de navegação, escolha Bancos de dados.

   d. Selecione o nome do cluster de banco de dados do Aurora MySQL para exibir seus detalhes.

   e. Na guia Conectividade e segurança, copie o endpoint da instância do gravador. Além disso, anote o número da porta. Você precisará do endpoint e do número da porta para conectar-se ao cluster de banco de dados.



2. Conecte-se à instância do EC2 que você criou anteriormente, seguindo as etapas em Conecte-se à sua instância do Linux no Guia do usuário do Amazon EC2.

   Recomendamos que você se conecte à sua instância do EC2 utilizando SSH. Se o utilitário cliente SSH estiver instalado no Windows, Linux ou Mac, você poderá se conectar à instância utilizando o seguinte formato de comando:

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Por exemplo, suponha que `ec2-database-connect-key-pair.pem` esteja armazenado em `/dir1` no Linux e que o DNS IPv4 público para sua instância do EC2 seja `ec2-12-345-678-90.compute-1.amazonaws.com`. Depois, seu comando SSH teria a seguinte aparência:

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Obtenha as últimas correções de bugs e atualizações de segurança atualizando o software na instância do EC2. Para fazer isso, use o comando a seguir.

   > **Note**
   >
   > A opção `-y` instala as atualizações sem solicitar confirmação. Para examinar atualizações antes da instalação, omita essa opção.

   ```
   sudo dnf update -y
   ```

4. Para instalar o cliente da linha de comando mysql do MariaDB no Amazon Linux 2023, execute o seguinte comando:

   ```
   sudo dnf install mariadb105
   ```

5. Conecte-se ao cluster de banco de dados do Aurora MySQL. Por exemplo, insira o comando a seguir. Essa ação permite que você se conecte ao cluster de banco de dados do Aurora MySQL usando o cliente do MySQL.

   Substitua o endpoint da instância do gravador por *`endpoint`* e o nome do usuário principal usado por *`admin`*. Forneça a senha mestra usada quando for solicitada uma senha.

```
mysql -h endpoint -P 3306 -u admin -p
```

Depois de inserir a senha do usuário, você deverá ver uma saída semelhante à seguinte.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 217
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Para ter mais informações sobre a conexão com um cluster de banco de dados do Aurora MySQL, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora MySQL. Se você não conseguir se conectar ao seu cluster de banco de dados, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS.

Por questões de segurança, é uma prática recomendada usar conexões criptografadas. Use uma conexão MySQL não criptografada apenas quando o cliente e o servidor estiverem na mesma VPC e a rede for confiável. Para obter informações sobre como usar conexões criptografadas, consulte Conectar-se ao Aurora MySQL usando SSL.

6. Executar comandos SQL.

   Por exemplo, o seguinte comando SQL mostra a data e a hora atuais:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Etapa 4: Excluir a instância do EC2 e o cluster de banco de dados

Depois de se conectar e explorar a instância do EC2 e o cluster de banco de dados criados, exclua-os para não receber mais cobranças por eles.

Se você usou o CloudFormation para criar recursos, ignore essa etapa e passe para a próxima.

Como excluir a instância do EC2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, escolha Instâncias.
3. Selecione a instância do EC2 e escolha Estado da instância, Encerrar instância.
4. Quando a confirmação for solicitada, escolha Encerrar.

Consulte mais informações sobre como excluir uma instância do EC2 em Como encerrar uma instância no Guia do usuário do Amazon EC2.

Como excluir o cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados) e depois a instância de banco de dados associada ao cluster de banco de dados.
3. Em Actions, selecione Delete.
4. Desmarque a caixa Criar snapshot final?.
5. Conclua a confirmação e escolha Excluir.

Depois que todas as instâncias de banco de dados associadas a um cluster de banco de dados forem excluídas, o cluster de banco de dados será excluído automaticamente.

(Opcional) Excluir a instância do EC2 e o cluster de banco de dados criado com o CloudFormation

Se você usou o CloudFormation para criar recursos, exclua a pilha do CloudFormation depois de conectar e explorar a amostra de instância do EC2 e de cluster de banco de dados para não receber mais cobranças por eles.

Para excluir os recursos do CloudFormation

1. Abra o CloudFormation Console.
2. Na página Pilhas no console do CloudFormation, selecione a pilha raiz (a pilha sem o nome VPCStack, BastionStack ou AMSNS).

3. Escolha Excluir.
4. Selecione Excluir pilha quando a confirmação for solicitada.

Consulte mais informações de como excluir uma pilha no CloudFormation em Deleting a stack on the CloudFormation console no Guia do usuário do AWS CloudFormation.

(Opcional) Conectar o cluster de banco de dados a uma função do Lambda

Você também pode conectar o cluster de banco de dados do Aurora MySQL a um recurso de computação sem servidor do Lambda. As funções do Lambda permitem que você execute código sem provisionar nem gerenciar a infraestrutura. Uma função do Lambda também permite que você responda automaticamente a solicitações de execução de código em qualquer escala, de dezenas de eventos por dia a centenas por segundo. Para obter mais informações, consulte Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora.

Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele

Há duas maneiras de criar um cluster do Aurora PostgreSQL: configuração completa e configuração expressa. Consulte as seções abaixo para entender melhor esses dois tipos de criação.

Tópicos

- Criar com a configuração completa
- Criar com a configuração expressa

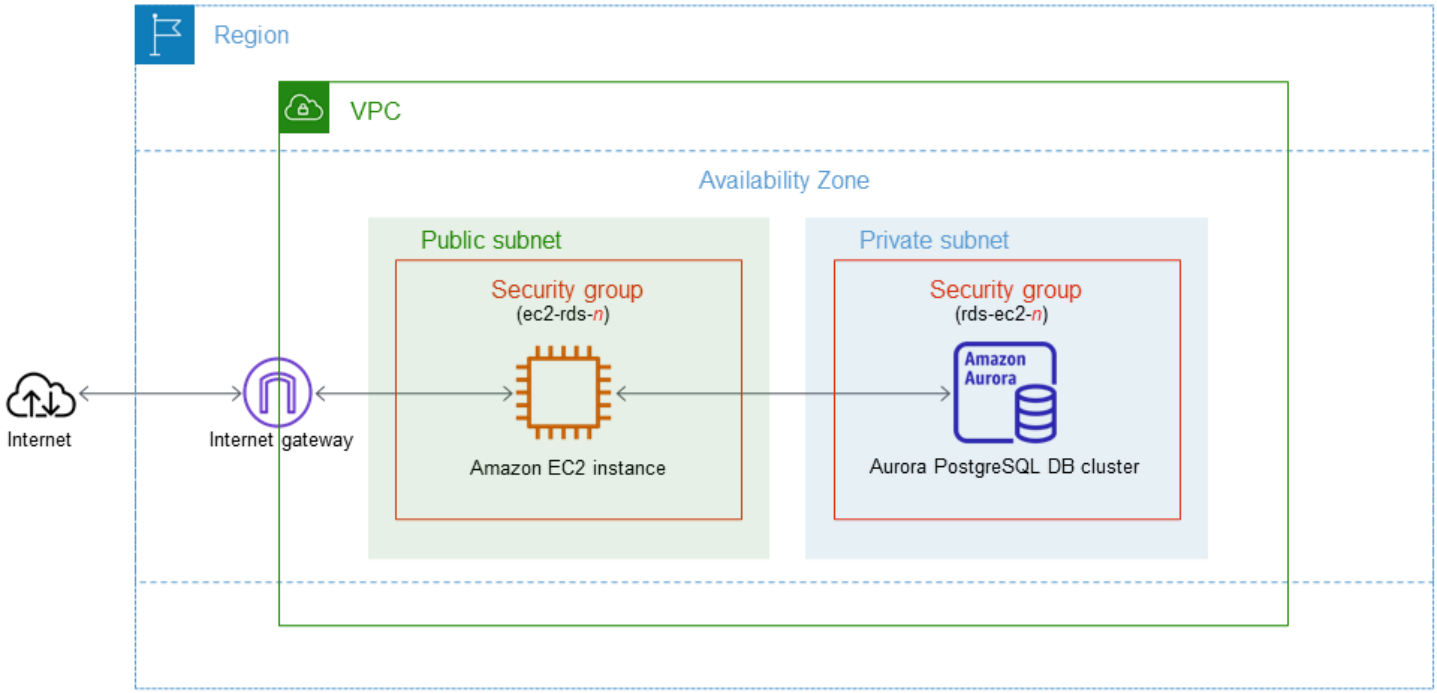
Criar com a configuração completa

Este tutorial cria uma instância do EC2 e um cluster de banco de dados do Aurora PostgreSQL. O tutorial mostra como acessar o cluster de banco de dados pela instância do EC2 usando um cliente PostgreSQL padrão. Como prática recomendada, este tutorial cria um cluster de banco de dados privado em uma nuvem privada virtual (VPC). Na maioria dos casos, outros recursos na mesma VPC, como instâncias do EC2, podem acessar o cluster de banco de dados, mas recursos fora da VPC não podem acessá-lo.

Depois de concluir o tutorial, haverá uma sub-rede pública e privada em cada zona de disponibilidade na sua VPC. Em uma zona de disponibilidade, a instância do EC2 está na sub-rede pública e a instância de banco de dados está na sub-rede privada.

> **Important**
>
> Não há cobrança para criar uma conta da AWS. No entanto, ao concluir este tutorial, você poderá incorrer em custos para os recursos da AWS utilizados. Se esses recursos não forem mais necessários após a conclusão do tutorial, você poderá excluí-los.

O diagrama a seguir mostrará a configuração quando o tutorial estiver completo.



Esse tutorial permite criar recursos usando um dos seguintes métodos:

1. Use o Console de gerenciamento da AWS: Etapa 1: Criar uma instância do EC2 e Etapa 2: Criar um cluster de banco de dados do Aurora PostgreSQL
2. Use o CloudFormation para criar a instância de banco de dados e a instância do EC2: (Opcional) Criar VPC, instância do EC2 e cluster do Aurora PostgreSQL usando o CloudFormation

O primeiro método usa Criação fácil para criar um cluster de banco de dados privado do Aurora PostgreSQL com o Console de gerenciamento da AWS. Aqui, especifique apenas o tipo do

mecanismo de banco de dados, o tamanho da instância de banco de dados e o identificador do cluster de banco de dados. A opção Easy create (Criação fácil) usa a configuração padrão para as outras opções de configuração.

Ao usar a opção Criação padrão, é possível especificar mais opções de configuração ao criar um cluster de banco de dados. Essas opções incluem configurações de disponibilidade, segurança, backups e manutenção. Para criar um cluster de banco de dados público, você deve usar a Criação padrão. Para mais informações, consulte the section called “Criar um cluster de banco de dados”.

Tópicos

- Pré-requisitos
- Etapa 1: Criar uma instância do EC2
- Etapa 2: Criar um cluster de banco de dados do Aurora PostgreSQL
- (Opcional) Criar VPC, instância do EC2 e cluster do Aurora PostgreSQL usando o CloudFormation
- Etapa 3: Conectar-se a um cluster de bancos de dados do Aurora PostgreSQL
- Etapa 4: Excluir a instância do EC2 e o cluster de banco de dados
- (Opcional) Excluir a instância do EC2 e o cluster de banco de dados criado com o CloudFormation
- (Opcional) Conectar o cluster de banco de dados a uma função do Lambda

Pré-requisitos

Antes de começar, conclua as etapas nas seguintes seções:

- Inscrever-se para uma Conta da AWS
- Criar um usuário com acesso administrativo

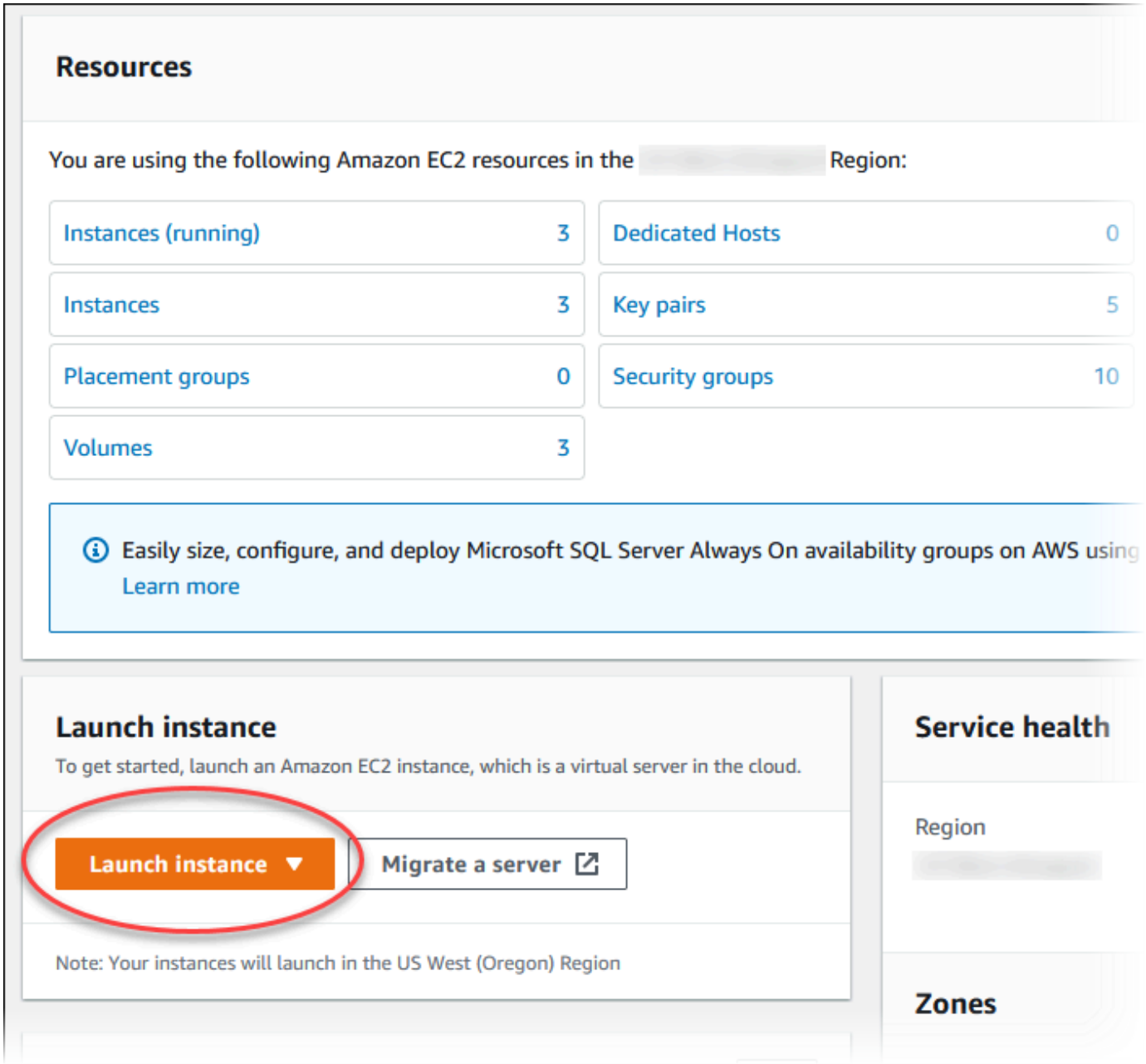
Etapa 1: Criar uma instância do EC2

Crie uma instância do Amazon EC2 que você usará para se conectar ao banco de dados.

Para criar uma instância do EC2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No canto superior direito do Console de gerenciamento da AWS, selecione a Região da AWS em que você deseja criar a instância do EC2.

3. Escolha Painel do EC2 e Executar instância, conforme mostrado na imagem a seguir.

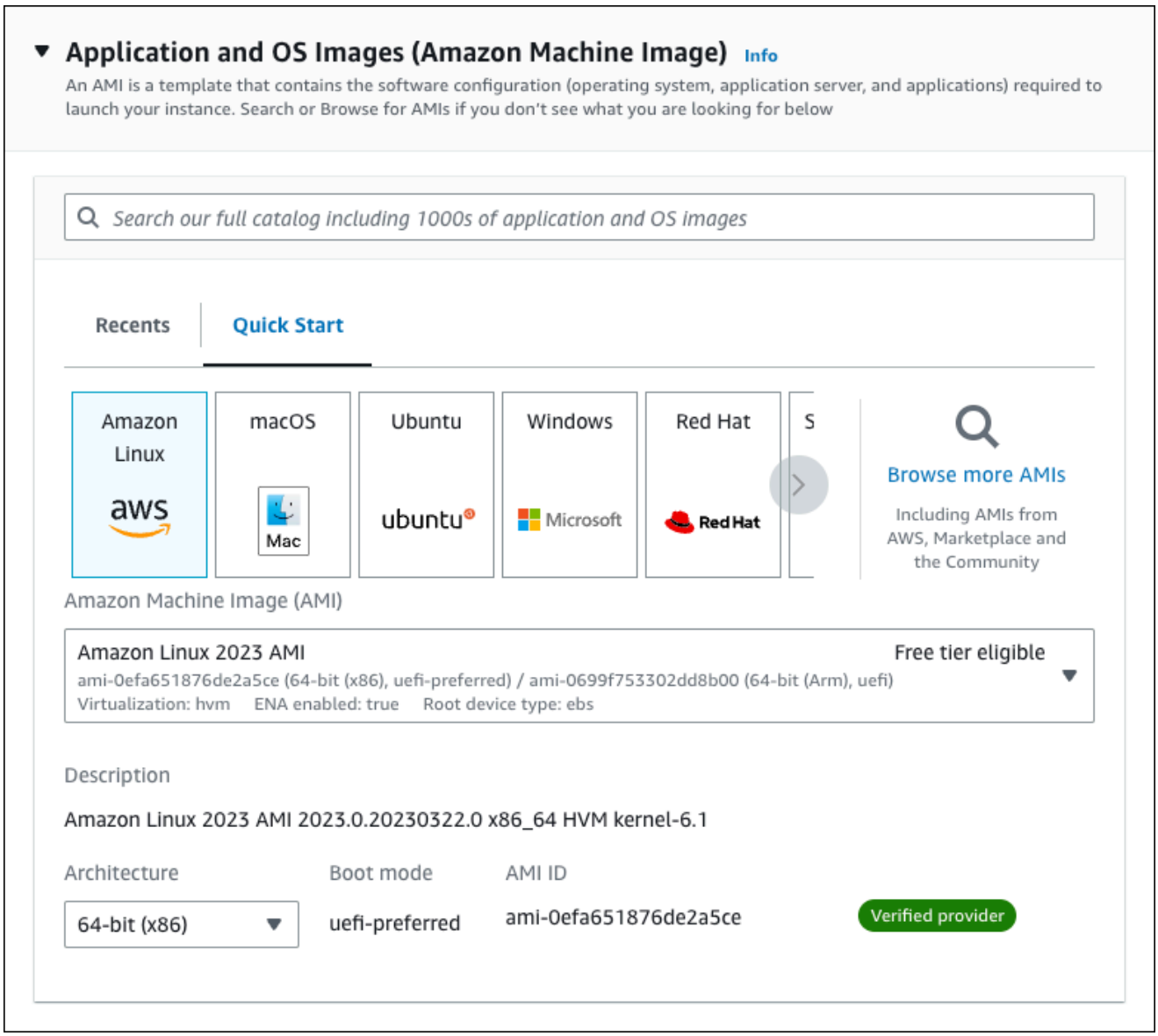


A página Iniciar uma instância é aberta.

4. Escolha as configurações a seguir na página Iniciar uma instância.

   a. Em Name and tags (Nome e etiquetas), em Name (Nome), insira **`ec2-database-connect`**.

   b. Em Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon), selecione Amazon Linux e, depois, AMI do Amazon Linux 2023. Mantenha as seleções padrão nas outras opções.



c. Em Instance type (Tipo de instância), escolha t2.micro.

d. Em Key pair (login) (Par de chaves (login)), escolha um Key pair name (Nome do par de chaves) para usar um par de chaves existente. Para criar um par de chaves para a instância do Amazon EC2, escolha Create new key pair (Criar um novo key pair), depois use a janela Create key pair (Criar par de chaves) para criá-lo.

   Consulte mais informações sobre como criar um par de chaves em Criar um par de chaves no Guia do usuário do Amazon EC2.

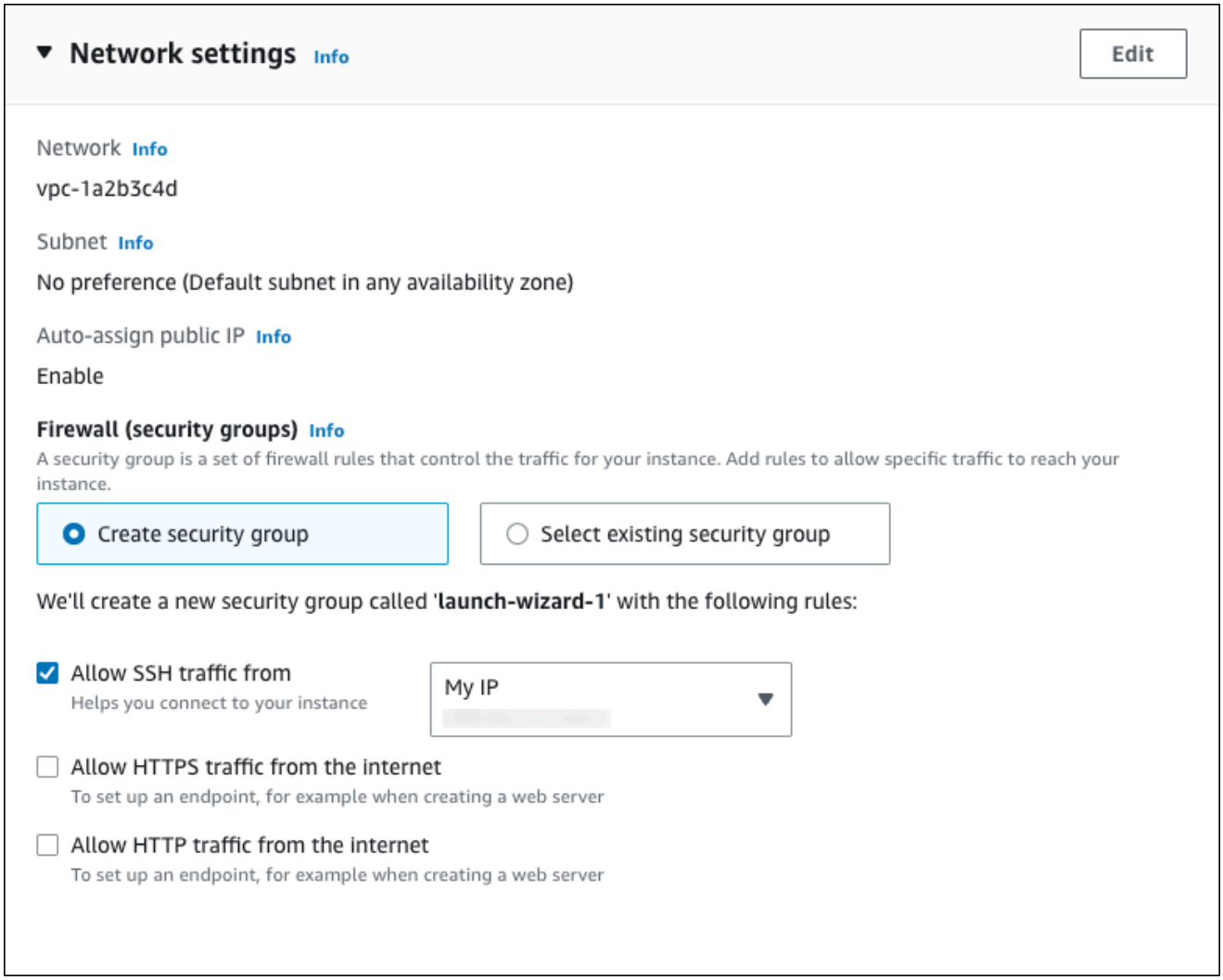
e. Para Permitir tráfego SSH, em Configurações de rede, selecione a origem das conexões SSH com a instância do EC2.

Você pode escolher My IP (Meu IP) se o endereço IP exibido estiver correto para conexões SSH. Caso contrário, determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, é possível usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP é 192.0.2.1/32.

Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Em caso afirmativo, determine o intervalo de endereços IP utilizado por computadores cliente.

> **Warning**
>
> Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas do EC2 usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias do EC2 usando SSH.

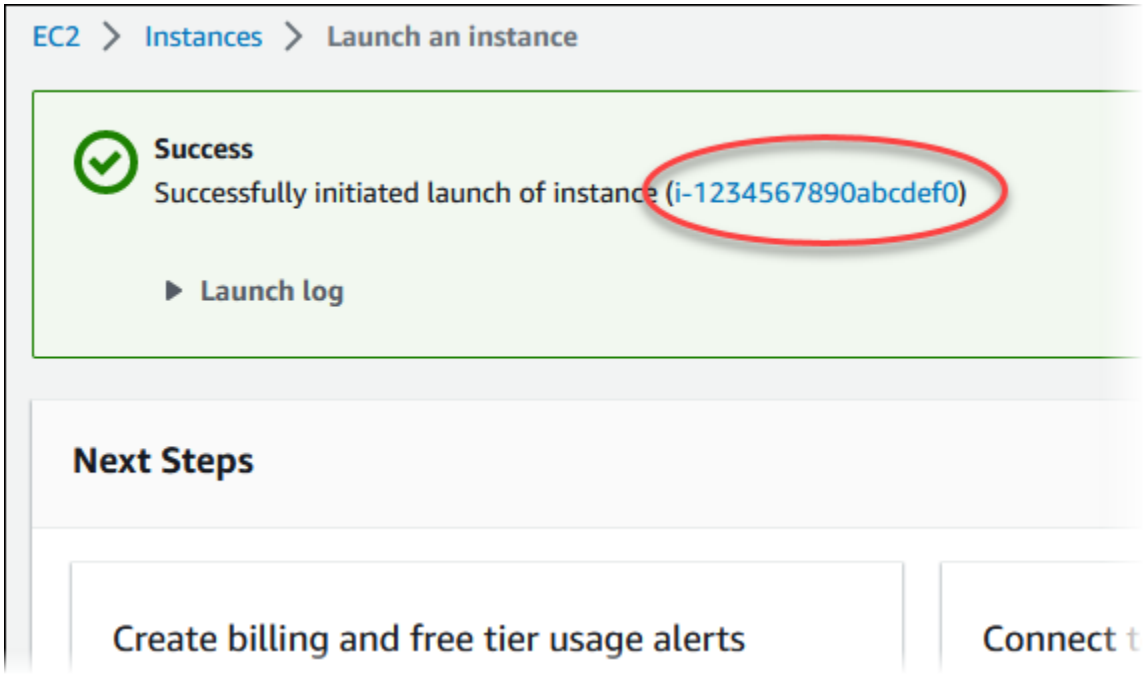
A imagem a seguir mostra um exemplo da seção Configurações de rede.



f. Mantenha os valores padrão para as seções restantes.

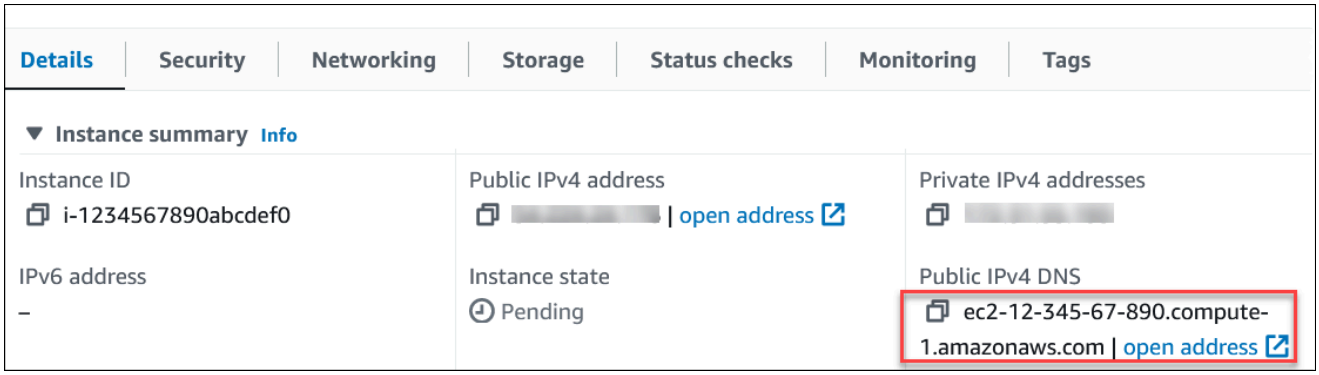
g. Revise um resumo da configuração da instância do EC2 no painel Resumo e, quando estiver com tudo pronto, escolha Executar instância.

5. Na página Status de inicialização, anote o identificador de sua nova instância do EC2, por exemplo: `i-1234567890abcdef0`.

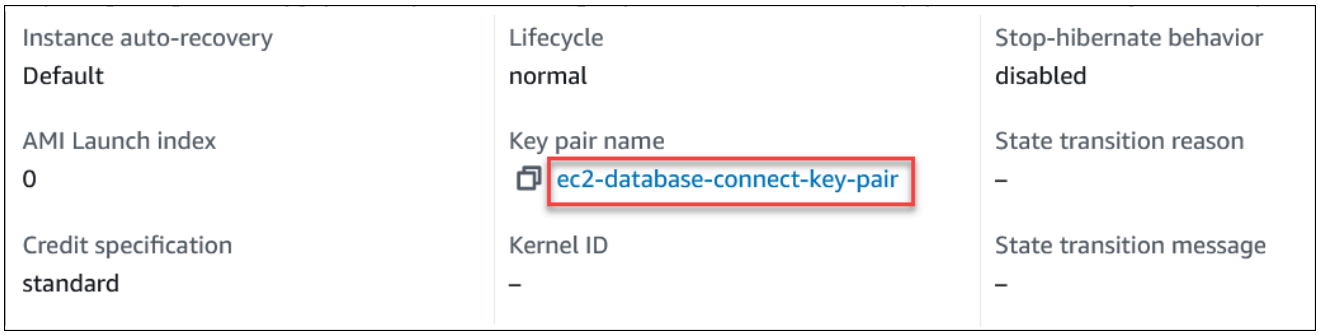


6. Selecione o identificador de instância do EC2 para abrir a lista de instâncias do EC2 e, depois, selecione sua instância do EC2.
7. Na guia Detalhes, observe os seguintes valores, necessários ao se conectar utilizando SSH:

   a. No Resumo da instância, observe o valor do DNS IPv4 público.



   b. Em Detalhes da instância, observe o valor do nome do par de chaves.



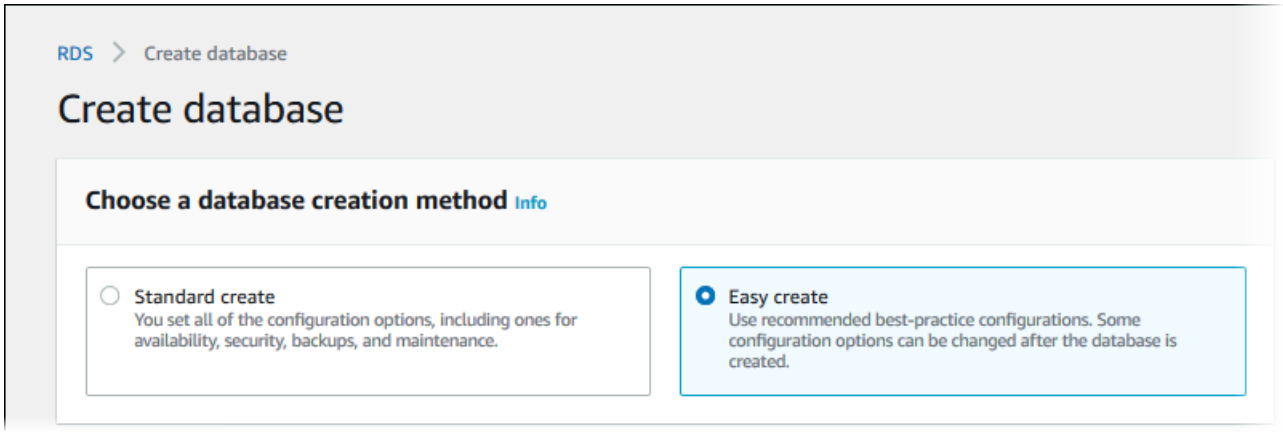
8. Espere até o Estado da instância para a instância do EC2 ficar como Executando antes de continuar.

Etapa 2: Criar um cluster de banco de dados do Aurora PostgreSQL

Neste exemplo, use a opção Criação fácil para criar um cluster de banco de dados do Aurora PostgreSQL com uma classe de instância de banco de dados db.t4g.large.

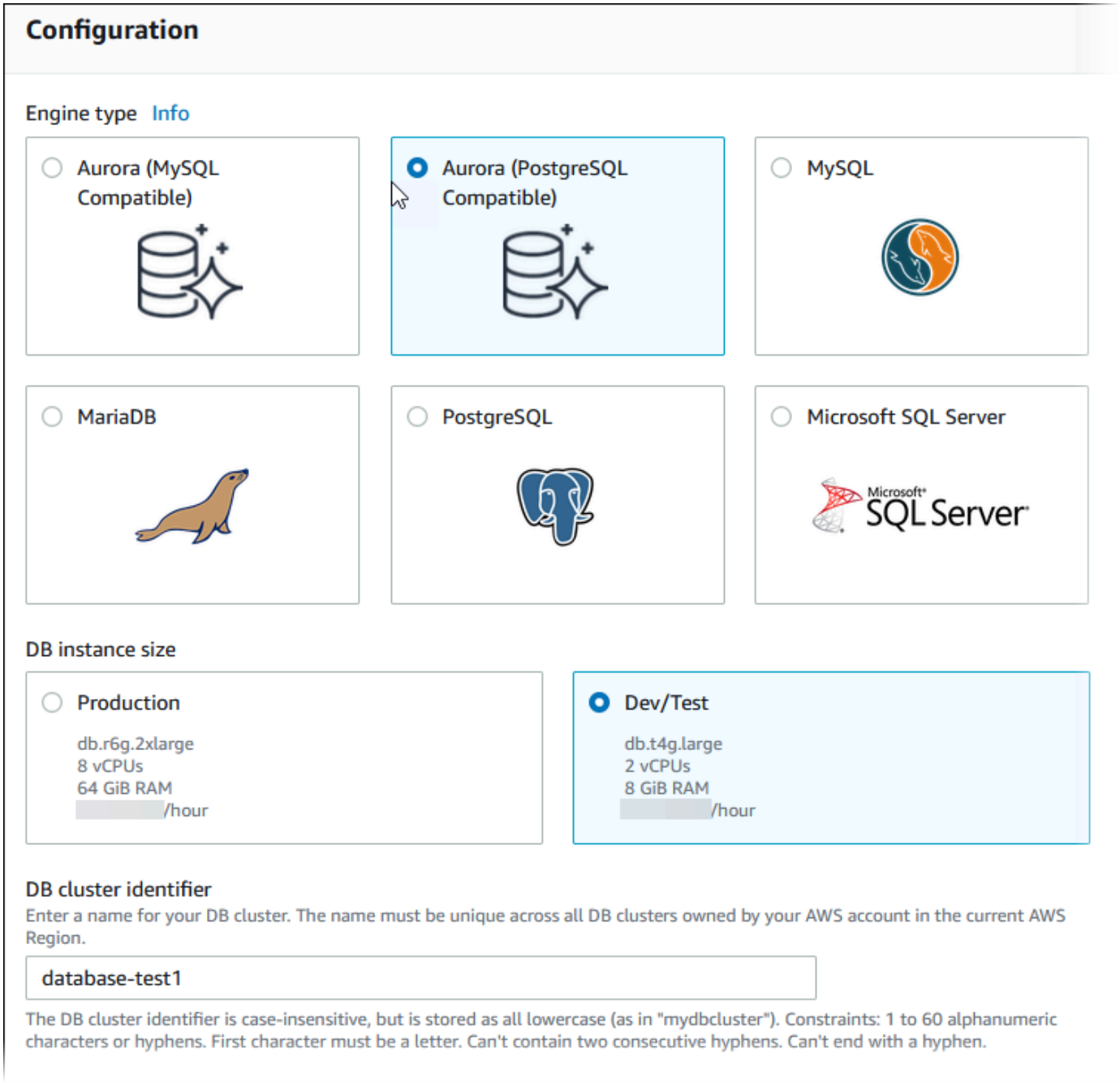
Como criar um cluster de banco de dados do Aurora PostgreSQL com a criação fácil

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do console do Amazon RDS, selecione a Região da AWS na qual você deseja criar o cluster de banco de dados.
3. No painel de navegação, escolha Bancos de dados.
4. Selecione a opção Criar banco de dados e verifique se a opção Criação fácil está selecionada.



5. Em Configuração, selecione Aurora (compatível com PostgreSQL) para Tipo de mecanismo.
6. Em DB instance size (Tamanho da instância de banco de dados), escolha Dev/Test (Desenvolvimento/teste).
7. Em Identificador do cluster de banco de dados, insira **`database-test1`**.

   A página Create database (Criar banco de dados) deve ser semelhante à imagem a seguir.

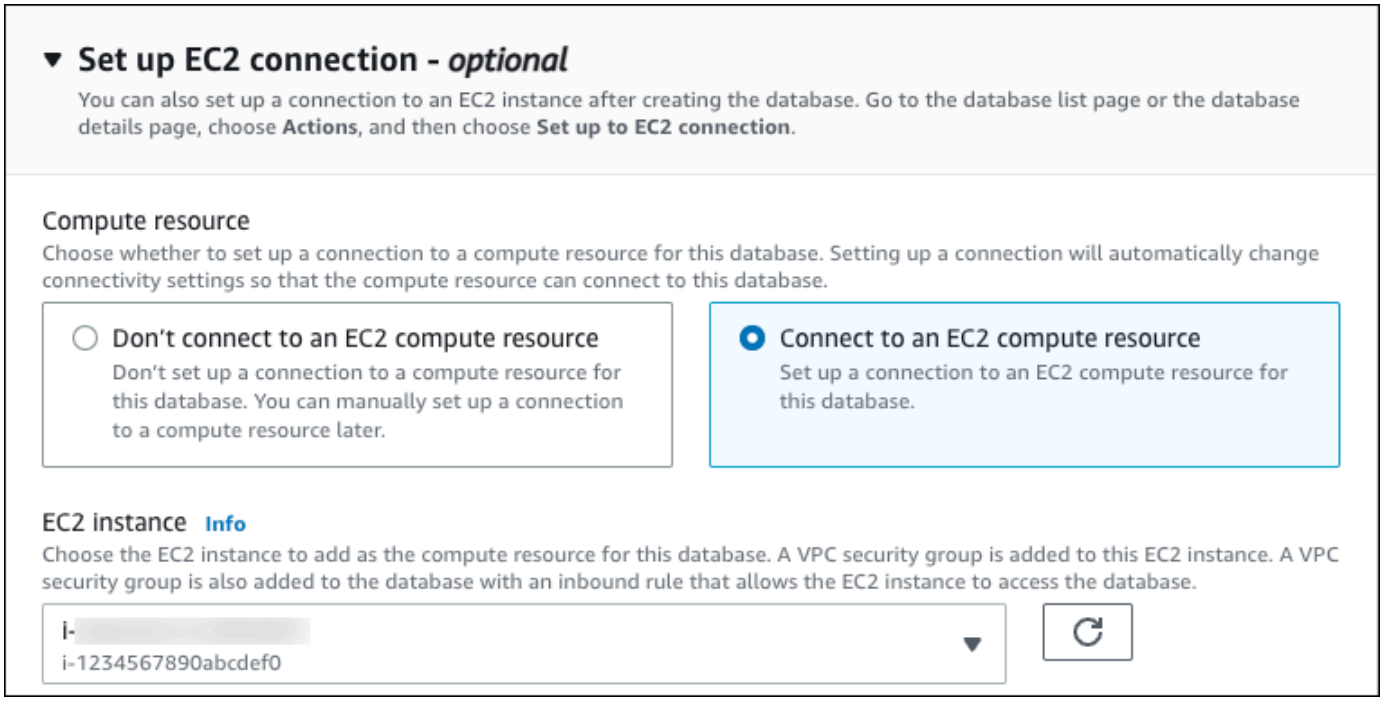


8. Em Nome de usuário principal, insira um nome para o usuário ou mantenha o nome padrão (**`postgres`**).
9. Para usar uma senha primária gerada automaticamente para o cluster de banco de dados, marque a opção Gerar uma senha automaticamente.

   Para inserir sua senha primária, desmarque a opção Gerar uma senha automaticamente, depois insira a mesma senha em Senha primária e em Confirmar senha.

10. Para configurar uma conexão com a instância do EC2 que você criou anteriormente, abra Configurar conexão do EC2: opcional.

    Selecione Conectar-se a um recurso computacional do EC2. Selecione a instância do EC2 que você criou anteriormente.



11. Abra Visualizar as configurações padrão da criação fácil.

View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-postgresql-13	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	13.6	Yes
DB parameter group	default.aurora-postgresql13	Yes
DB cluster parameter group	default.aurora-postgresql13	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Você pode examinar as configurações padrão usadas com a opção Easy Create (Criação fácil). A coluna Editável após a criação do banco de dados mostra quais opções podem ser alteradas após a criação do banco de dados.

- Se uma configuração tiver Não nessa coluna e você quiser uma configuração diferente, poderá usar a opção Criação padrão para criar o cluster de banco de dados.
- Se uma configuração tiver Sim nessa coluna e você quiser uma configuração diferente, poderá usar a opção Criação padrão para criar o cluster de banco de dados, ou modificar o cluster de banco de dados depois de criá-lo para alterar a configuração.

12. Selecione Criar banco de dados.

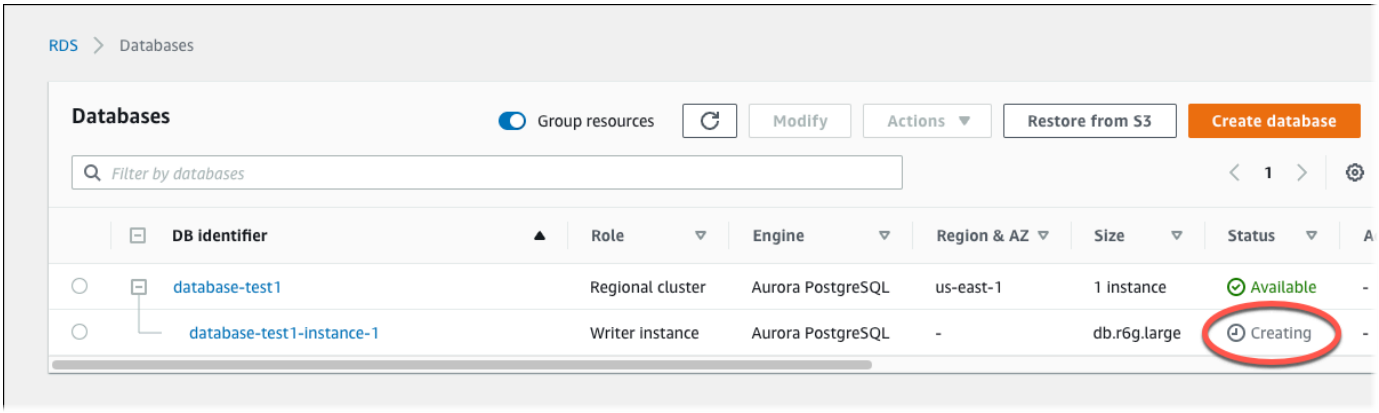
    Para visualizar o nome do usuário principal e a senha do cluster de banco de dados, escolha Visualizar detalhes da credencial.

    Use o nome de usuário e a senha que aparecem para se conectar ao cluster de banco de dados como o usuário principal.

    > **Important**
    >
    > Você não pode visualizar a senha do usuário principal novamente. Caso você não a registre, talvez seja necessário alterá-la.
    > Se for necessário alterar a senha do usuário primário depois que o cluster de banco de dados estiver disponível, será possível modificar o cluster de banco de dados para fazer isso. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

13. Na lista Bancos de dados, selecione o nome do novo cluster de banco de dados do Aurora PostgreSQL para mostrar seus detalhes.

    A instância do gravador tem o status Criando até o cluster de banco de dados estar pronto para uso.



Quando o status da instância do gravador muda para Disponível, você pode se conectar ao cluster de banco de dados. Dependendo da classe da instância de banco de dados e da quantidade de armazenamento, pode levar até 20 minutos para que o novo cluster de banco de dados esteja disponível.

(Opcional) Criar VPC, instância do EC2 e cluster do Aurora PostgreSQL usando o CloudFormation

Em vez de usar o console para criar a VPC, a instância do EC2 e o cluster de banco de dados do Aurora PostgreSQL, você pode usar o CloudFormation para provisionar recursos da AWS tratando a infraestrutura como código. Para ajudar a organizar os recursos da AWS em unidades menores e mais gerenciáveis, você pode usar a funcionalidade de pilha aninhada do CloudFormation. Consulte mais informações em Criar uma pilha no console do CloudFormation e Trabalhar com pilhas aninhadas.

> **Important**
>
> O CloudFormation é gratuito, mas os recursos que o CloudFormation são ativos. Você incorre nas taxas de uso padrão para esses recursos até que os encerre. Para obter mais informações, consulte Definição de preço do Amazon Aurora.

Para criar recursos usando o console do CloudFormation, conclua as seguintes etapas:

- Etapa 1: baixar o modelo do CloudFormation
- Etapa 2: configurar recursos usando o CloudFormation

Baixar o modelo do CloudFormation

Um modelo do CloudFormation é um arquivo de texto JSON ou YAML que contém as informações da configuração dos recursos que você deseja criar na pilha. Esse modelo também cria uma VPC e um bastion host para você com o cluster do Aurora.

Para baixar o arquivo de modelo, abra o link Aurora PostgreSQL CloudFormation template.

Na página do Github, clique no botão Baixar arquivo bruto para salvar o arquivo YAML do modelo.

Configurar recursos usando o CloudFormation

> **Note**
>
> Antes de iniciar esse processo, verifique se você tem um par de chaves para uma instância do EC2 na Conta da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2 e instâncias do Linux.

Ao usar o modelo do CloudFormation, você deve selecionar os parâmetros certos para garantir que os recursos sejam criados corretamente. Siga as etapas abaixo:

1. Faça login no Console de gerenciamento da AWS e abra o console CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Selecione Create Stack (Criar pilha).
3. Na seção Especificar modelo, selecione Fazer upload de um arquivo de modelo do computador e escolha Próximo.
4. Na página Especificar detalhes da pilha, defina os seguintes parâmetros:
   a. Defina o Nome da pilha como AurPostgreSQLTestStack.
   b. Em Parâmetros, defina Zonas de disponibilidade selecionando duas zonas de disponibilidade.
   c. Em Configuração do bastion host do Linux, em Nome da chave, selecione um par de chaves para fazer login na instância do EC2.
   d. Em Configurações do bastion host do Linux, defina o Intervalo de IP permitido para o endereço IP. Para conectar instâncias do EC2 à VPC usando o Secure Shell (SSH), determine o endereço IP público usando o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP é 192.0.2.1/32.

> **Warning**
>
> Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas do EC2 usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias do EC2 usando SSH.

e. Em Configuração geral do banco de dados, defina a Classe da instância de banco de dados como db.t4g.large.

f. Defina o Nome do banco de dados como **`database-test1`**.

g. Em Nome de usuário principal do banco de dados, insira um nome para o usuário principal.

h. Defina a Gerenciar senha de usuário principal do banco de dados com o Secrets Manager como `false` para esse tutorial.

i. Em Senha do banco de dados, defina uma senha de sua escolha. Lembre-se dessa senha para as etapas seguintes do tutorial.

j. Defina Implantação multi-AZ como `false`.

k. Deixe todas as outras configurações com os valores padrão. Clique em Próximo para continuar.

5. Na página Configurar opções de pilha, mantenha todas as opções padrão. Clique em Próximo para continuar.
6. Na página Revisar pilha, selecione Enviar depois de verificar as opções do banco de dados e do bastion host do Linux.

Depois que o processo de criação da pilha for concluído, visualize as pilhas com os nomes BastionStack e APGNS para anotar as informações necessárias para se conectar ao banco de dados. Consulte mais informações em Viewing CloudFormation stack data and resources on the Console de gerenciamento da AWS.

Etapa 3: Conectar-se a um cluster de bancos de dados do Aurora PostgreSQL

Você pode usar qualquer aplicação cliente padrão PostgreSQL para se conectar ao cluster de banco de dados. Neste exemplo, você se conecta a um cluster de banco de dados do Aurora PostgreSQL usando o cliente da linha de comando psql.

Como se conectar ao cluster de banco de dados do Aurora PostgreSQL

1. Encontre o endpoint (nome de DNS) e o número da porta da instância do gravador para seu cluster de banco de dados.

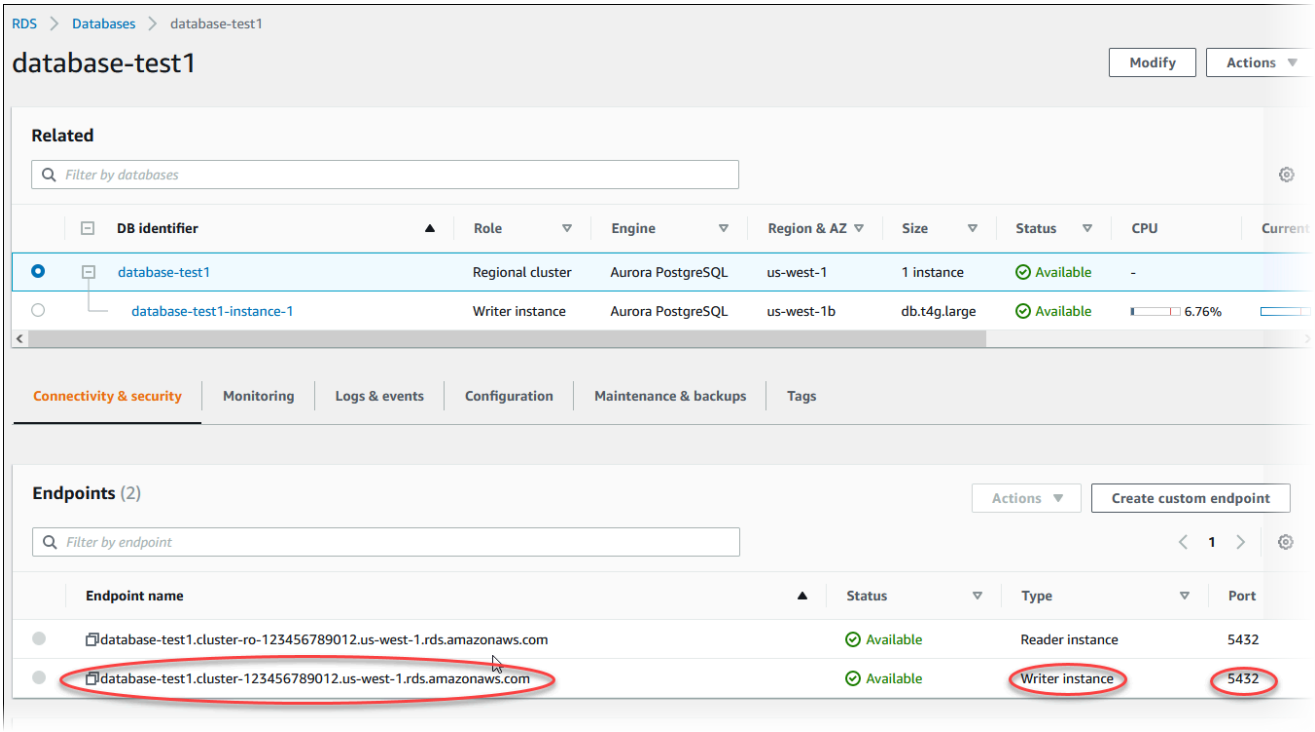
   a. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

   b. No canto superior direito do console do Amazon RDS, escolha a Região da AWS do cluster de banco de dados.

   c. No painel de navegação, escolha Bancos de dados.

   d. Escolha o nome do cluster de banco de dados do Aurora PostgreSQL para exibir os detalhes.

   e. Na guia Conectividade e segurança, copie o endpoint da instância do gravador. Além disso, anote o número da porta. Você precisará do endpoint e do número da porta para conectar-se ao cluster de banco de dados.



2. Conecte-se à instância do EC2 que você criou anteriormente, seguindo as etapas em Conecte-se à sua instância do Linux no Guia do usuário do Amazon EC2.

   Recomendamos que você se conecte à sua instância do EC2 utilizando SSH. Se o utilitário cliente SSH estiver instalado no Windows, Linux ou Mac, você poderá se conectar à instância utilizando o seguinte formato de comando:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Por exemplo, suponha que `ec2-database-connect-key-pair.pem` esteja armazenado em `/dir1` no Linux e que o DNS IPv4 público para sua instância do EC2 seja `ec2-12-345-678-90.compute-1.amazonaws.com`. Seu comando SSH teria a seguinte aparência:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. Obtenha as últimas correções de bugs e atualizações de segurança atualizando o software na instância do EC2. Para fazer isso, use o comando a seguir.

> **Note**
>
> A opção `-y` instala as atualizações sem solicitar confirmação. Para examinar atualizações antes da instalação, omita essa opção.

```
sudo dnf update -y
```

4. Instale o cliente da linha de comando psql do PostgreSQL no Amazon Linux 2023, usando o seguinte comando:

```
sudo dnf install postgresql15
```

5. Conecte-se ao cluster de banco de dados do Aurora PostgreSQL. Por exemplo, insira o comando a seguir. Essa ação permite que você se conecte ao cluster de banco de dados do Aurora PostgreSQL usando o cliente psql.

   Substitua o endpoint da instância do gravador por *`endpoint`*, substitua o nome do banco de dados `--dbname` que você deseja acessar por *`postgres`* e substitua o nome do usuário principal usado por *`postgres`*. Forneça a senha mestra usada quando for solicitada uma senha.

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

   Depois de inserir a senha do usuário, você deverá ver uma saída semelhante à seguinte.

```
psql (14.3, server 14.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

Para ter mais informações sobre a conexão com um cluster de bancos de dados do Aurora PostgreSQL, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL. Se você não conseguir se conectar ao seu cluster de banco de dados, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS.

Por questões de segurança, é uma prática recomendada usar conexões criptografadas. Use uma conexão PostgreSQL não criptografada apenas quando o cliente e o servidor estiverem na mesma VPC e a rede for confiável. Para ter informações sobre como usar conexões criptografadas, consulte . Como proteger dados do Aurora PostgreSQL com SSL/TLS.

6. Executar comandos SQL.

   Por exemplo, o seguinte comando SQL mostra a data e a hora atuais:

```
SELECT CURRENT_TIMESTAMP;
```

Etapa 4: Excluir a instância do EC2 e o cluster de banco de dados

Depois de se conectar e explorar a instância do EC2 e o cluster de banco de dados criados, exclua-os para não receber mais cobranças por eles.

Se você usou o CloudFormation para criar recursos, ignore essa etapa e passe para a próxima.

Como excluir a instância do EC2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, escolha Instâncias.
3. Selecione a instância do EC2 e escolha Estado da instância, Encerrar instância.
4. Quando a confirmação for solicitada, escolha Encerrar.

Consulte mais informações sobre como excluir uma instância do EC2 em Como encerrar uma instância no Guia do usuário do Amazon EC2.

Para excluir um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados) e depois a instância de banco de dados associada ao cluster de banco de dados.
3. Em Actions, selecione Delete.
4. Escolha Delete (Excluir).

   Depois que todas as instâncias de banco de dados associadas a um cluster de banco de dados forem excluídas, o cluster de banco de dados será excluído automaticamente.

(Opcional) Excluir a instância do EC2 e o cluster de banco de dados criado com o CloudFormation

Se você usou o CloudFormation para criar recursos, exclua a pilha do CloudFormation depois de conectar e explorar a amostra de instância do EC2 e de cluster de banco de dados para não receber mais cobranças por eles.

Para excluir os recursos do CloudFormation

1. Abra o CloudFormation Console.
2. Na página Pilhas no console do CloudFormation, selecione a pilha raiz (a pilha sem o nome VPCStack, BastionStack ou APGNS).
3. Escolha Excluir.
4. Selecione Excluir pilha quando a confirmação for solicitada.

Consulte mais informações de como excluir uma pilha no CloudFormation em Deleting a stack on the CloudFormation console no Guia do usuário do AWS CloudFormation.

(Opcional) Conectar o cluster de banco de dados a uma função do Lambda

Você também pode conectar o cluster de banco de dados do Aurora PostgreSQL a um recurso de computação sem servidor do Lambda. As funções do Lambda permitem que você execute código

sem provisionar nem gerenciar a infraestrutura. Uma função do Lambda também permite que você responda automaticamente a solicitações de execução de código em qualquer escala, de dezenas de eventos por dia a centenas por segundo. Para obter mais informações, consulte Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora.

Criar com a configuração expressa

É possível criar e se conectar a um banco de dados sem servidor do Aurora PostgreSQL em questão de segundos usando a configuração expressa, que oferece uma experiência simplificada de criação de banco de dados com padrões pré-configurados para ajudar você a começar de maneira fácil e rápida.

Com dois cliques, é possível ter um cluster do Aurora com uma instância sem servidor pronta para uso em questão de segundos. Também é possível usar a AWS Command Line Interface (AWS CLI) ou SDKs da AWS com o parâmetro `--express-configuration` para criar tanto um cluster quanto uma instância dentro do cluster com uma única chamada de API e prepará-los para a execução de consultas em apenas alguns segundos. Você tem a flexibilidade de modificar a maioria das configurações durante e a maioria das outras configurações após a criação do banco de dados. Para ter mais informações sobre essas configurações, consulte Definições da configuração expressa. Após a criação, você tem acesso à maioria dos recursos e benefícios do Aurora, como a capacidade de adicionar leitores para ter alta disponibilidade e escalabilidade e recursos de failover automático. Algumas opções, como especificar uma chave de criptografia gerenciada pelo cliente, selecionar versões específicas do mecanismo ou associar os clusters a uma nuvem privada virtual (VPC), só estão disponíveis ao usar a opção the section called “Criar com a configuração completa”. Essa experiência de criação para o Aurora PostgreSQL também está disponível com o nível gratuito da AWS.

Tópicos

- Pré-requisitos
- Regiões com suporte
- Gateway de acesso à internet
- Criar um cluster de banco de dados com a configuração expressa
- Definições da configuração expressa
- Conectar-se a clusters com configuração expressa
- Restaurar um cluster criado por meio da configuração expressa
- Limitações

Pré-requisitos

Antes de criar um cluster de banco de dados do Aurora PostgreSQL com a configuração expressa, é necessário ter pelo menos as seguintes permissões:

- `ec2:DescribeAvailabilityZones`
- `iam:CreateServiceLinkedRole`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:EnableInternetAccessGateway`

Regiões com suporte

É possível criar um cluster sem servidor do Aurora PostgreSQL com a configuração expressa em todas as Regiões da AWS, exceto nas regiões da AWS na China, AWS GovCloud (EUA), Oriente Médio (EAU) e Oriente Médio (Bahrein).

Gateway de acesso à internet

O Aurora também oferece um gateway de acesso à internet habilitado por padrão para estabelecer uma conexão segura com os clusters criados por meio da configuração expressa. No momento, é possível usar esse recurso somente com clusters criados com a configuração expressa. Ele é compatível com o protocolo de conexão do PostgreSQL e permite que você se conecte pela internet por meio de uma ampla variedade de ferramentas e plataformas de desenvolvimento executadas fora da sua infraestrutura AWS. Tudo isso sem precisar usar uma VPN ou o AWS Direct Connect. O gateway é distribuído em várias zonas de disponibilidade, oferecendo o mesmo nível de alta disponibilidade do cluster do Aurora. Como ele é um componente gerenciado do Aurora, você não precisa aplicar nenhuma atualização ou patch de software.

O gateway de acesso à internet está integrado ao AWS Identity and Access Management (IAM), o que permite aplicar permissões baseadas em identidade para acesso autorizado e proteger os dados com acesso temporário baseado em tokens. Ao usar o gateway, a configuração necessária para o usuário administrador do banco de dados, inclusive a concessão do perfil rds_iam, é gerenciada automaticamente para você.

Criar um cluster de banco de dados com a configuração expressa

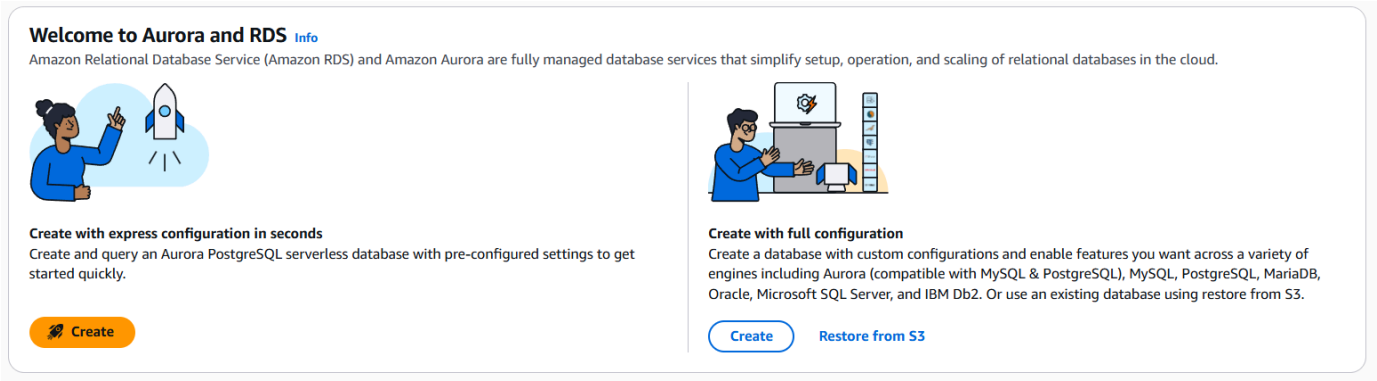
É possível criar um cluster de banco de dados do Aurora PostgreSQL com a configuração expressa usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

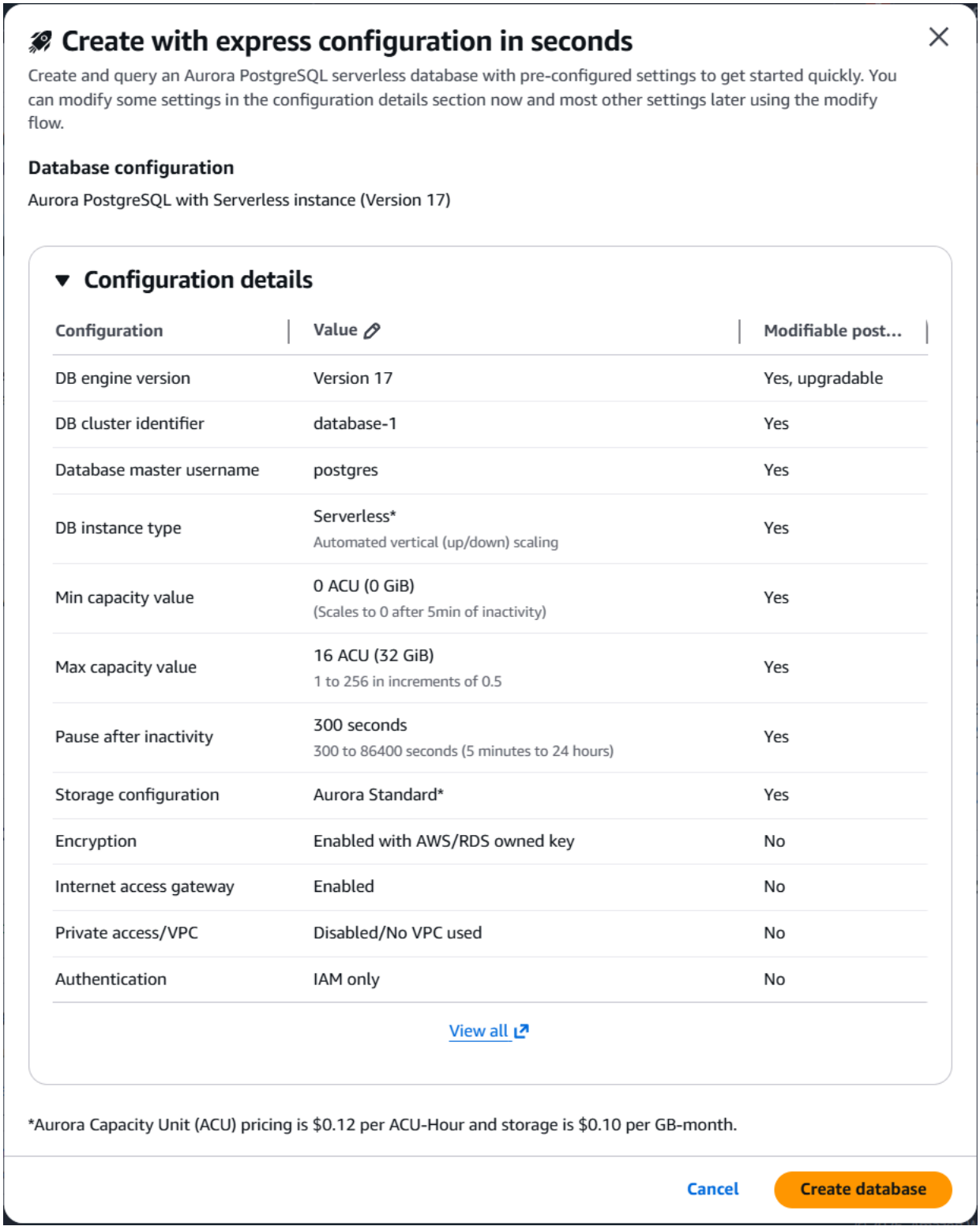
Para criar com a configuração expressa usando o console:

Como criar com a configuração expressa usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região da AWS em que você deseja criar o cluster de banco de dados.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Na página “Boas-vindas ao Aurora e RDS”, localize a seção “Criar com configuração expressa em segundos” no lado esquerdo e escolha “Criar”.

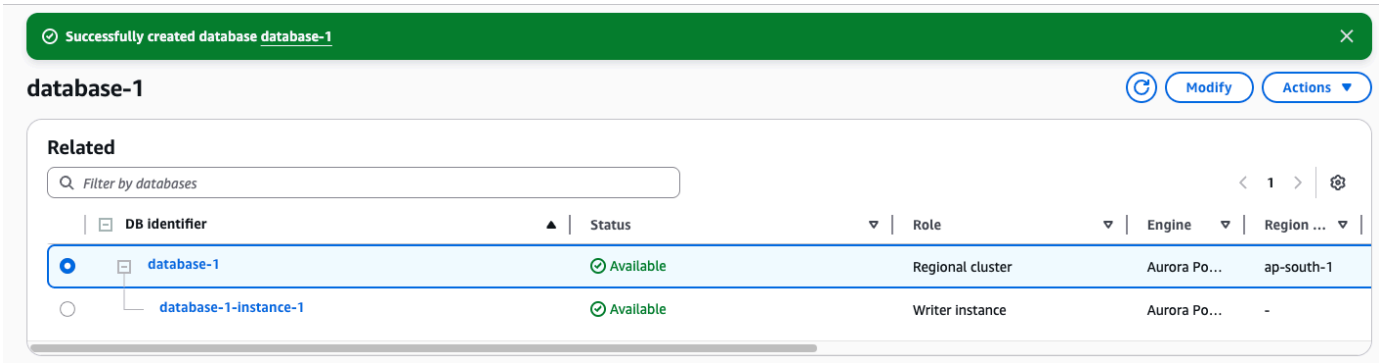


5. Analise as configurações predefinidas na caixa de diálogo “Criar com configuração expressa”.



6. (Opcional) Modifique o identificador do cluster de banco de dados ou o intervalo de capacidade de acordo com sua necessidade.
7. Selecione Criar banco de dados.

8. Seu banco de dados Aurora PostgreSQL Sem Servidor deve estar pronto em segundos. Um banner de êxito confirma a criação e o status do banco de dados muda para “Disponível”.



CLI

É possível usar a AWS CLI para criar clusters do Aurora PostgreSQL com a configuração expressa.

Para Linux ou macOS.

```
aws rds create-db-cluster --db-cluster-identifier sample-express-cluster \
    --engine aurora-postgresql \
    --with-express-configuration
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-express-cluster ^
    --engine aurora-postgresql ^
    --with-express-configuration
```

O parâmetro de configuração expressa cuida da criação de um cluster de banco de dados, uma instância sem servidor do Aurora, da configuração do gateway de acesso à internet e da autenticação do IAM para o usuário administrador do banco de dados (padrão = postgres).

solicitações de

É possível usar o SDK da AWS pertinente e chamar a operação CreateDBCluster para criar um cluster de banco de dados e uma instância de banco de dados com a configuração expressa.

Definições da configuração expressa

A tabela a seguir descreve as definições referentes à configuração expressa e indica se elas podem ser modificadas após a criação.

Configuração	Descrição da configuração	Configuração padrão e limitações
Auto minor version upgrade (Atualização da versão secundária automática)	Esta configuração permite que o cluster de banco de dados do Aurora receba automaticamente as atualizações das versões secundárias escolhidas do mecanismo de banco de dados quando elas forem disponibilizadas. Para ter mais informações sobre as atualizações de mecanismo do Aurora PostgreSQL, consulte Atualizações de mecanismo de banco de dados do Amazon Aurora PostgreSQL.	Habilitada por padrão. É possível alterar após a conclusão da operação de criação.
Chave do AWS KMS	Os clusters com configuração expressa são criptografados com chaves de propriedade da AWS/do serviço RDS.	Habilitada com a chave de propriedade da AWS/do serviço RDS. Não é possível modificar.
Autoridade de certificação	Os clusters com configuração expressa são habilitados com o gateway de acesso à internet, que usa os certificados raiz da AWS. Para ter mais informações, consulte CAs raiz do ACM.	Não aplicável para conexão por meio do gateway de acesso à internet
Configuração de armazenamento do cluster	O tipo de armazenamento do cluster de banco de dados: Aurora Básico ou Aurora Otimizado para E/S. Para ter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.	Aurora Básico por padrão. É possível alterar após a conclusão da operação de criação.
Copiar tags para snapshots	Escolha esta opção para copiar qualquer tag da instância de banco	Desabilitada por padrão. É possível alterar durante ou após a criação.

Configuração	Descrição da configuração	Configuração padrão e limitações
	de dados para um snapshot de banco de dados quando você cria um snapshot. Para ter mais informações, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.	
Autenticação do banco de dados	Os clusters com configuração expressa e gateway de acesso à internet permitem somente a autenticação do IAM. Para ter mais informações, consulte Autenticação de banco de dados do IAM.	Habilitada com a autenticação do IAM por padrão. Não é possível modificar.
Porta do banco de dados	Os clusters com configuração expressa e gateway de acesso à internet aceitam somente a porta padrão do PostgreSQL.	Valor padrão: 5432. Não é possível modificar.

Configuração	Descrição da configuração	Configuração padrão e limitações
Identificador do cluster de banco de dados	Insira um nome para o cluster de banco de dados exclusivo da sua conta na região da AWS escolhida . Esse identificador é usado no endereço do endpoint do cluster para o seu cluster de banco de dados. Para ter informações sobre o endpoint do cluster, consulte Conexões de endpoints do Amazon Aurora. O identificador do cluster de banco de dados tem as seguintes restrições: deve conter de 1 a 63 caracteres alfanuméricos ou hifens. O primeiro caractere deve ser uma letra. Não pode terminar com um hífen ou conter dois hifens consecuti vos. Deve ser exclusivo para todos os clusters de banco de dados por conta da AWS, por região da AWS.	Parâmetro necessário. É possível alterar durante ou após a criação.
Parameter group do cluster de banco de dados	Os clusters com configuração expressa usam o grupo de parâmetro s de cluster de banco de dados padrão do Aurora.	Grupo de parâmetros padrão. É possível alterar após a conclusão da operação de criação.
DB instance class	Os clusters com configuração expressa começam com uma instância do Aurora Sem Servidor.	Instância do Sem Servidor v2. É possível alterar após a conclusão da operação de criação.
Grupo de sub-rede de banco de dados	Os clusters com configuração expressa não são criados dentro de uma VPC. O acesso ocorre por meio do gateway de acesso à internet do Aurora.	Nenhuma VPC associada. Não é possível definir.

Configuração	Descrição da configuração	Configuração padrão e limitações
Enable deletion protection (Habilitar proteção contra exclusão)	Escolha Enable deletion protection (Habilitar proteção contra exclusão) para impedir que seu cluster de banco de dados seja excluído. Por padrão, se você criar um cluster de banco de dados de produção com o console, a proteção contra exclusão será habilitada.	Desabilitada por padrão. É possível alterar durante ou após a conclusão da criação.
Habilitar criptografia	Os clusters com configuração expressa são criptografados com chaves de propriedade da AWS/do serviço RDS.	Habilitada com a chave de proprieda de da AWS/do serviço RDS. Não é possível modificar.
Enable Enhanced Monitoring	Selecione Enable enhanced monitoring (Habilitar monitoramento aprimorado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.	Desabilitada por padrão. É possível alterar após a conclusão da operação de criação.
Habilitar a API de dados do RDS	Selecione Habilitar a API de dados do RDS para habilitar a API de dados do RDS (API de dados). A API de dados fornece um endpoint HTTP seguro para executar declarações SQL sem gerenciar conexões. Para ter mais informações, consulte Usar a API de dados do Amazon RDS.	Desabilitada por padrão. É possível alterar após a conclusão da operação de criação.

Configuração	Descrição da configuração	Configuração padrão e limitações
Tipo de mecanismo	É possível usar clusters com configuração expressa somente com o Aurora PostgreSQL.	Parâmetro necessário. Valor compatível: “aurora-postgresql”. Não é possível modificar.
Versão do mecanismo	Os clusters com configuração expressa usam a versão principal e secundária padrão. Consulte as versões do Aurora para ver detalhes sobre as versões padrão.	Versão principal e secundária padrão. A versão do mecanismo pode ser atualizada posteriormente.
Prioridade de failover	Escolha uma prioridade de failover para a instância. Se você não selecionar um valor, o padrão será tier-1. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Para ter mais informações, consulte Tolerância a falhas para um cluster de bancos de dados Aurora.	Valor padrão = 1. É possível alterar após a conclusão da operação de criação.

Configuração	Descrição da configuração	Configuração padrão e limitações
Nome do banco de dados inicial	Insira um nome para o banco de dados padrão. Se você não fornecer um nome para um cluster de bancos de dados do Aurora PostgreSQL, o Amazon RDS criará um banco de dados chamado postgres. Para o Aurora PostgreSQL, o nome do banco de dados padrão tem estas restrições: deve conter de 1 a 63 caracteres alfanuméricos. Deve começar com uma letra. Os caracteres subsequentes podem ser letras, sublinhados ou dígitos (0 a 9). Não pode ser uma palavra reservada pelo mecanismo de banco de dados. Para criar bancos de dados adicionais, conecte-se ao cluster de banco de dados e use o comando CREATE DATABASE do SQL. Para ter mais informações sobre como se conectar ao cluster de banco de dados, consulte “Como conectar-se a um cluster de banco de dados do Amazon Aurora com o gateway de acesso à internet”.	Valor padrão = postgres. É possível alterar durante ou após a conclusão da criação.

Configuração	Descrição da configuração	Configuração padrão e limitações
Exportações de log	Na seção Log exports (Exportações de log), escolha os logs que deseja começar a publicar no Amazon CloudWatch Logs. Para ter mais informações sobre como publicar logs do Aurora MySQL no CloudWatch Logs, consulte “Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs”. Para ter mais informações sobre como publicar logs do Aurora PostgreSQL no CloudWatch Logs, consulte “Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs”.	Desabilitada por padrão. É possível alterar durante ou após a criação.
Maintenance window	Escolha]”Selecionar janela” e especifique o período semanal durante o qual a manutenção do sistema pode ser realizada. Ou selecione “Sem preferência” para que o Amazon RDS atribua um período aleatoriamente.	A janela de manutenção padrão varia de acordo com a região. É possível alterar durante ou após a criação.
Gerenciar credenciais principais no AWS Secrets Manager	Os clusters com configuração expressa e gateway de acesso à internet permitem somente a autenticação do IAM. Esta configuração não se aplica.	N/D
Senha mestre	Os clusters com configuração expressa e gateway de acesso à internet permitem somente a autenticação do IAM. Esta configuração não se aplica.	N/D

Configuração	Descrição da configuração	Configuração padrão e limitações
Tipo de autenticação do usuário principal	Os clusters com configuração expressa definem automaticamente o tipo de autenticação como autenticação do IAM.	Valor compatível = iam-db-auth. Não é possível modificar.
Master username (Nome de usuário mestre)	Insira um nome a ser usado como o nome do usuário principal para fazer login no cluster de banco de dados. Para o Aurora PostgreSQL, deve conter de 1 a 63 caracteres alfanuméricos. O primeiro caractere deve ser uma letra. O nome não pode ser uma palavra reservada pelo mecanismo de banco de dados. Não é possível alterar o nome do usuário principal após a criação do cluster de banco de dados.	Valor padrão = postgres. É possível alterar durante ou após a criação.
Multi-AZ deployment (Implantação multi-AZ)	Os clusters com configuração expressa não estão associados a uma VPC, e as zonas de disponibilidade são escolhidas automaticamente para você. É possível adicionar réplicas de leitura após a criação do banco de dados.	As zonas de disponibilidade são escolhidas automaticamente. As instâncias de gravador e leitor são colocadas em diferentes zonas de disponibilidade.
Tipo de rede	Os clusters com configuração expressa não estão associados a uma VPC.	Não aplicável

Configuração	Descrição da configuração	Configuração padrão e limitações
Acesso público	Os clusters com configuração expressa são criados com o gateway de acesso à internet, que oferece acesso seguro à internet com integração predefinida com o AWS Identity and Access Management e o AWS Shield.	Não aplicável
Suporte estendido do RDS	Selecione Habilitar Suporte estendido do RDS para permitir que as versões principais do mecanismo compatíveis continuem funcionando após a data de término do suporte padrão do Aurora. Quando você cria um cluster de banco de dados, o Amazon Aurora usa como padrão o Suporte estendido do RDS. Para evitar a criação de um cluster de banco de dados após a data de fim do suporte padrão do Aurora e para evitar cobranças pelo Suporte estendido do RDS, desabilite essa configuração. Os clusters de banco de dados existentes não incorrerão em cobranças até a data de início dos preços do Suporte estendido do RDS. Para ter mais informações, consulte Suporte estendido do Amazon RDS com Amazon Aurora.	Não compatível
RDS Proxy	O RDS Proxy não é compatível com clusters que não estão associados a uma VPC. Para ter mais informações, consulte Amazon RDS Proxy para Aurora.	Não compatível

Configuração	Descrição da configuração	Configuração padrão e limitações
Período de retenção	Escolha o tempo, de 1 a 35 dias, que o Aurora retém cópias de backup do banco de dados. As cópias de backup podem ser usadas para restaurações point-in-time (PITR) do banco de dados, contabilizando até os segundos.	Valor padrão = 1. É possível alterar após a conclusão da operação de criação.
Janela de backup	Uma janela de 30 minutos seleciona da aleatoriamente em um intervalo de tempo de 8 horas para fazer backup automático do banco de dados.	Usa o grupo de parâmetros padrão. É possível alterar posteriormente usando grupos de parâmetros.
Ativar DevOps Guru	Para que o DevOps Guru para RDS ofereça uma análise detalhada de anomalias de desempenho, o Insights de Performance deve estar ativado. Para ter mais informações, consulte Configurar o DevOps Guru para RDS.	Desabilitada por padrão. É possível alterar após a conclusão da operação de criação.
Ativar o Performance Insights	O Insights de Performance do Amazon RDS fornece informaçõ es de monitoramento do banco de dados. Para ter mais informaçõ es, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.	Desabilitada por padrão. É possível alterar após a conclusão da operação de criação.
Nuvem privada virtual (VPC)	Os clusters com configuração expressa não estão associados a uma VPC.	N/D

Configuração	Descrição da configuração	Configuração padrão e limitações
VPC security group (firewall) [Grupo de segurança da VPC (firewall)]	Os clusters com configuração expressa não estão associados a uma VPC.	N/D
Encaminha mento de gravação	Permite encaminhar as gravações enviadas no endpoint do leitor ao nó do gravador para processam ento. Permite obter consistência de leitura após gravação. Para ter mais informações, consulte Encaminha mento de gravação local no Aurora PostgreSQL.	Desabilitada por padrão. É possível alterar após a conclusão da operação de criação.

Conectar-se a clusters com configuração expressa

Os clusters criados com a configuração expressa são configurados automaticamente com um gateway de acesso à internet, que oferece acesso público ao banco de dados com a autenticação do IAM. A identidade do IAM usada na conexão com o banco de dados deve ter permissões para rds-db:connect para gerar o token de autenticação necessário para acessar o banco de dados. Para saber mais, consulte Autenticação de banco de dados do IAM.

O guia a seguir demonstra como usar as várias opções na guia “Conectividade e segurança” no Console de gerenciamento da AWS para estabelecer conexão com o banco de dados.

O console do RDS exibe informações relevantes, como trechos de código, detalhes do endpoint e outros pormenores da conexão, para ajudar você a se conectar ao banco de dados. Ele também oferece um utilitário para gerar um token e acesso direto por meio do AWS CloudShell. Tanto o utilitário de geração de token quanto o CloudShell se conectam usando o nome do usuário principal configurado no momento da criação do banco de dados.

Usar a guia “Conectividade e segurança”

Depois que o banco de dados for criado, acesse a guia “Conectividade e segurança” para acessar as opções de conexão. A seção “Conectar-se ao banco de dados” oferece três métodos de conexão:

Trechos de código

Use ao se conectar por meio de SDK, APIs ou ferramentas de terceiros. O console fornece trechos de código prontos para uso de acordo com a linguagem de programação selecionada. Os trechos também reproduzem dinamicamente a configuração de autenticação do banco de dados.

As etapas a seguir mostram a conexão com o cluster de banco de dados usando o trecho de código do utilitário de linha de comandos psql.

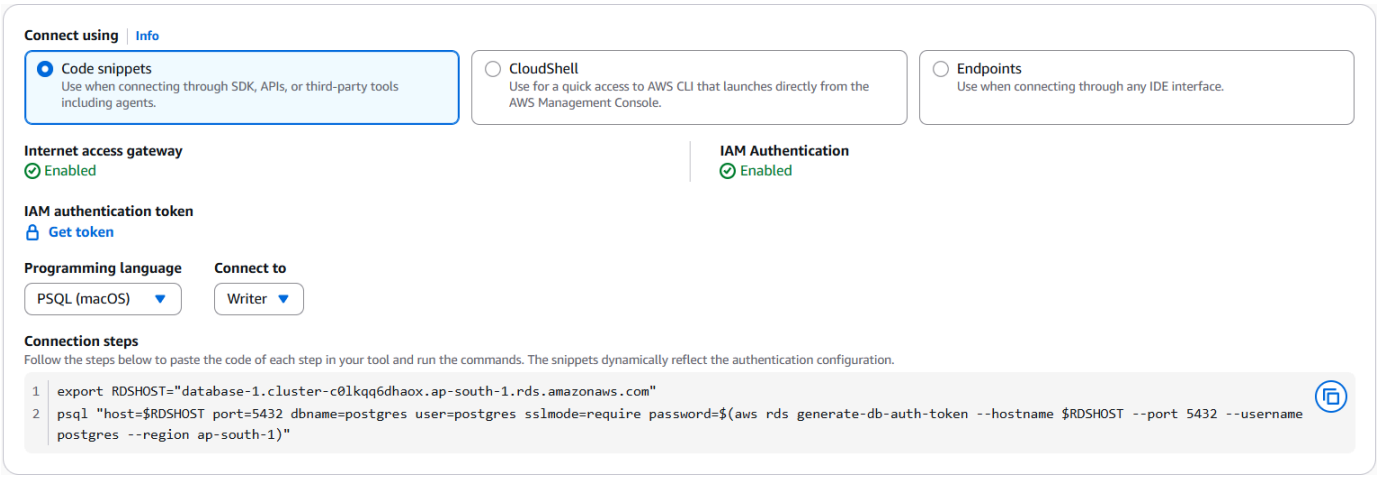
Pré-requisitos

- Você pode se conectar a clusters de bancos de dados do Aurora PostgreSQL usando determinadas ferramentas, como o psql, o terminal interativo do PostgreSQL. Para ter mais informações sobre como usar o terminal interativo do PostgreSQL, consulte “psql” na documentação do PostgreSQL.
- É necessário instalar a versão mais recente da AWS CLI. Para instalar ou atualizar a AWS CLI, consulte Instalar ou atualizar a versão mais recente da AWS CLI.

Conectar-se com o psql

Como se conectar com o psql

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Acesse seu cluster de banco de dados na guia “Lista de bancos de dados”.
3. Na guia “Conectividade e segurança”, selecione o trecho de código PSQL para seu sistema operacional (macOS, Linux ou Windows).
4. Copie o código mostrado no modal das etapas de conexão.



5. Abra um terminal em seu ambiente/máquina de desenvolvimento local.
6. (Opcional) Se já tiver configurado a AWS CLI e as credenciais, você poderá ignorar esta etapa. Do contrário, use o comando a seguir para uma sessão de login temporária.

```
aws login
```

7. Cole o comando que você copiou na Etapa 4. Você deve ver o login do psql e o prompt “postgres =>”, que permite executar comandos SQL.

CloudShell

O console oferece uma maneira automatizada para estabelecer conexão com o cluster usando o AWS CloudShell. O guia a seguir mostra as etapas para estabelecer conexão usando o psql e o CloudShell.

Pré-requisitos

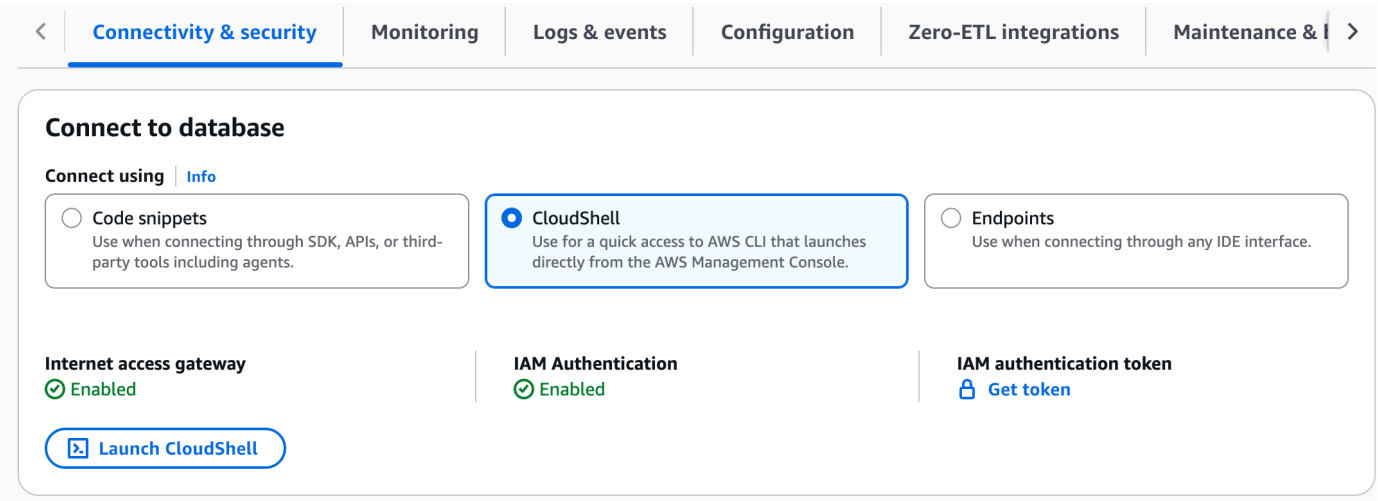
Nenhum

Usar o CloudShell e o psql

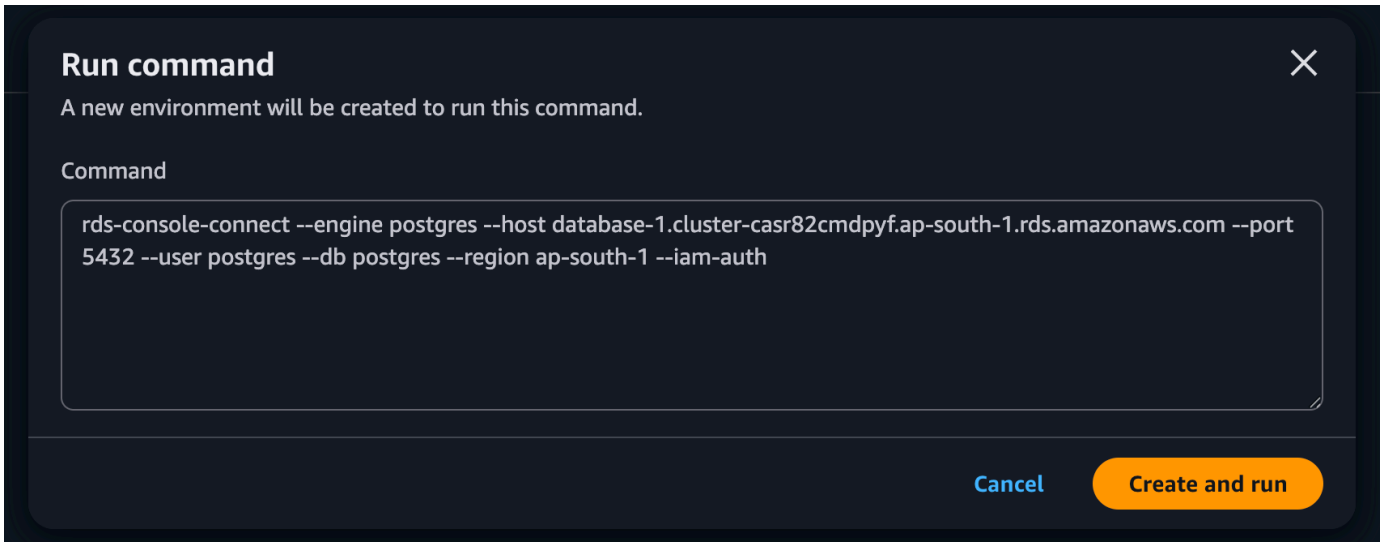
Como se conectar usando o CloudShell e o psql

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Acesse seu cluster de banco de dados na guia “Lista de bancos de dados”.
3. Na guia “Conectividade e segurança”, selecione CloudShell.

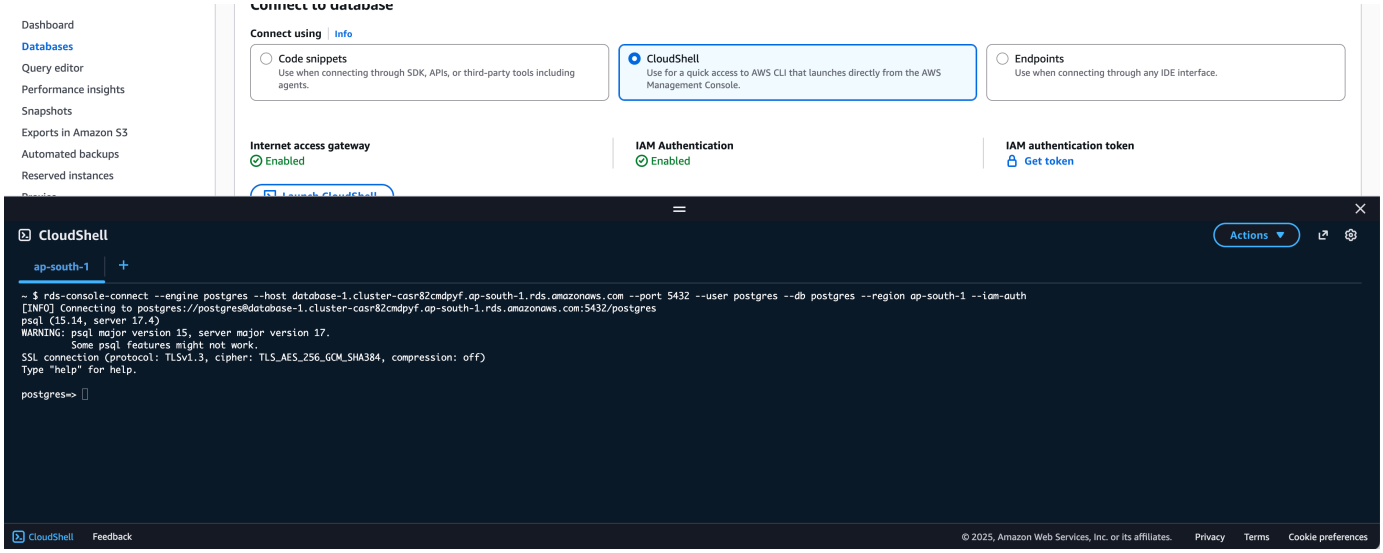
4. Clique em “Iniciar CloudShell”.



5. Clique em “Executar” na janela do prompt. O comando é pré-preenchido com informações relevantes para você se conectar ao seu cluster específico.



6. Você deve ver o login do psql e o prompt “postgres =>”, que permite executar comandos SQL.



Use para acesso rápido à AWS CLI, que é iniciada diretamente no Console de gerenciamento da AWS. O CloudShell oferece um ambiente de shell baseado em navegador com acesso pré-autenticado à AWS CLI.

Endpoints com o utilitário Obter token

Se estiver usando ferramentas que permitem usar apenas credenciais de nome de usuário e senha, poderá usar o utilitário Obter token para se conectar. Nesse caso, use o token gerado pelo utilitário no campo de senha. O token é gerado para o nome de usuário principal configurado no momento da criação do banco de dados. O token é válido por 15 minutos por vez. Se a ferramenta que você está usando encerrar a conexão, será necessário gerar o token novamente.

As etapas a seguir mostram como usar o utilitário Obter token para se conectar ao cluster de banco de dados usando pgAdmin.

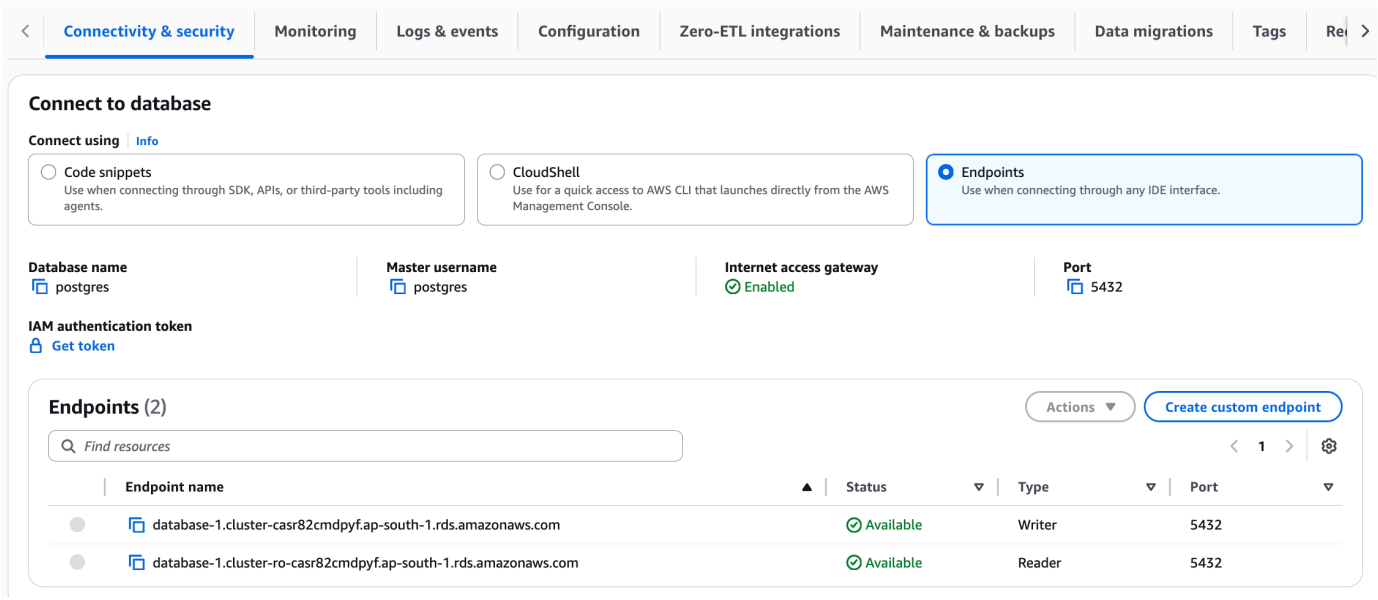
Pré-requisitos

- Você pode usar o utilitário pgAdmin para se conectar a clusters de banco de dados do Aurora PostgreSQL usando uma interface de usuário. Para obter mais informações, consulte a página Download do site pgAdmin.

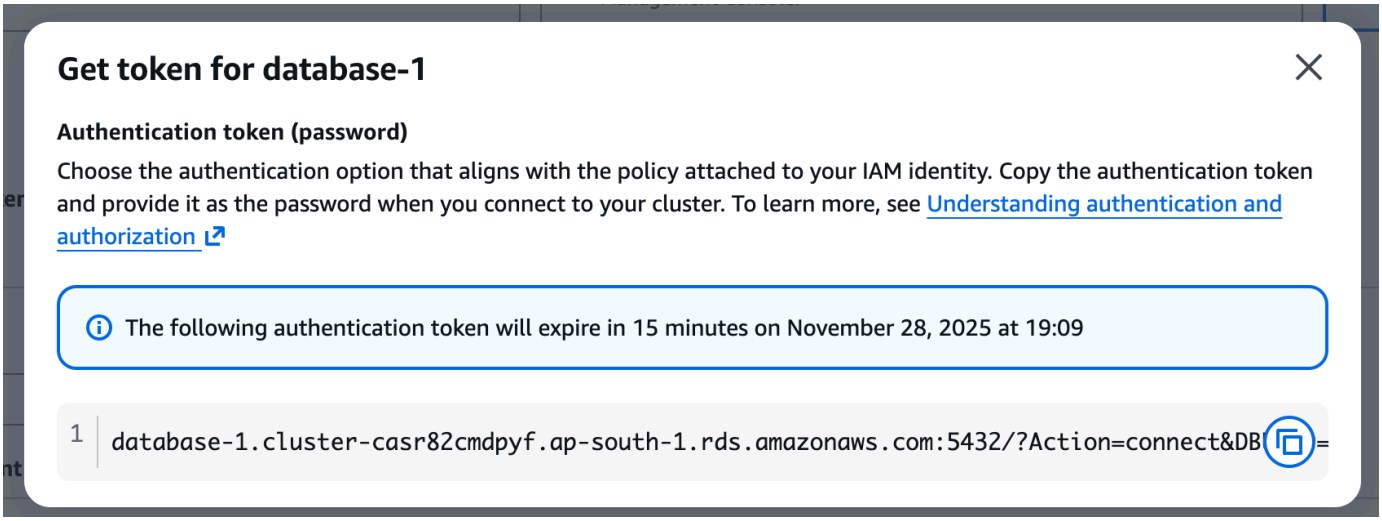
Conectar-se ao pgAdmin

Como se conectar usando o pgAdmin

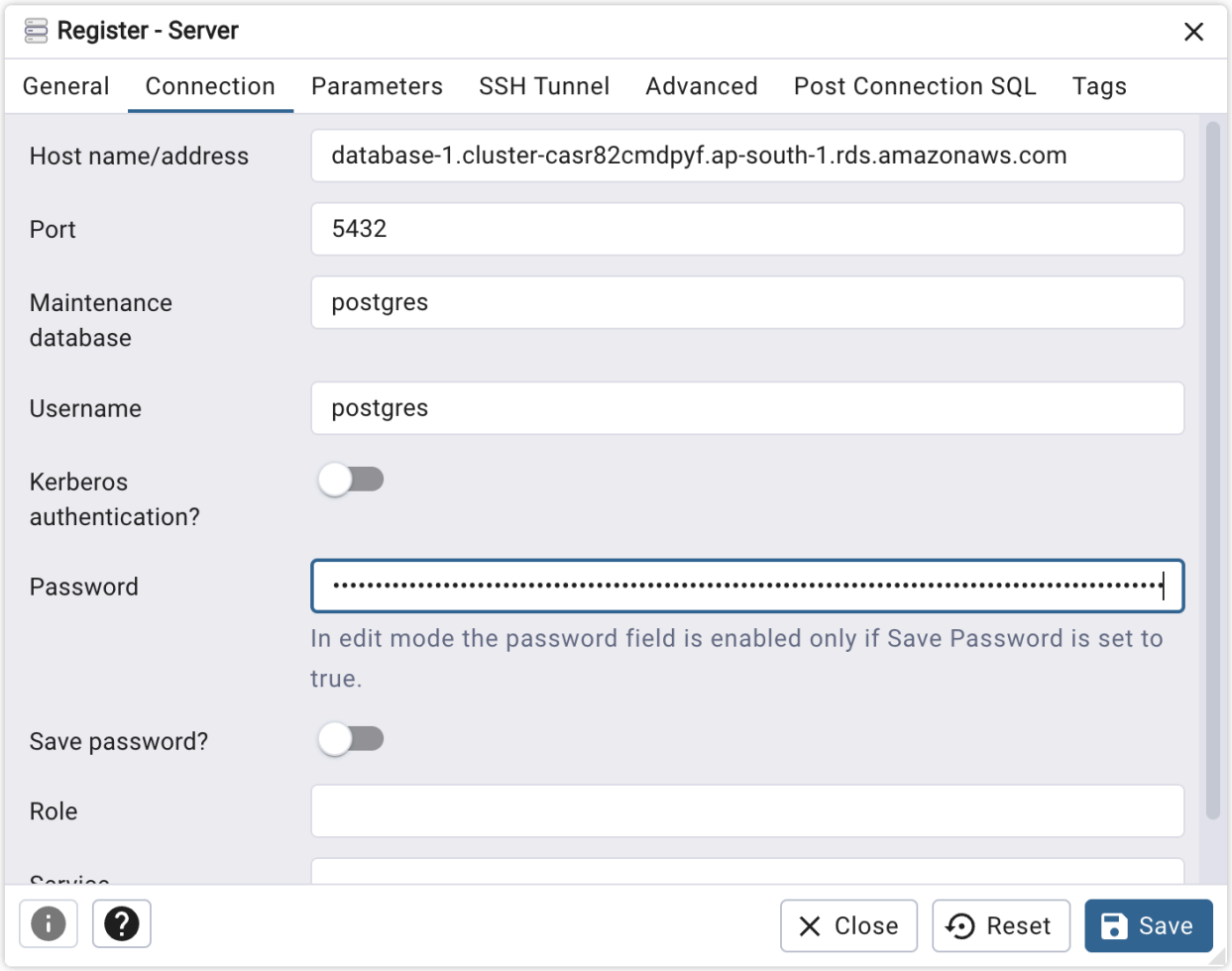
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Acesse seu cluster de banco de dados na guia “Lista de bancos de dados”.
3. Na guia “Conectividade e segurança”, selecione a guia “Endpoints”.



4. Copie as informações relevantes, como endpoint do gravador, porta, nome de usuário principal e nome do banco de dados.
5. Gere e copie um novo token (senha de curta duração) do utilitário Obter token.



6. Abra o pgAdmin em seu computador e “adicione um novo servidor”. Insira os detalhes na guia “Conexão”. Use o token no campo de senha.



O painel de conexão exibe: status do gateway de acesso à internet (habilitado), status da autenticação do IAM (habilitado) e opção de token de autenticação do IAM. Selecione sua linguagem de programação no menu suspenso e siga as etapas de conexão para colar o código e executar os comandos.

Restaurar um cluster criado por meio da configuração expressa

O processo de restauração de clusters criados com a configuração expressa segue o mesmo fluxo de trabalho dos clusters padrão. É possível restaurar por meio de um snapshot manual usando a operação restore-db-cluster-from-snapshot ou restaurar para um ponto específico no tempo usando a operação restore-db-cluster-to-point-in-time. Você pode restaurar um cluster com configuração expressa para um cluster com configuração completa ou um cluster com configuração expressa. Se quiser restaurar para um cluster com configuração expressa, ambas as operações de restauração exigem dois parâmetros adicionais: VPCNetworkingEnabled deve ser definido como false e InternetAccessGatewayEnabled deve ser definido como true. Se você estiver restaurando um

cluster comum que usou a autenticação de nome de usuário principal e senha em um cluster com configuração expressa, precisará modificar o cluster de banco de dados restaurado para atualizar MasterUserAuthenticationType para iam-db-auth.

Limitações

As limitações a seguir se aplicam a clusters criados com a configuração expressa e com o gateway de acesso à internet habilitado por padrão.

- É possível usar a configuração expressa somente com o Aurora PostgreSQL.
- Os clusters criados com a configuração expressa são criptografados com a chave gerenciada da AWS/do serviço RDS. Não é possível usar uma chave do AWS KMS personalizada para criptografar o banco de dados.
- Não é possível associar clusters expressos à Amazon Virtual Private Cloud (Amazon VPC). Ao se conectar ao banco de dados por meio de uma máquina que esteja dentro de uma VPC e em que o cluster expresso tenha o gateway de acesso à internet habilitado, a máquina deverá permitir tráfego de entrada e saída da internet.
- Não é possível desabilitar o gateway de acesso à internet para clusters criados com a configuração expressa.
- É possível usar a autenticação do IAM para o RDS somente com o gateway de acesso à internet. Consulte Autenticação de banco de dados do IAM para ver a lista de limitações e recomendações. Recursos relacionados a outras formas de autenticação, como credenciais gerenciadas no AWS Secrets Manager e autenticação Kerberos, não podem ser usados com clusters que tenham o gateway de acesso à internet habilitado. Também não é possível desabilitar a autenticação do IAM.

  > **Warning**
  >
  > Se você desabilitar o perfil `rds_iam` do nome de usuário principal, perderá o acesso a novas conexões com o banco de dados. É possível restaurar o acesso modificando o cluster e configurando `--master-user-authentication-type` como `iam-db-auth`. Consulte modify-db-cluster.

- Não é possível selecionar uma versão específica do mecanismo com a configuração expressa. A versão do mecanismo pode ser atualizada, mas não revertida para uma versão anterior.

- Os clusters com configuração expressa permitem criar somente instâncias do Aurora Sem Servidor v2. É possível modificar o tipo de instância ou adicionar instâncias após a criação do banco de dados.
- Os clusters com configuração expressa aceitam somente o grupo de parâmetros padrão. O grupo de parâmetros pode ser atualizado após a criação do banco de dados.
- Não é possível usar os recursos a seguir com clusters criados com a configuração expressa, pois eles não estão associados a uma VPC.
  - Aurora Limitless Database
  - Banco de dados global Aurora.
  - RDS Proxy
  - Integrações ETL zero do Aurora
  - Editor de Consultas do RDS
  - Implantações azul/verde
  - Fluxos de atividades do banco de dados
  - Aplicação de patches com tempo de inatividade zero
  - Babelfish
- A API de dados pode ser habilitada para um cluster com configuração expressa após a criação usando a operação ModifyDBCluster. No entanto, não é possível usar a autenticação com nome de usuário principal/senha. É necessário criar outras credenciais de usuário para acessar a API de dados.
- O modo Database Insights Avançado pode ser habilitado em um cluster com configuração expressa após a criação usando a operação ModifyDBCluster.
- Os clusters com configuração expressa aceitam somente endereços IPv4. O IPv6 não é compatível.
- Os clusters com configuração expressa não permitem alterar todas as configurações no momento das operações de criação. Por exemplo, somente é possível escolher o armazenamento padrão do Aurora ao criar com a configuração expressa. Não é possível alterar o tipo de armazenamento depois que o cluster é criado. Consulte “Definições da configuração expressa” para ver uma lista de todas as outras definições e valores padrão que se aplicam à configuração expressa.
- Os clusters criados por meio da configuração expressa com o gateway de acesso à internet habilitado não aceitam suítes de cifras configuráveis. Embora o Amazon RDS para PostgreSQL permita configurar o parâmetro ssl_ciphers para restringir as suítes de cifras SSL permitidas,

esse parâmetro é ignorado em clusters com gateway de acesso à internet habilitado, que usam a configuração padrão da suíte de cifras.

Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora

Este tutorial mostra como instalar um servidor web Apache com PHP e criar um banco de dados MariaDB, MySQL ou PostgreSQL. O servidor web é executado em uma instância do Amazon EC2 usando o Amazon Linux 2023, e você pode escolher entre um cluster de banco de dados do Aurora MySQL ou do Aurora PostgreSQL. Tanto a instância do Amazon EC2 quanto o cluster de banco de dados são executadas em uma virtual private cloud (VPC) com base no serviço da Amazon VPC.

Important

Não há cobrança para criar uma conta da AWS. No entanto, ao concluir este tutorial, é possível gerar custos para os recursos da AWS que você usa. Se esses recursos não forem mais necessários após a conclusão do tutorial, você poderá excluí-los.

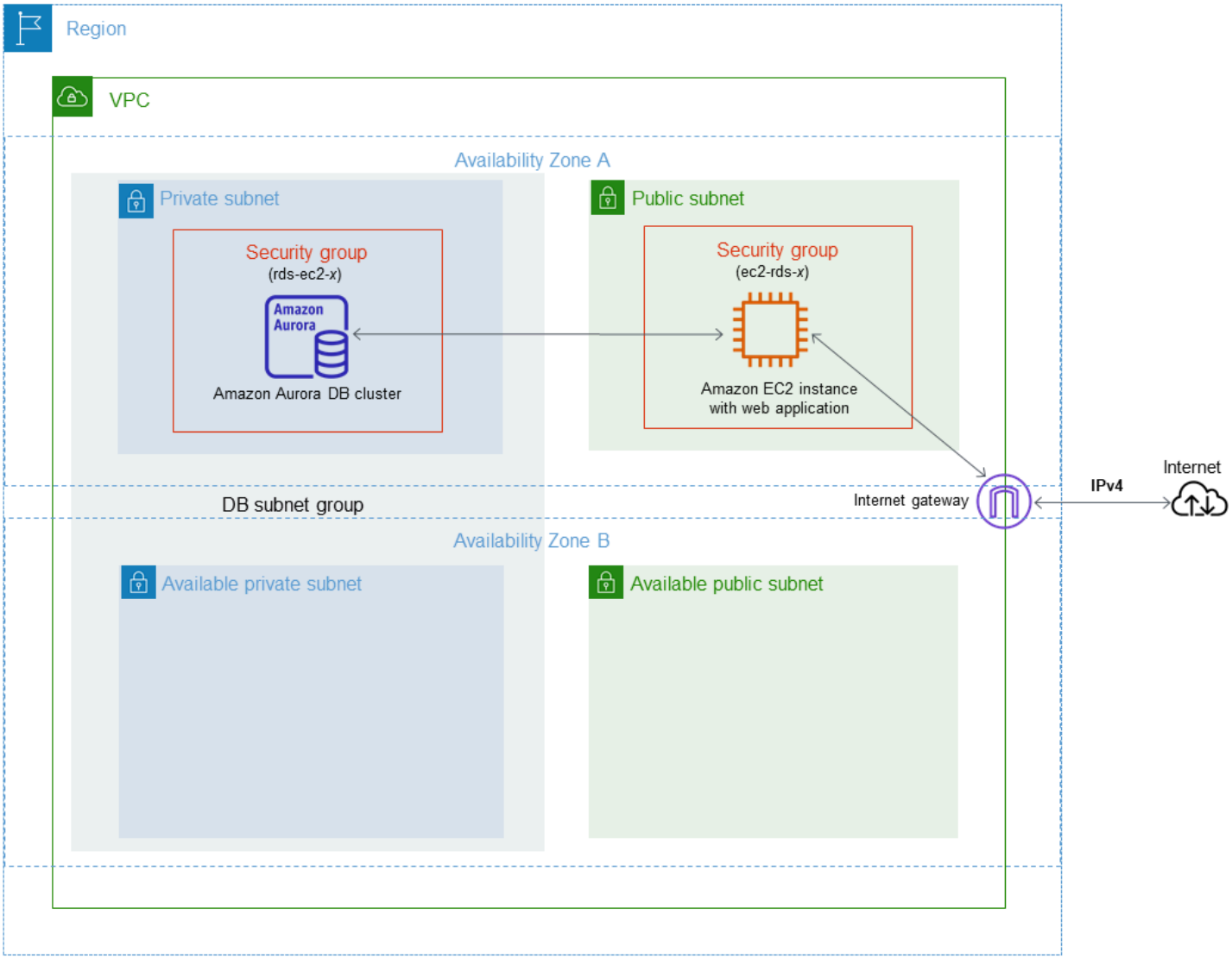
Note

Este tutorial funciona com Amazon Linux 2023 e pode não funcionar para outras versões do Linux.

No tutorial a seguir, crie uma instância do EC2 que usa a VPC, as sub-redes e o grupo de segurança padrão para a sua Conta da AWS. Este tutorial mostra como criar o cluster de banco de dados e configurar automaticamente a conectividade com a instância do EC2 que você criou. Depois, o tutorial mostra como instalar o servidor Web na instância do EC2. Conecte o servidor Web ao cluster de banco de dados na VPC usando o endpoint do gravador de cluster de banco de dados.

1. Iniciar uma instância do EC2 para se conectar ao cluster de banco de dados
2. Criar um cluster de banco de dados do Amazon Aurora
3. Instalar um servidor Web na instância do EC2

O diagrama a seguir mostrará a configuração quando o tutorial estiver completo.



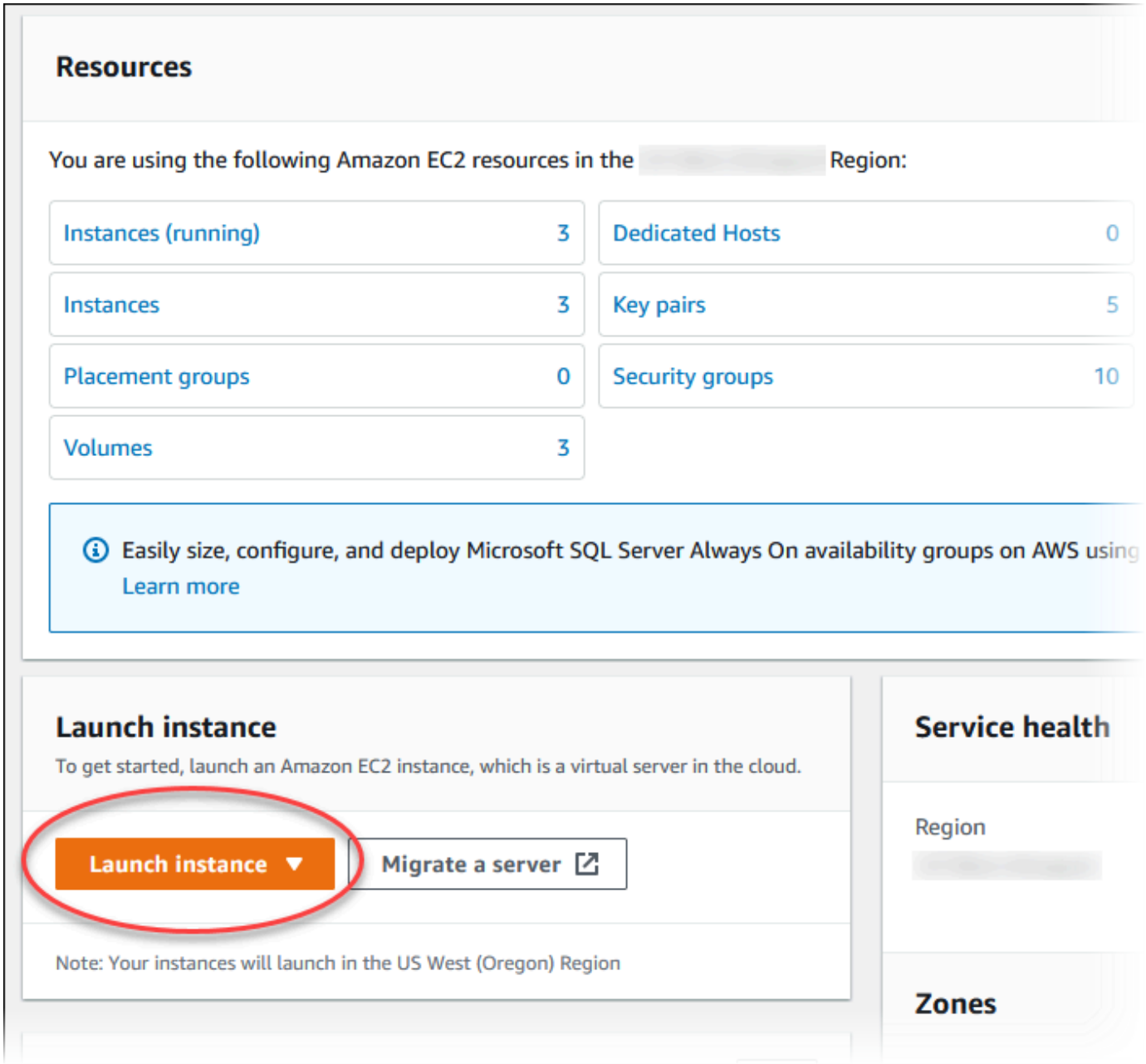
> **Note**
>
> Depois de concluir o tutorial, haverá uma sub-rede pública e privada em cada zona de disponibilidade na sua VPC. Este tutorial usa a VPC padrão para sua Conta da AWS e configura automaticamente a conectividade entre sua instância do EC2 e o cluster de banco de dados. Se você preferir configurar uma nova VPC para esse cenário, conclua as tarefas em Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

Iniciar uma instância do EC2 para se conectar ao cluster de banco de dados

Crie uma instância do Amazon EC2 na sub-rede pública da VPC.

Para iniciar uma instância do EC2

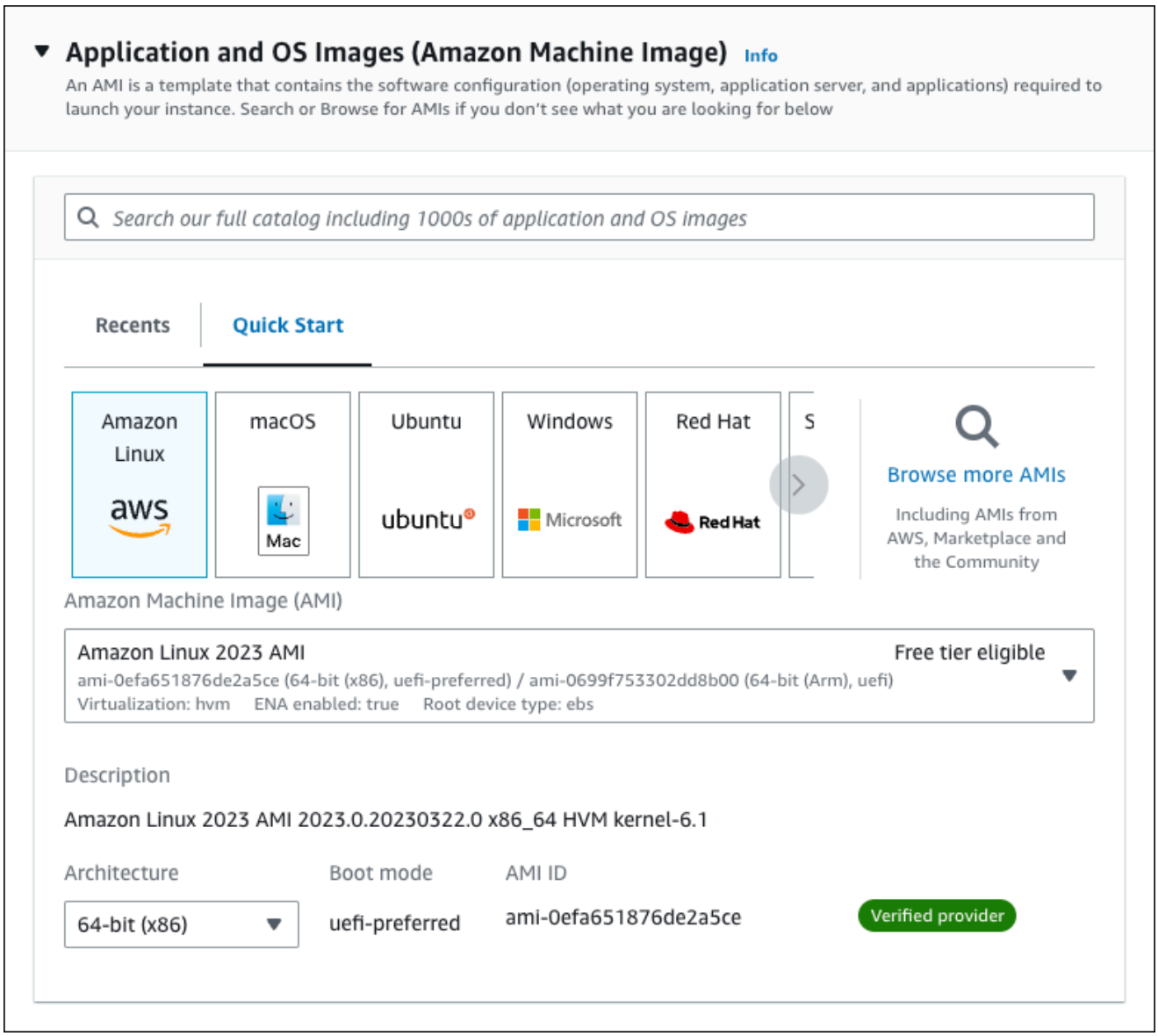
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a Região da AWS em que você deseja criar a instância do EC2.
3. Escolha Painel EC2 e Executar instância, conforme mostrado a seguir.



4. Escolha as configurações a seguir na página Iniciar uma instância.

   a. Em Name and tags (Nome e etiquetas), em Name (Nome), insira **`tutorial-ec2-instance-web-server`**.

   b. Em Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon), selecione Amazon Linux e, depois, AMI do Amazon Linux 2023. Mantenha os padrões nas outras opções.



c. Em Instance type (Tipo de instância), escolha t2.micro.

d. Em Key pair (login) (Par de chaves (login)), escolha um Key pair name (Nome do par de chaves) para usar um par de chaves existente. Para criar um par de chaves para a instância do Amazon EC2, escolha Create new key pair (Criar um novo key pair), depois use a janela Create key pair (Criar par de chaves) para criá-lo.

   Consulte mais informações sobre como criar um par de chaves em Criar um par de chaves no Guia do usuário do Amazon EC2.

e. Em Network settings (Configurações de rede), defina esses valores e mantenha os outros valores como padrão:

- Para Allow SSH traffic from (Permitir tráfego SSH de), escolha a origem das conexões SSH com a instância do EC2.

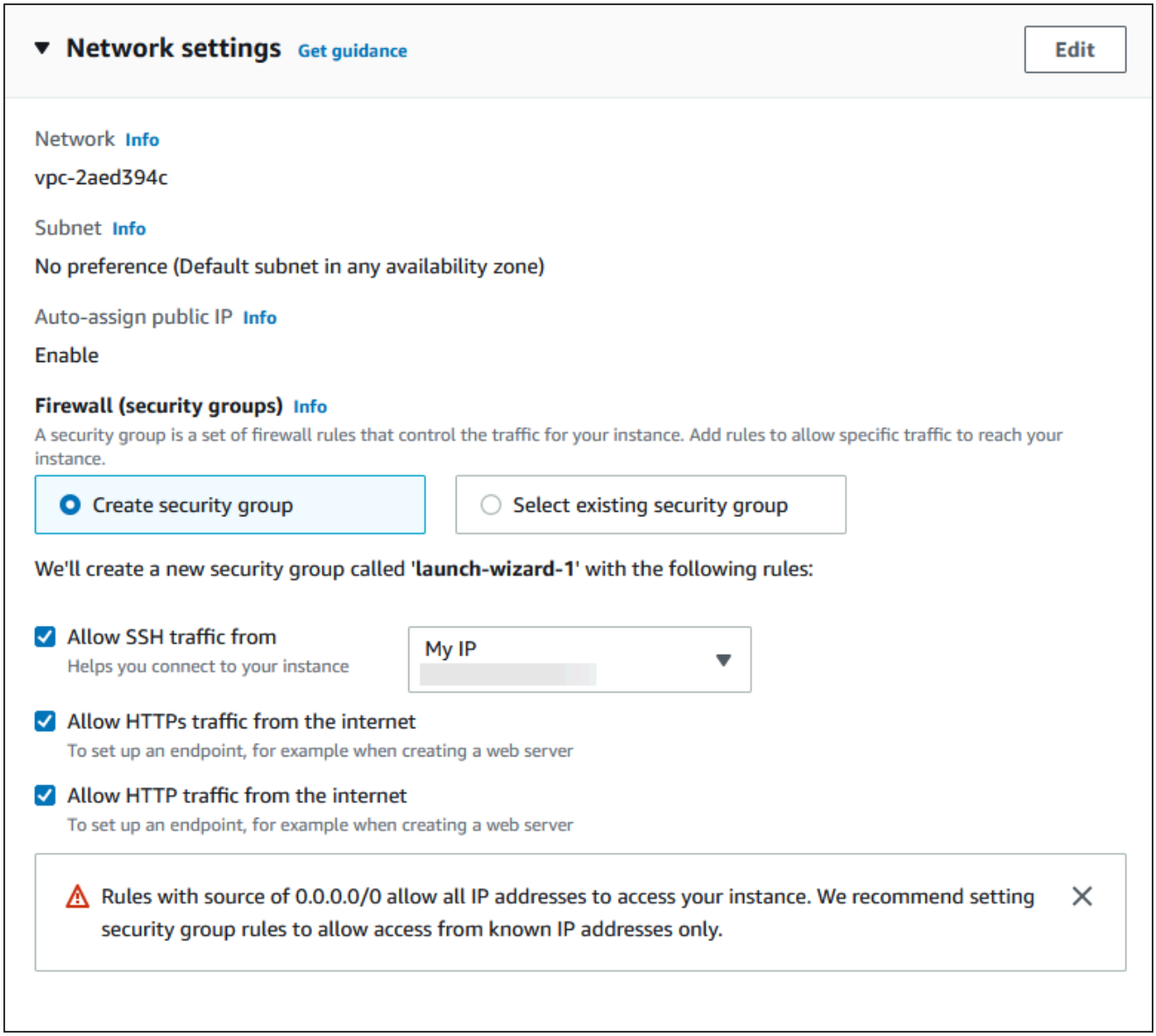
  Você pode escolher My IP (Meu IP) se o endereço IP exibido estiver correto para conexões SSH.

  Caso contrário, determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, é possível usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP: `203.0.113.25/32`.

  Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Em caso afirmativo, determine o intervalo de endereços IP utilizado por computadores cliente.

  > **Warning**
  >
  > Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias usando SSH.

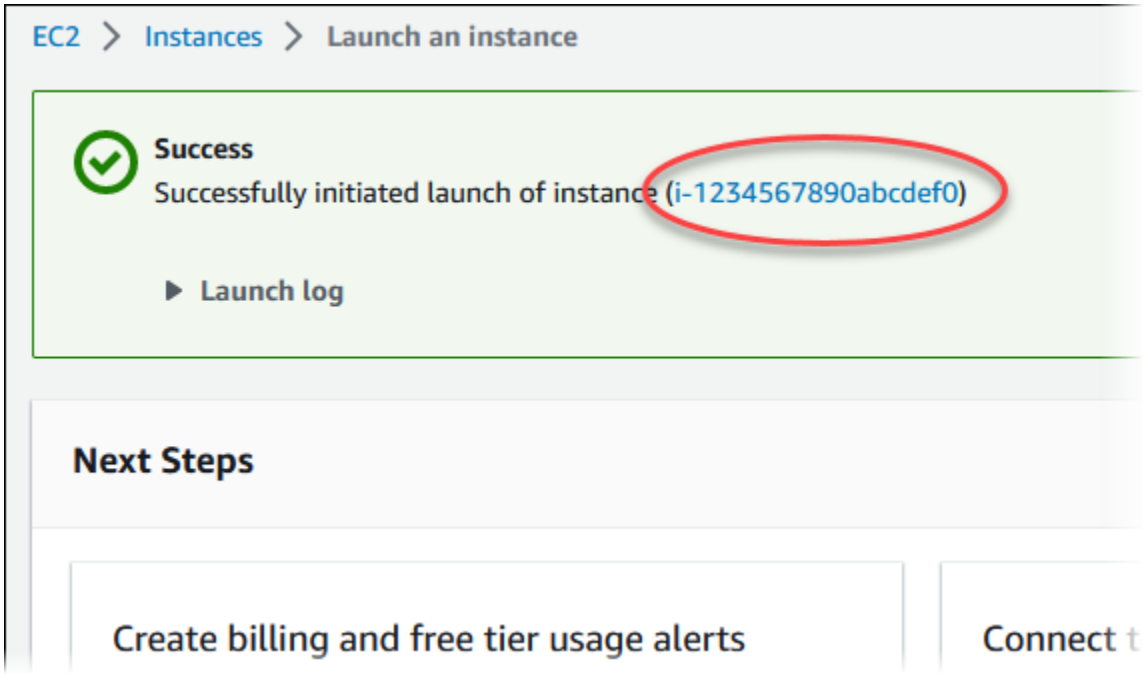
- Ative Allow HTTPs traffic from the internet (Permitir tráfego HTTPs da Internet).
- Ative Allow HTTP traffic from the internet (Permitir tráfego HTTP da Internet).



f. Mantenha os valores padrão para as seções restantes.

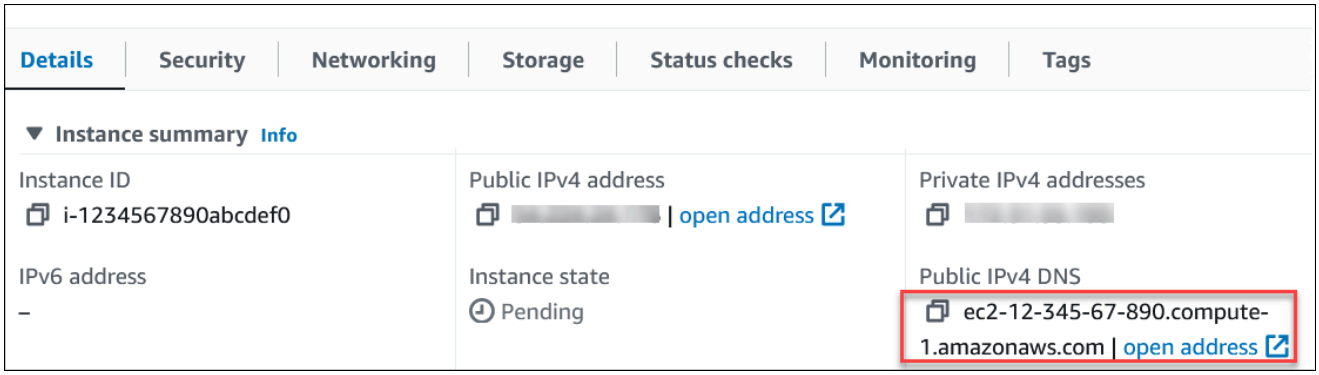
g. Revise um resumo da configuração da instância no painel Summary (Resumo) e, quando você estiver pronto, escolha Launch instance (Iniciar instância).

5. Na página Status de inicialização, anote o identificador de sua nova instância do EC2, por exemplo: `i-1234567890abcdef0`.

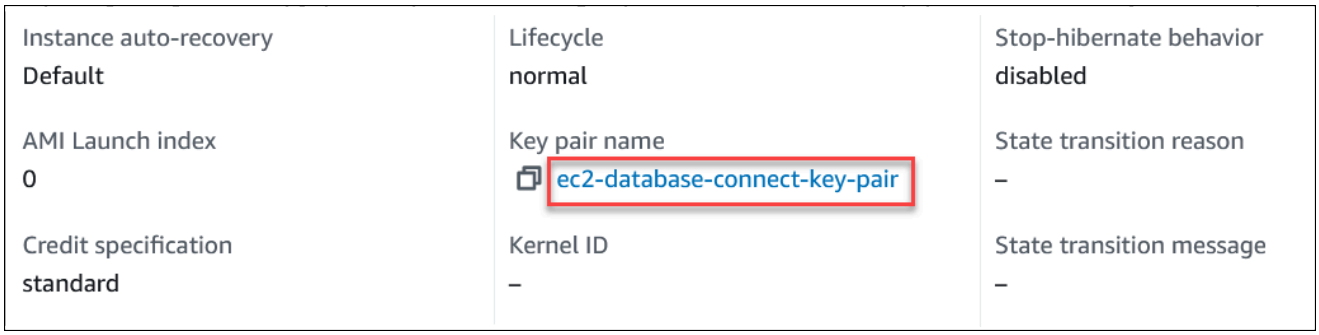


6. Selecione o identificador de instância do EC2 para abrir a lista de instâncias do EC2 e, depois, selecione sua instância do EC2.
7. Na guia Detalhes, observe os seguintes valores, necessários ao se conectar utilizando SSH:

   a. No Resumo da instância, observe o valor do DNS IPv4 público.



   b. Em Detalhes da instância, observe o valor do nome do par de chaves.



8. Aguarde até Instance state (Estado da instância) exibir Running (Em execução) para a instância antes de continuar.
9. Completa Criar um cluster de banco de dados do Amazon Aurora.

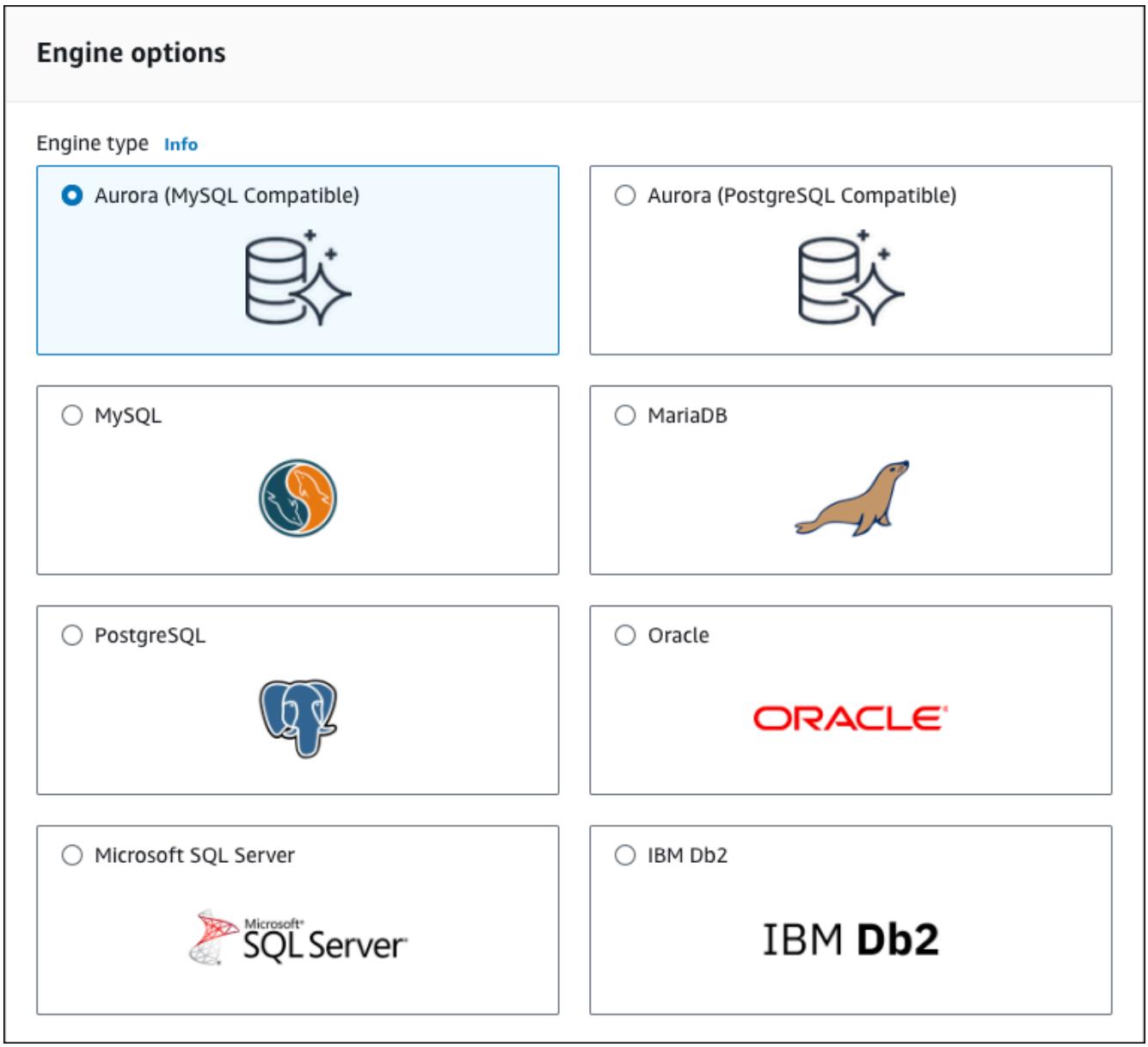
Criar um cluster de banco de dados do Amazon Aurora

Crie um cluster de banco de dados do Amazon Aurora MySQL ou do Aurora PostgreSQL que mantenha os dados usados por uma aplicação web.

Aurora MySQL

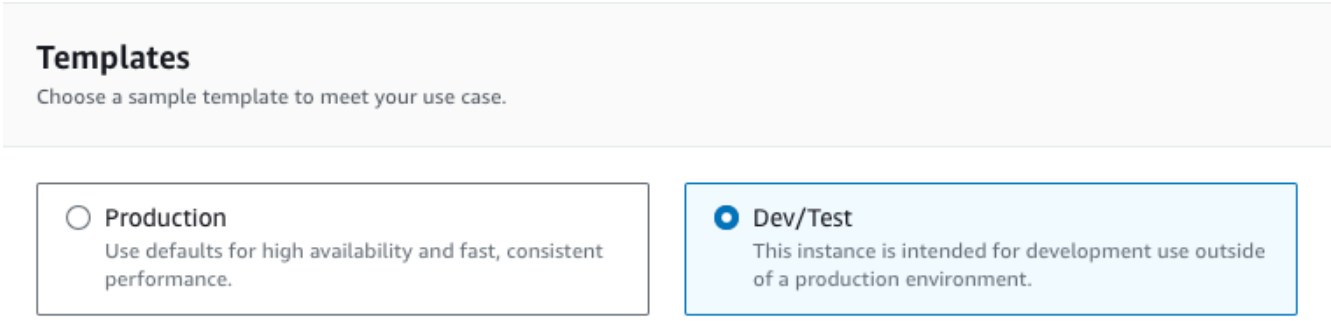
Para criar um cluster de banco de dados Aurora MySQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, verifique se o Região da AWS é a mesma em que você criou sua instância do EC2.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Selecione Criar banco de dados.
5. Na página Criar banco de dados, escolha Criação padrão.
6. Em Opções de mecanismo, selecione Aurora (compatível com MySQL).



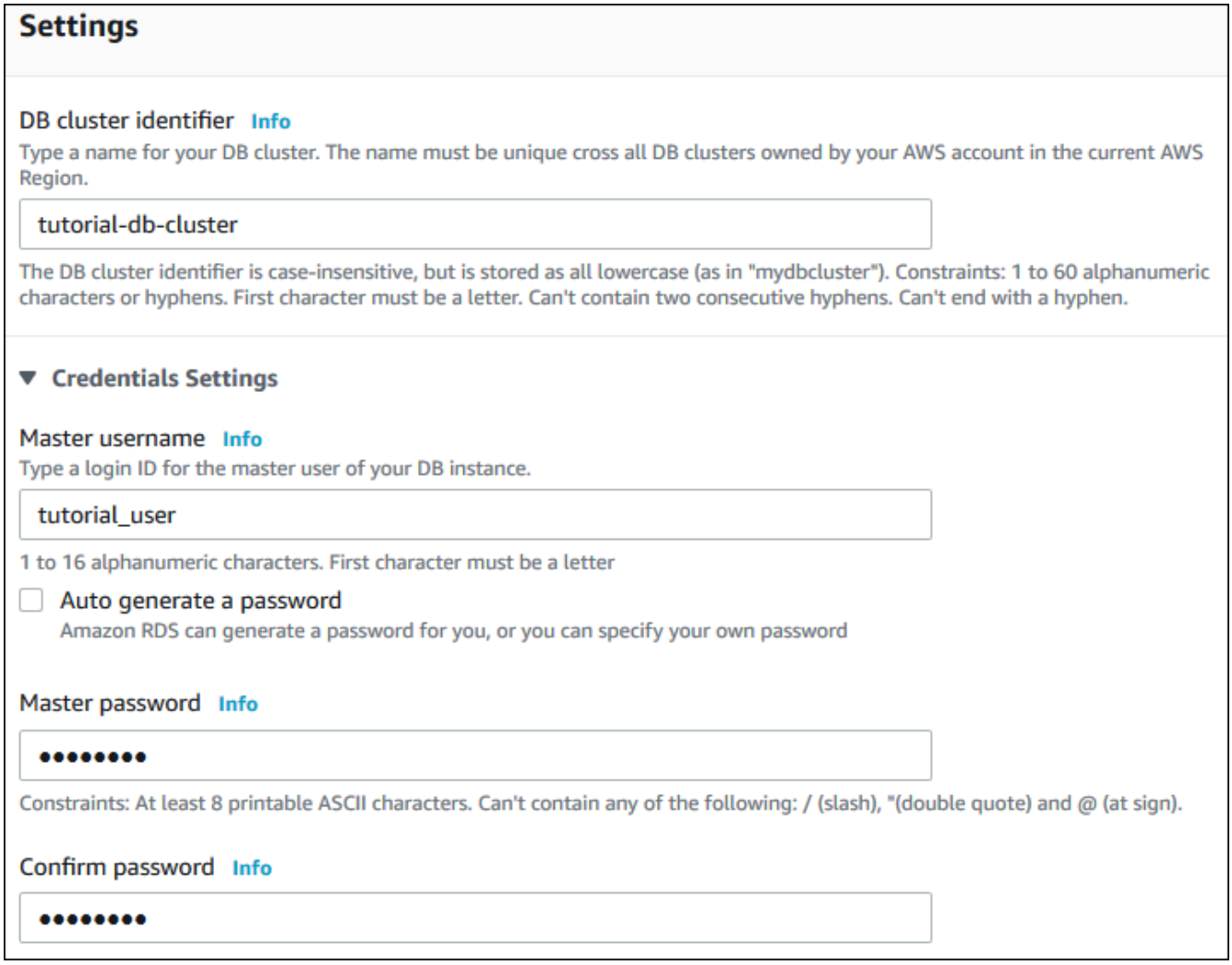
Mantenha os valores padrão para Versão e as outras opções do mecanismo.

7. Na seção Templates (Modelos), escolha Dev/Test.



8. Na seção Settings (Configurações), defina estes valores:

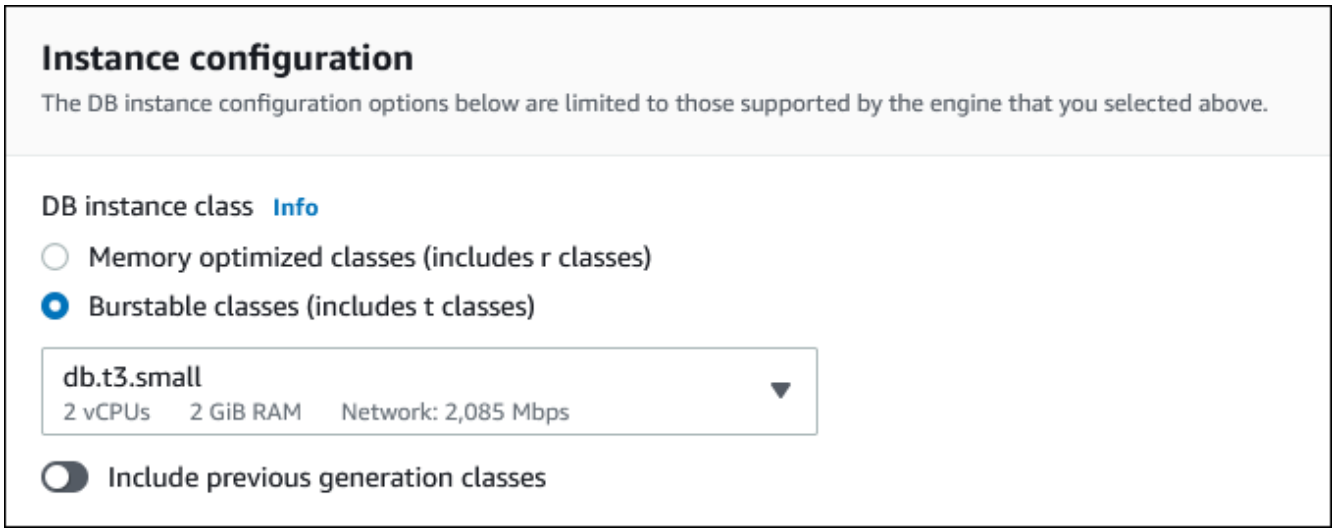
   - DB cluster identifier (Identificador do cluster de banco de dados): digite **`tutorial-db-cluster`**.
   - Master username (Nome de usuário principal): digite **`tutorial_user`**.
   - Auto generate a password (Gerar uma senha automaticamente): desabilite a opção.
   - Master password (Senha principal): digite uma senha.
   - Confirm password (Confirmar senha) – digite novamente a senha.



9. Na seção Instance configuration (Configuração da instância), defina estes valores:

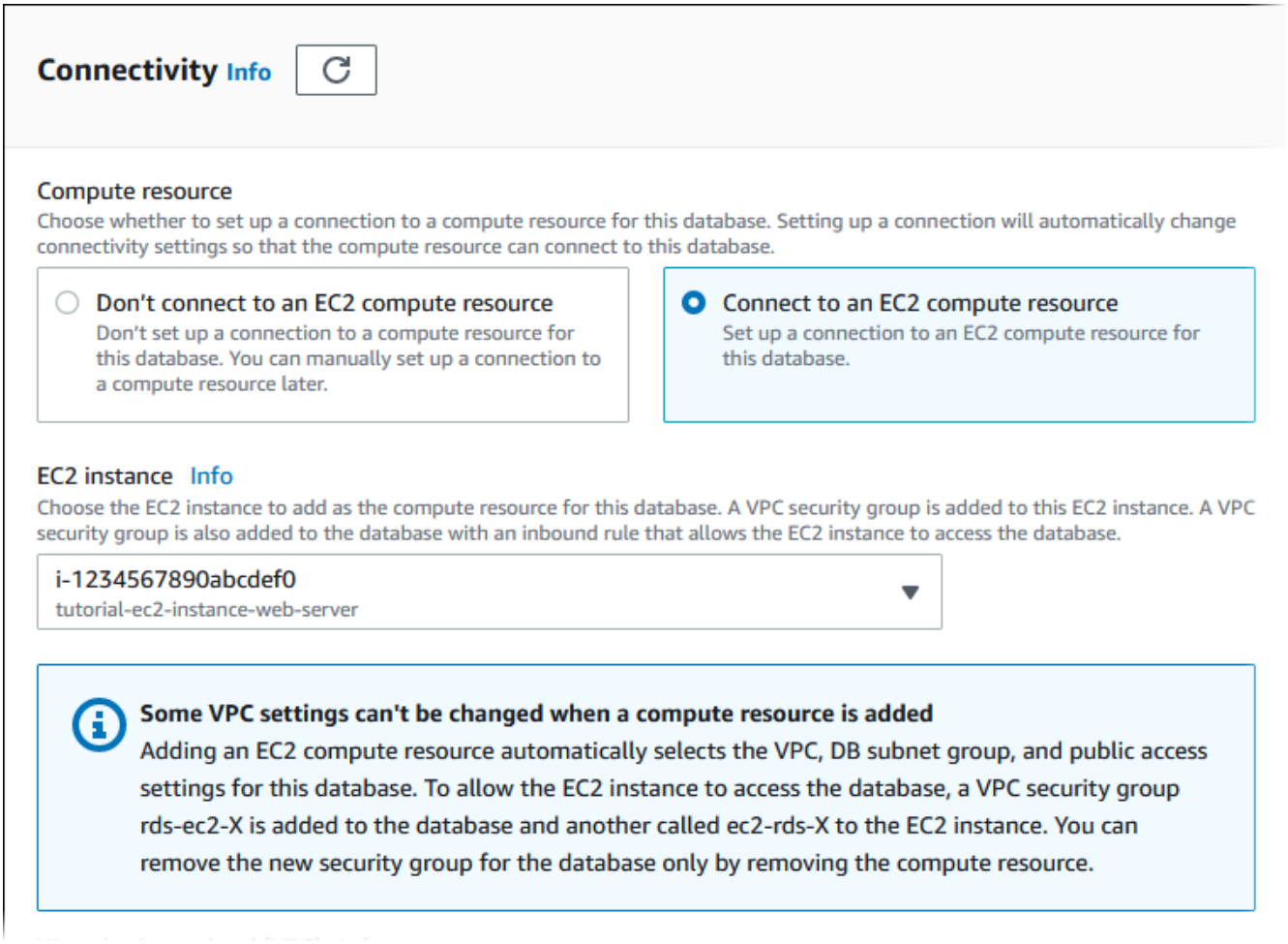
   - Classes com capacidade de intermitência (inclui classes t)
   - db.t3.small ou db.t3.medium

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.



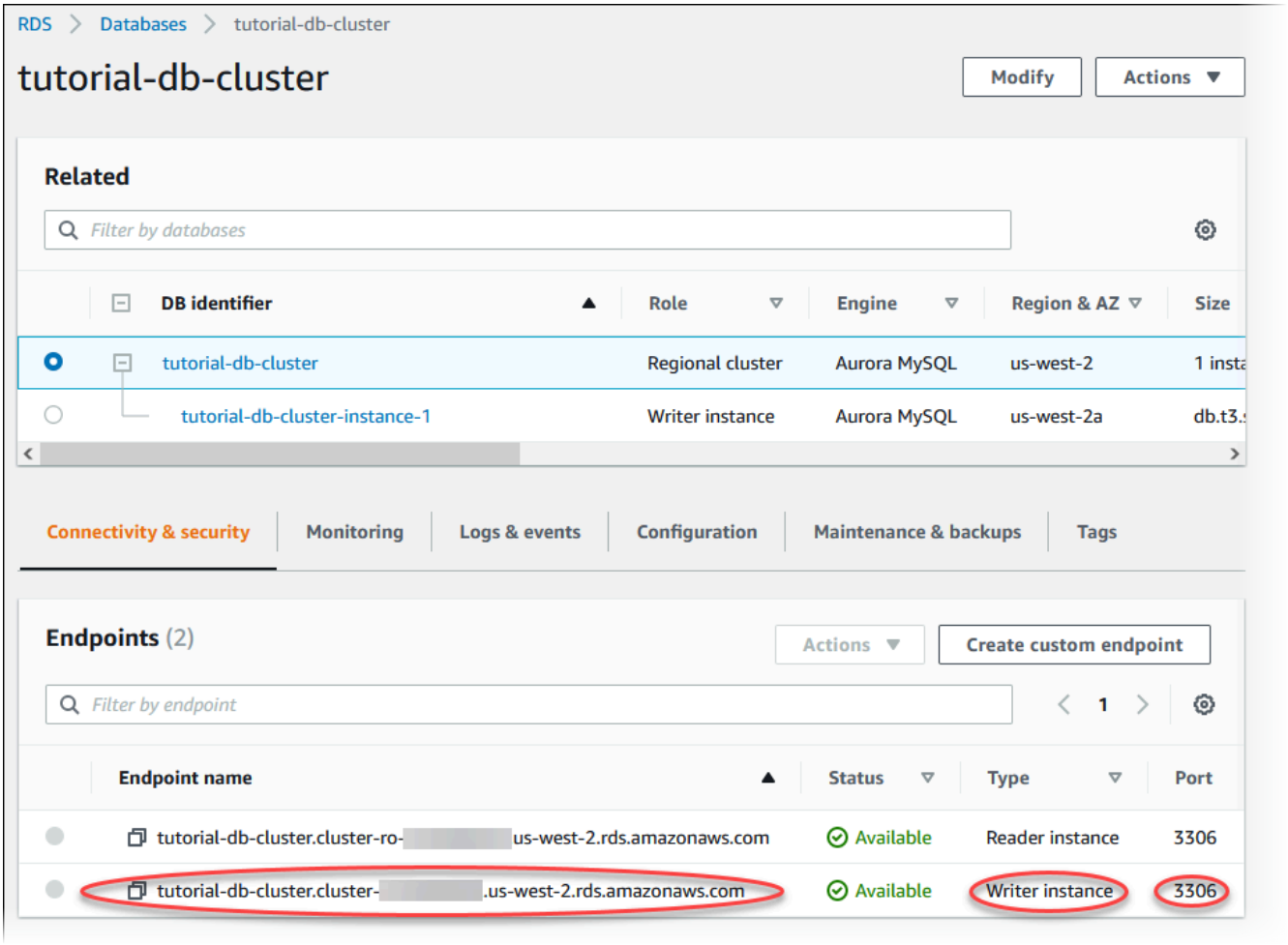
10. Na seção Availability and durability (Disponibilidade e durabilidade), use os valores padrão.
11. Na seção Connectivity (Conectividade), defina esses valores e mantenha os outros valores como padrão:

    - Em Compute resource (Recurso de computação), escolha Connect to an EC2 compute resource (Conectar-se a um recurso de computação do EC2).
    - Em EC2 instance (Instância do EC2), escolha a instância do EC2 criada anteriormente, como tutorial-ec2-instance-web-server.



12. Abra a seção Additional configuration (Configuração adicional) e insira **`sample`** em Initial database name (Nome do banco de dados inicial). Mantenha as configurações padrão para as outras opções.
13. Para criar o cluster de banco de dados Aurora MySQL, escolha Criar banco de dados.

    O novo cluster de banco de dados é exibido na lista Bancos de dados com o status Criando.
14. Aguarde o Status do novo cluster de banco de dados exibir Disponível. Escolha o nome do cluster de banco de dados para mostrar os detalhes.
15. Na seção Conectividade e segurança, visualize o Endpoint e a Porta da instância de banco de dados do gravador.



Anote o endpoint e a porta da instância de banco de dados do gravador. Use essas informações para conectar o servidor Web ao cluster de banco de dados.

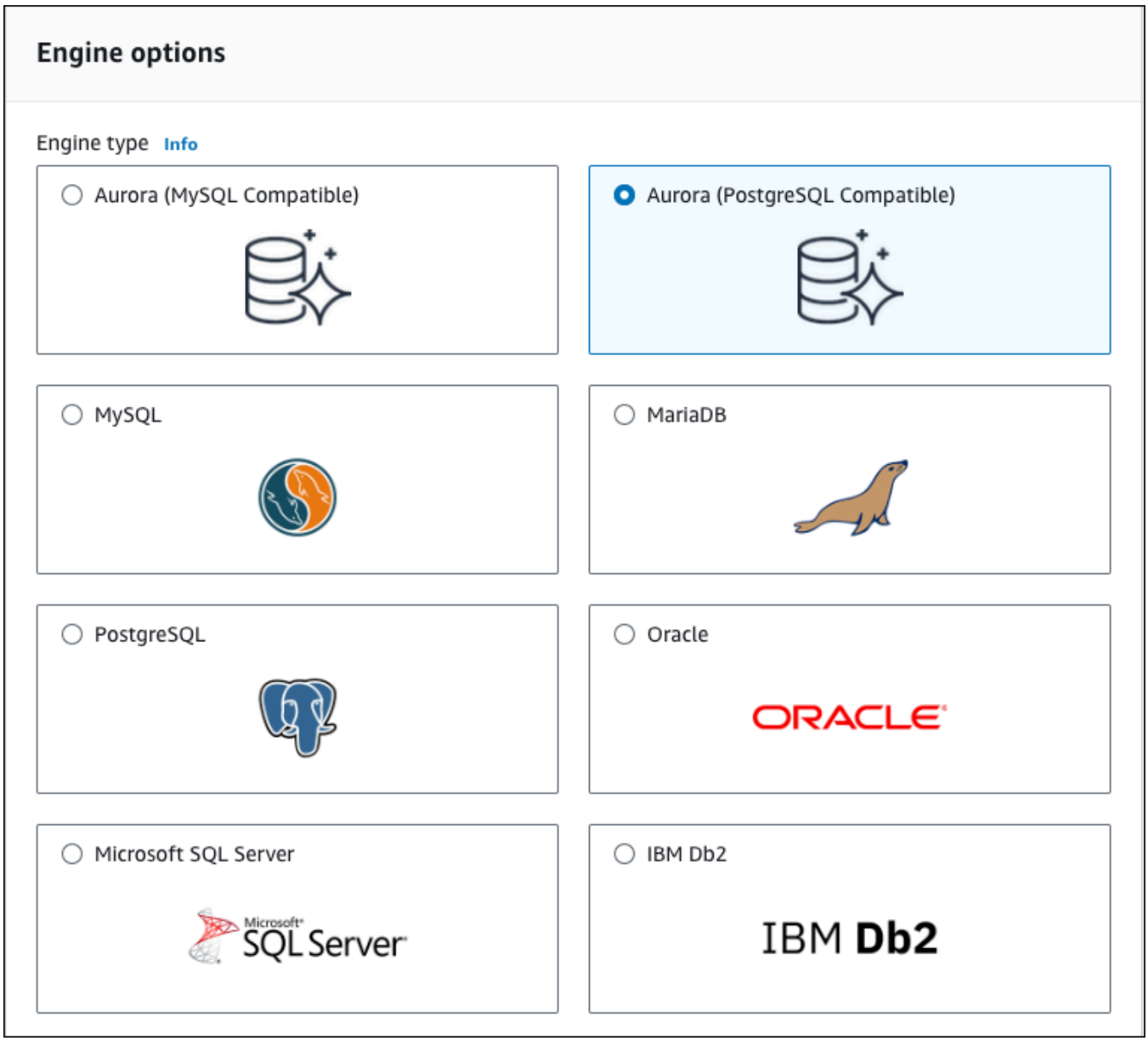
16. Completa Instalar um servidor Web na instância do EC2.

Aurora PostgreSQL

Criar um cluster de banco de dados do Aurora PostgreSQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, verifique se o Região da AWS é a mesma em que você criou sua instância do EC2.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Selecione Criar banco de dados.

5. Na página Criar banco de dados, escolha Criação padrão.
6. Em Opções de mecanismo, escolha Aurora (compatível com PostgreSQL).

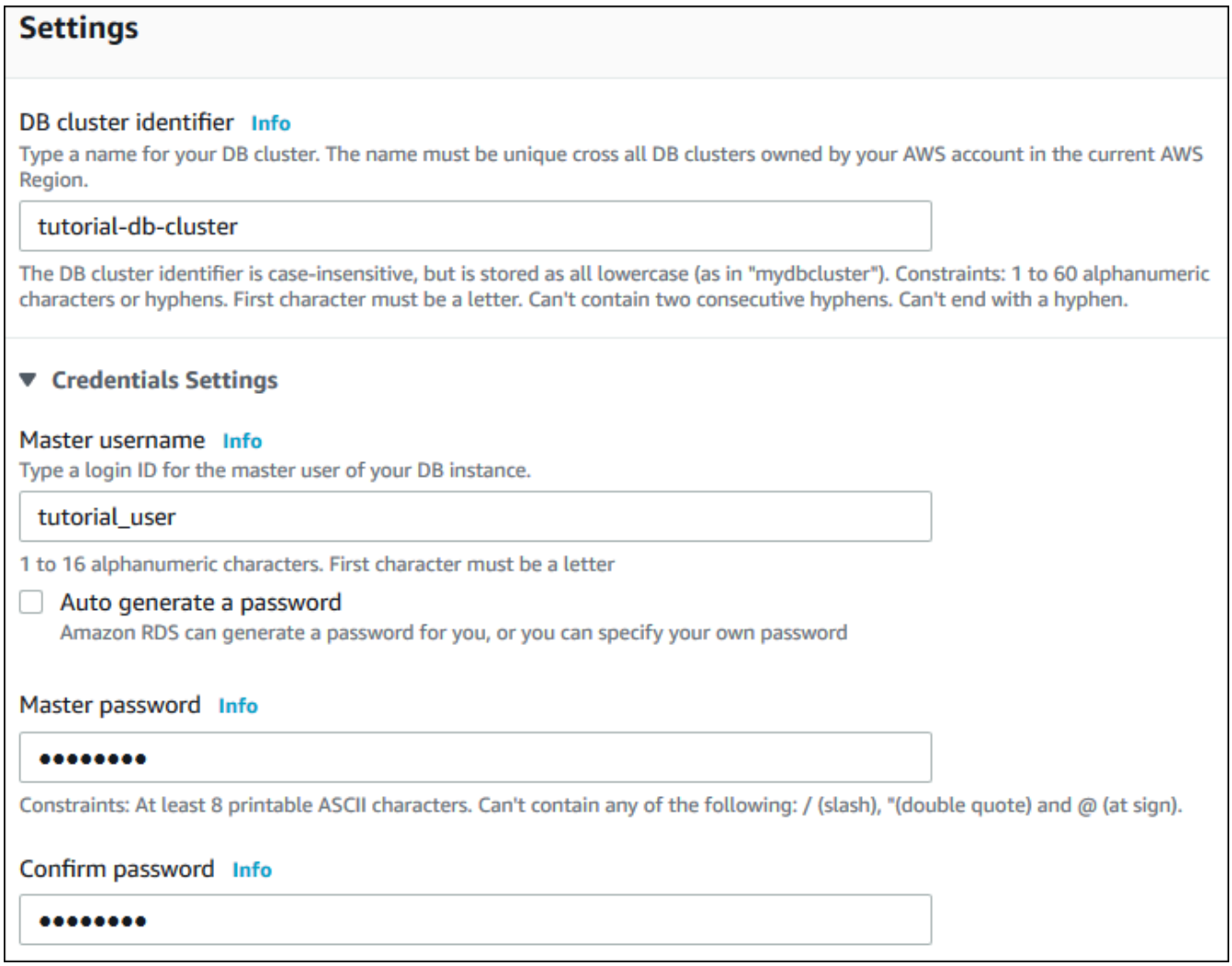


Mantenha os valores padrão para Versão e as outras opções do mecanismo.

7. Na seção Templates (Modelos), escolha Dev/Test.

8. Na seção Settings (Configurações), defina estes valores:

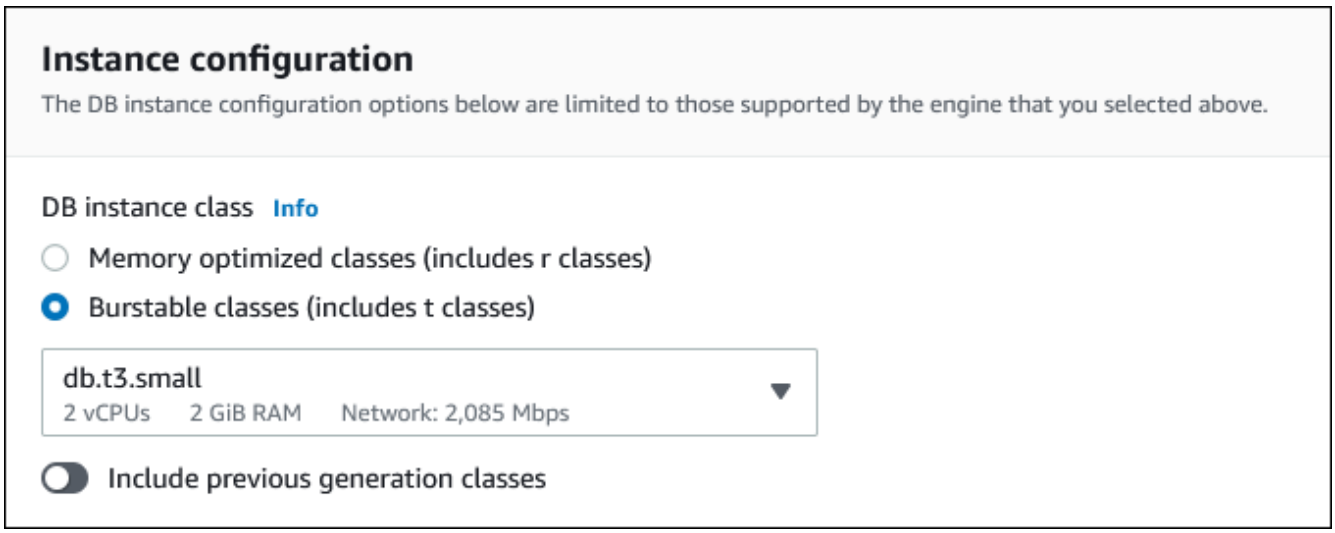
   - DB cluster identifier (Identificador do cluster de banco de dados): digite **`tutorial-db-cluster`**.
   - Master username (Nome de usuário principal): digite **`tutorial_user`**.
   - Auto generate a password (Gerar uma senha automaticamente): desabilite a opção.
   - Master password (Senha principal): digite uma senha.
   - Confirm password (Confirmar senha) – digite novamente a senha.



9. Na seção Instance configuration (Configuração da instância), defina estes valores:

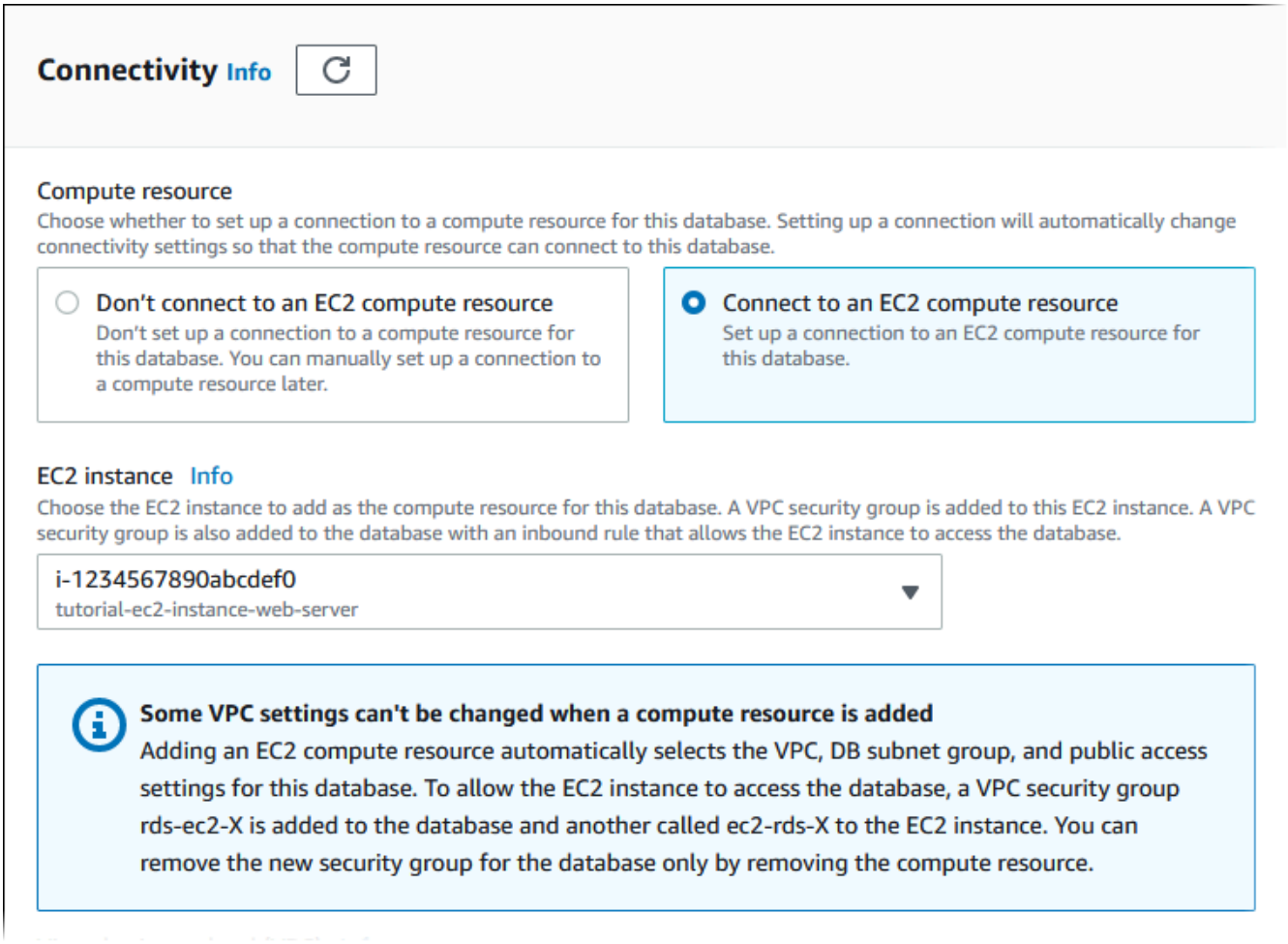
   - Classes com capacidade de intermitência (inclui classes t)
   - db.t3.small ou db.t3.medium

   > **Note**
   >
   > Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.



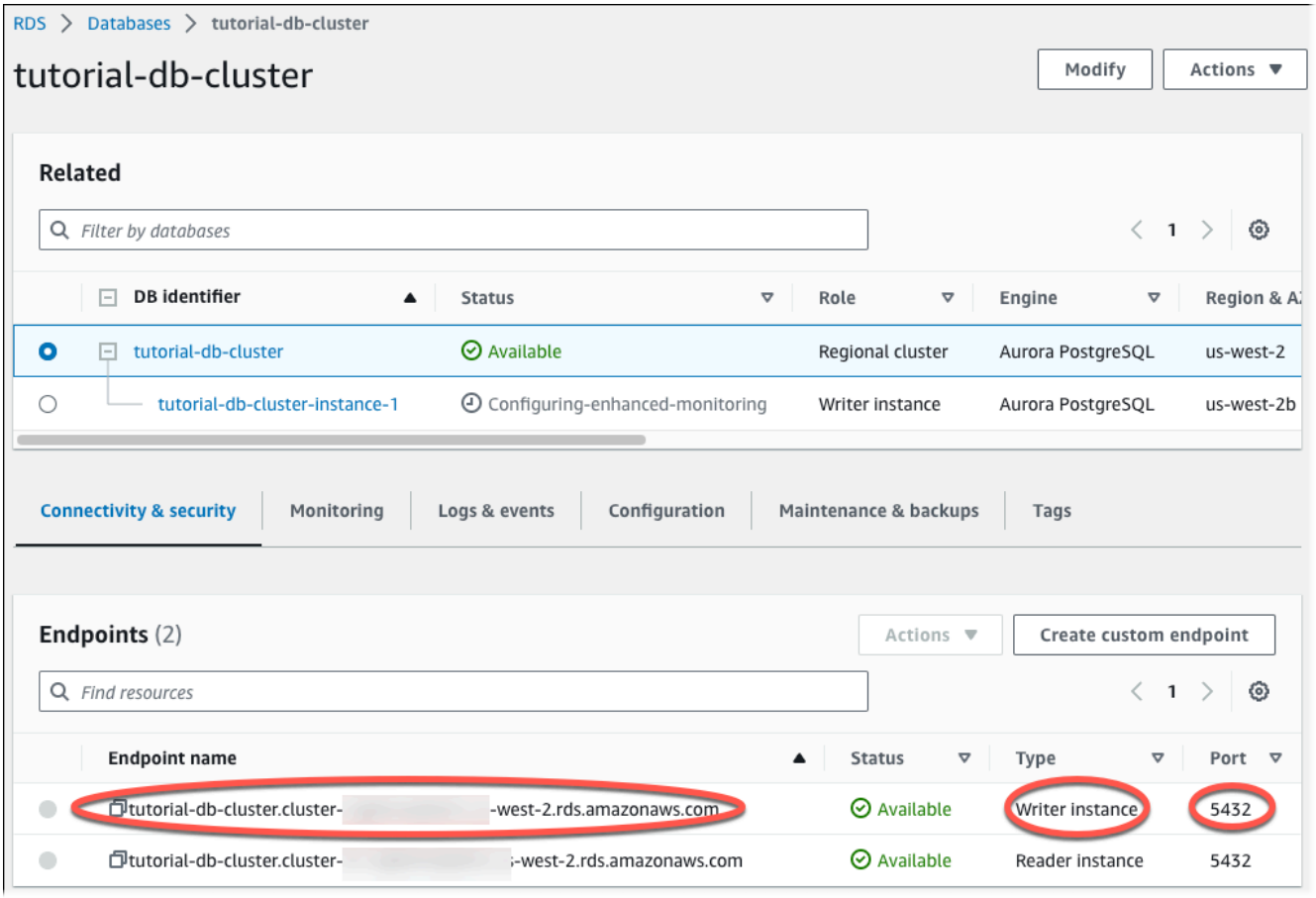
10. Na seção Availability and durability (Disponibilidade e durabilidade), use os valores padrão.
11. Na seção Connectivity (Conectividade), defina esses valores e mantenha os outros valores como padrão:

    - Em Compute resource (Recurso de computação), escolha Connect to an EC2 compute resource (Conectar-se a um recurso de computação do EC2).
    - Em EC2 instance (Instância do EC2), escolha a instância do EC2 criada anteriormente, como tutorial-ec2-instance-web-server.



12. Abra a seção Additional configuration (Configuração adicional) e insira **`sample`** em Initial database name (Nome do banco de dados inicial). Mantenha as configurações padrão para as outras opções.
13. Para criar o cluster de banco de dados do Aurora MySQL, escolha Criar banco de dados.

    O novo cluster de banco de dados é exibido na lista Bancos de dados com o status Criando.
14. Aguarde o Status do novo cluster de banco de dados exibir Disponível. Escolha o nome do cluster de banco de dados para mostrar os detalhes.
15. Na seção Conectividade e segurança, visualize o Endpoint e a Porta da instância de banco de dados do gravador.



Anote o endpoint e a porta da instância de banco de dados do gravador. Use essas informações para conectar o servidor Web ao cluster de banco de dados.

16. Completa Instalar um servidor Web na instância do EC2.

Instalar um servidor Web na instância do EC2

Instale um servidor Web na instância do EC2 criada em Iniciar uma instância do EC2 para se conectar ao cluster de banco de dados. O servidor Web se conecta ao cluster de banco de dados do Amazon Aurora criada em Criar um cluster de banco de dados do Amazon Aurora.

Instale um servidor Web do Apache com PHP e MariaDB

Conecte-se à sua instância do EC2 e instale o servidor Web.

Para conectar-se à sua instância do EC2 e instalar o servidor na web com PHP

1. Conecte-se à instância do EC2 que você criou anteriormente, seguindo as etapas em Conecte-se à sua instância do Linux no Guia do usuário do Amazon EC2.

Recomendamos que você se conecte à sua instância do EC2 utilizando SSH. Se o utilitário cliente SSH estiver instalado no Windows, Linux ou Mac, você poderá se conectar à instância utilizando o seguinte formato de comando:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Por exemplo, suponha que `ec2-database-connect-key-pair.pem` esteja armazenado em `/dir1` no Linux e que o DNS IPv4 público para sua instância do EC2 seja `ec2-12-345-678-90.compute-1.amazonaws.com`. Seu comando SSH teria a seguinte aparência:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

2. Obtenha as últimas correções de bugs e atualizações de segurança atualizando o software na instância do EC2. Para fazer isso, execute o seguinte comando.

> **Note**
>
> A opção `-y` instala as atualizações sem solicitar confirmação. Para examinar atualizações antes da instalação, omita essa opção.

```
sudo dnf update -y
```

3. Após a conclusão das atualizações, instale o servidor web Apache, o PHP e o software MariaDB ou PostgreSQL utilizando os comandos a seguir. Este comando instala vários pacotes de software e dependências relacionadas ao mesmo tempo.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

Se você receber um erro, isso significa que sua instância provavelmente não foi iniciada com uma AMI do Amazon Linux 2023. Em vez disso é, possível utilizar a AMI do Amazon Linux 2. Você pode visualizar sua versão do Amazon Linux usando o comando a seguir.

```
cat /etc/system-release
```

Para ter mais informações, consulte Atualização de software da instância.

4. Inicie o servidor Web com o comando mostrado a seguir.

```
sudo systemctl start httpd
```

É possível testar se o servidor Web está instalado e se foi iniciado corretamente. Para fazer isso, insira o nome de Domain Name System (DNS) público da instância do EC2 na barra de endereços de um navegador da web, por exemplo: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. Se o seu servidor na web estiver em execução, você verá a página de teste do Apache.

Se você não vir a página de teste do Apache, verifique as regras de entrada para o grupo de segurança da VPC criado no Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4). Verifique se as regras de entrada incluem uma regra que permita o acesso HTTP (porta 80) ao endereço IP para se conectar ao servidor da Web.

> **Note**
>
> A página de teste do Apache aparece apenas quando não há conteúdo no diretório raiz do documento, `/var/www/html`. Depois de adicionar o conteúdo ao diretório raiz de documentos, o conteúdo aparecerá no endereço DNS público da instância do EC2. Antes desse ponto, ele aparece na página de teste do Apache.

5. Configure o servidor Web para começar com cada inicialização do sistema usando o comando `systemctl`.

```
sudo systemctl enable httpd
```

Para permitir que `ec2-user` gerencie arquivos no diretório raiz padrão de servidor Web do Apache, modifique a propriedade e as permissões do diretório `/var/www`. Existem diversas maneiras de realizar essa tarefa. Neste tutorial, você adiciona o usuário `ec2-user` ao grupo `apache` para dar ao grupo `apache` a propriedade do diretório `/var/www` e atribuir permissões de gravação ao grupo.

Para definir as permissões de arquivos para o servidor na web Apache

1. Adicione o usuário `ec2-user` ao grupo `apache`.

   ```
   sudo usermod -a -G apache ec2-user
   ```

2. Faça logout para atualizar as permissões e incluir o novo grupo `apache`.

   ```
   exit
   ```

3. Faça login novamente e verifique se o grupo `apache` existe com o comando `groups`.

   ```
   groups
   ```

   A saída será semelhante à seguinte:

   ```
   ec2-user adm wheel apache systemd-journal
   ```

4. Altere a propriedade do grupo do diretório `/var/www` e o seu conteúdo para o grupo `apache`.

   ```
   sudo chown -R ec2-user:apache /var/www
   ```

5. Altere as permissões do diretório do `/var/www` e dos subdiretórios para adicionar permissões de gravação do grupo e definir o ID do grupo em subdiretórios criados futuramente.

   ```
   sudo chmod 2775 /var/www
   find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

6. Altere recursivamente as permissões de arquivos do diretório `/var/www` e os subdiretórios para adicionar permissões de gravação.

   ```
   find /var/www -type f -exec sudo chmod 0664 {} \;
   ```

Agora, `ec2-user` (e todos os outros membros do grupo `apache`) pode adicionar, excluir e editar arquivos na raiz do documento Apache. Isso possibilita que você adicione conteúdo, como um site estático ou uma aplicação PHP.

> **Note**
>
> Um servidor web que executa o protocolo HTTP não fornece nenhuma segurança de transporte para os dados que envia ou recebe. Quando você se conecta a um servidor HTTP usando um navegador da Web, muitas informações ficam visíveis para os espiões em qualquer ponto da rede. Essas informações incluem os URLs que você acessa, o conteúdo de páginas da web recebido e o conteúdo (inclusive senhas) de todos os formulários HTML. A prática recomendada para proteger o servidor da Web é instalar suporte para HTTPS (HTTP seguro). Esse protocolo protege seus dados com criptografia SSL/TLS. Para ter mais informações, consulte Tutorial: Configurar o SSL/TLS com a AMI do Amazon Linux no Guia do Usuário do Amazon EC2.

Conectar o servidor Web Apache ao cluster de banco de dados

Depois, adicione o conteúdo ao servidor Web Apache que se conecta ao cluster de banco de dados do Amazon Aurora.

Como adicionar o conteúdo ao servidor Web Apache que se conecta ao cluster de banco de dados

1. Enquanto estiver conectado à instância do EC2, altere o diretório para `/var/www` e crie um novo subdiretório chamado `inc`.

   ```
   cd /var/www
   mkdir inc
   cd inc
   ```

2. Crie um novo arquivo no diretório `inc` chamado `dbinfo.inc` e edite o arquivo chamando nano (ou o editor de sua escolha).

   ```
   >dbinfo.inc
   nano dbinfo.inc
   ```

3. Adicione o conteúdo a seguir ao arquivo `dbinfo.inc`. Aqui, *`db_instance_endpoint`* é o endpoint do gravador de cluster de banco de dados, sem a porta, para seu cluster de banco de dados.

   > **Note**
   >
   > Recomendamos colocar as informações de nome de usuário e senha em uma pasta que não faça parte da raiz do documento do servidor Web. Isso reduz a possibilidade de suas informações de segurança serem expostas.
   > Altere `master password` para uma senha adequada em sua aplicação.

   ```
   <?php

   define('DB_SERVER', 'db_cluster_writer_endpoint');
   define('DB_USERNAME', 'tutorial_user');
   define('DB_PASSWORD', 'master password');
   define('DB_DATABASE', 'sample');
   ?>
   ```

4. Salve e feche o arquivo `dbinfo.inc`. Se você estiver usando nano, salve e feche o arquivo usando Ctrl+S e Ctrl+X.
5. Altere o diretório para `/var/www/html`.

   ```
   cd /var/www/html
   ```

6. Crie um novo arquivo no diretório `html` chamado `SamplePage.php` e edite o arquivo chamando nano (ou o editor de sua escolha).

   ```
   >SamplePage.php
   nano SamplePage.php
   ```

7. Adicione o conteúdo a seguir ao arquivo `SamplePage.php`:

   MariaDB & MySQL

   ```
   <?php include "../inc/dbinfo.inc"; ?>
   <html>
   <body>
   <h1>Sample page</h1>
   ```

```
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>

<!-- Display table data. -->
```

```
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";
```

```
   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
<?php
```

```
/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}
```

```
/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. Salve e feche o arquivo `SamplePage.php`.
9. Verifique se o servidor Web se conecta com êxito ao cluster de banco de dados, abrindo um navegador da Web e navegando até `http://`*`EC2 instance endpoint`*`/SamplePage.php`, por exemplo: `http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php`.

É possível usar `SamplePage.php` para adicionar dados ao cluster de banco de dados. Os dados que você adicionar serão exibidos na página. Para verificar se os dados foram inseridos na tabela, instale o cliente MySQL na instância do Amazon EC2. Depois, conecte-se ao cluster de banco de dados e consulte a tabela.

Para informações sobre como se conectar a um cluster de banco de dados, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Para garantir que o cluster de banco de dados seja o mais seguro possível, verifique se as fontes fora da VPC não podem se conectar ao cluster de banco de dados.

Depois de terminar de testar o servidor web e o banco de dados, é necessário excluir o cluster de banco de dados e a instância do Amazon EC2.

- Para criar um cluster de banco de dados, siga as instruções em Excluir clusters de banco de dados e instâncias de banco de dados do Aurora. Não é necessário criar um snapshot final.
- Para encerrar uma instância do Amazon EC2, siga as instruções em Encerrar a instância no Guia do usuário do Amazon EC2.

Tutoriais do Amazon Aurora e código de exemplo

A documentação da AWS contém vários tutoriais que guiarão você por meio de casos de uso comuns do Amazon Aurora. Muitos desses tutoriais mostram como usar o Amazon Aurora com outros serviços da AWS. Além disso, é possível acessar o código de exemplo em .

> **Note**
>
> Você pode encontrar mais tutoriais no AWS Database Blog. Para ter mais informações sobre treinamento, consulte AWS Training and Certification.

Tópicos

- Tutoriais neste guia
- Tutoriais em outros guias daAWS
- Tutoriais e código de exemplo no GitHub
- Livro de fórmulas para bancos de dados da AWS
- Portal de conteúdo de workshops e laboratório da AWS para Amazon Aurora PostgreSQL
- Portal de conteúdo de workshops e laboratório da AWS para Amazon Aurora MySQL
- Using this service with an AWS SDK

Tutoriais neste guia

Os seguintes tutoriais neste guia mostram como realizar tarefas comuns ao usar o Amazon Aurora:

- Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4)

  Saiba como incluir um cluster de banco de dados em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Nesse caso, a VPC compartilha dados com um servidor da Web em execução em uma instância do Amazon EC2 na mesma VPC.
- Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla)

  Saiba como incluir um cluster de banco de dados em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Nesse caso, a VPC compartilha dados com uma instância do Amazon

EC2 na mesma VPC. Neste tutorial, você criará a VPC para esse cenário que funciona com um banco de dados em execução no modo de pilha dupla.

- Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora

  Este tutorial ajuda você a instalar um servidor Web Apache com PHP e a criar um banco de dados MySQL. O servidor da web é executado em uma instância do Amazon EC2 usando o Amazon Linux, e o banco de dados MySQL é um cluster de bancos de dados Aurora MySQL. Tanto a instância do Amazon EC2 quanto o cluster da de banco de dados são executados em uma Amazon VPC.

- Tutorial: Restaurar um cluster de banco de dados do Amazon Aurora de um snapshot de cluster de banco de dados

  Aprenda a restaurar um cluster de banco de dados de um snapshot do cluster de banco de dados.

- Tutorial: Uso de tags para especificar quais clusters de bancos de dados Aurora interromper

  Saiba como usar tags para especificar quais clusters de bancos de dados Aurora interromper.

- Tutorial: Registro de alterações de estado de uma instância de banco de dados usando o Amazon EventBridge

  Saiba como registrar uma alteração de estágio de instância de banco de dados usando o Amazon EventBridge e AWS Lambda.

Tutoriais em outros guias daAWS

Os seguintes tutoriais em outros guias da AWS mostram como realizar tarefas comuns com o Amazon Aurora:

> **Note**
>
> Alguns dos tutoriais usam instâncias de banco de dados do Amazon RDS, mas podem ser adaptados para usar clusters de bancos de dados Aurora.

- Tutorial: Aurora Serverless no Guia do desenvolvedor do AWS AppSync

  Aprenda a usar o AWS AppSync para fornecer uma fonte de dados a fim de executar comandos SQL em clusters de banco de dados do Aurora Serverless com a API de dados ativada. É possível

usar resolvedores do AWS AppSync para executar declarações SQL na API de dados com consultas, mutações e assinaturas do GraphQL.

- Tutorial: Alternar um segredo para um banco de dados da AWS no Manual do usuário do AWS Secrets Manager

  Neste tutorial, você criará um segredo para um banco de dados da AWS e configurará o banco de dados para alternar em um cronograma. Você acionará uma rotação manualmente e confirmará se a nova versão do segredo continua fornecendo acesso.

- Tutoriais e amostras no Guia do desenvolvedor do AWS Elastic Beanstalk

  Saiba como implantar aplicações que usam bancos de dados do Amazon RDS com o AWS Elastic Beanstalk.

- Uso dos dados de um banco de dados Amazon RDS para criar uma fonte de dados do Amazon ML no Amazon Machine Learning Developer Guide

  Aprenda a criar um objeto de fonte de dados do Amazon Machine Learning (Amazon ML) a partir de dados armazenados em uma instância de banco de dados MySQL.

- Habilitação manual de acesso a uma instância do Amazon RDS em uma VPC no Guia do usuário do Amazon Quick

  Saiba como habilitar o acesso do Quick a uma instância de banco de dados do Amazon RDS em uma VPC.

Tutoriais e código de exemplo no GitHub

Os seguintes tutoriais e código de exemplo do GitHub mostram como realizar tarefas comuns ao usar o Amazon Aurora:

- Criar uma biblioteca de empréstimos do Aurora Serverless v2

  Aprenda a criar uma aplicação de biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados. O exemplo usa Aurora Serverless v2 e AWS SDK para Python (Boto3).

- Criar uma aplicação de rastreador de itens do Amazon Aurora com uma API Spring REST que consulta dados do Aurora Serverless v2 usando o SDK para Java 2.x

Saiba como criar uma API Spring REST que consulta dados do Aurora Serverless v2. É para uso por uma aplicação React usando o SDK for Java 2.x.

- Criar uma aplicação de rastreador de itens do Amazon Aurora que consulta dados do Aurora Serverless v2 usando AWS SDK para PHP

  Aprenda a criar uma aplicação que use o `RdsDataClient` da API de dados e o Aurora Serverless v2 para monitorar e gerar relatórios sobre itens de trabalho. O exemplo usa AWS SDK para PHP.

- Criar uma aplicação de rastreador de itens do Amazon Aurora que consulta dados do Aurora Serverless v2 usando AWS SDK para Python (Boto3)

  Aprenda a criar uma aplicação que use o `RdsDataClient` da API de dados e o Aurora Serverless v2 para monitorar e gerar relatórios sobre itens de trabalho. O exemplo usa AWS SDK para Python (Boto3).

Livro de fórmulas para bancos de dados da AWS

O Livro de fórmulas para bancos de dados da AWS é um guia abrangente de banco de dados que ensina a criar, implantar e gerenciar soluções de banco de dados econômicas e de alta performance na AWS. Tutoriais passo a passo orientam você na criação de aplicações prontas para produção e na implantação das aplicações com os modelos do CloudFormation. Você conhecerá serviços essenciais da AWS ao criar infraestrutura, implementar redes, desenvolver arquiteturas sem servidor, gerenciar bancos de dados e integrar a IA generativa. Conheça as práticas recomendadas da AWS que ajudam você a criar soluções seguras e escaláveis e, ao mesmo tempo, otimizar os custos. Seja você iniciante na AWS ou um profissional experiente, o Livro de fórmulas para bancos de dados da AWS ajuda você a desenvolver habilidades para resolver desafios comuns de banco de dados e implementar soluções prontas para empresas. O livro de fórmulas inclui as seguintes seções:

- Conceitos básicos da AWS para aplicações de banco de dados: conheça os fundamentos da AWS, como configurar sua conta e o ambiente do caderno Jupyter.
- Fundamentos do banco de dados: conheça os conceitos essenciais do banco de dados e compare os serviços de banco de dados da AWS para escolher a solução certa para suas workloads.
- Aplicação Web sem servidor com o Amazon Aurora: crie uma aplicação de varejo de ponta a ponta com o Amazon Aurora PostgreSQL que gerencia inventário, pedidos e dados de clientes.
- Monitoramento e observabilidade: configure o acompanhamento de performance e defina alertas para identificar possíveis problemas no banco de dados antes que eles afetem suas aplicações.

- Escalabilidade com o Amazon Aurora: aprenda a criar implantações multirregionais resilientes com o Aurora DSQL e a aumentar a escala de seus bancos de dados verticalmente para conquistar mais poder de processamento ou horizontalmente em várias instâncias para aumentar a capacidade.
- Performance e custo de otimização: otimize a performance do seu banco de dados e reduza os custos com estratégias de ajuste comprovadas.
- Jornada para bancos de dados com propósito específico da AWS: crie uma infraestrutura segura e confiável que escale suas soluções de IA generativa e aplicações orientadas por dados, desde o protótipo até a implantação empresarial.
- Aplicações de IA generativa com RAG: crie um sistema de pesquisa inteligente para documentos de seguros e assistência médica que use geração aumentada via recuperação (RAG) para fornecer resultados precisos e sensíveis ao contexto.

Portal de conteúdo de workshops e laboratório da AWS para Amazon Aurora PostgreSQL

A seguinte coleção de workshops e outros conteúdos práticos ajudam você a entender os recursos e as capacidades do Amazon Aurora PostgreSQL:

- Criar um cluster de banco de dados do Aurora manualmente

  Saiba como criar um cluster do Amazon Aurora PostgreSQL manualmente.
- Configurar o Cloud9 e inicializar o banco de dados

  Saiba como configurar o Cloud9 e inicializar o banco de dados PostgreSQL.
- Clonagem rápida

  Saiba como criar um clone rápido do Aurora.
- Gerenciamento de planos de consulta

  Saiba como controlar os planos de execução de um conjunto de instruções utilizando o gerenciamento de planos de consultas.
- Gerenciamento de cache de cluster

  Saiba mais sobre o recurso de gerenciamento de cache de cluster no Aurora PostgreSQL.
- Fluxos de atividades do banco de dados

Saiba como criar monitorar e auditar a atividade de seu banco de dados com esse recurso.

- Utilizar o Performance Insights

  Saiba como monitorar e ajustar sua instância de banco de dados utilizando o Performance Insights.

- Monitoramento de performance com ferramentas do RDS

  Saiba como utilizar a AWS e as ferramentas do Postgres (Cloudwatch, Enhanced Monitoring, Slow Query Logs, Performance Insights, PostgreSQL Catalog Views) para entender os problemas de performance e identificar formas de melhorar a performance de seu banco de dados.

- Réplicas de leitura com ajuste de escala automático

  Saiba como o ajuste de escala automático de réplicas de leitura do Aurora funciona na prática utilizando um script gerador de carga.

- Testar a tolerância a falhas

  Saiba como um cluster de banco de dados pode tolerar uma falha.

- Banco de dados global Aurora

  Saiba mais sobre o Aurora Global Database.

- Utilizar o machine learning

  Saiba mais sobre o Aurora Machine Learning.

- Aurora Serverless v2

  Saiba mais sobre o Aurora Serverless v2.

- Trusted Language Extensions para o Aurora PostgreSQL

  Saiba como criar extensões de alta performance que são executadas com segurança no Aurora PostgreSQL.

Portal de conteúdo de workshops e laboratório da AWS para Amazon Aurora MySQL

A seguinte coleção de workshops e outros conteúdos práticos ajudam você a entender os recursos e as capacidades do Amazon Aurora MySQL:

- Criar um cluster do Aurora

  Saiba como criar um cluster do Amazon Aurora MySQL manualmente.

- Criar um ambiente IDE baseado em nuvem do Cloud9 para se conectar ao seu banco de dados

  Saiba como configurar o Cloud9 e inicializar o banco de dados MySQL.

- Clonagem rápida

  Saiba como criar um clone rápido do Aurora.

- Retroceder um cluster

  Saiba como retroceder um cluster de banco de dados.

- Utilizar o Performance Insights

  Saiba como monitorar e ajustar sua instância de banco de dados utilizando o Performance Insights.

- Monitoramento de performance com ferramentas do RDS

  Saiba como utilizar a AWS e as ferramentas SQL para entender os problemas de performance e identificar maneiras de melhorar a performance de seu banco de dados.

- Analisar a performance da consulta

  Saiba como solucionar problemas relacionados à performance do SQL utilizando ferramentas diferentes.

- Réplicas de leitura com ajuste de escala automático

  Saiba como funcionam as réplicas de leitura com ajuste de escala automático.

- Testar a tolerância a falhas

  Saiba mais sobre os recursos de alta disponibilidade e tolerância a falhas no Aurora MySQL.

- Banco de dados global Aurora

  Saiba mais sobre o Aurora Global Database.

- Aurora Serverless v2

  Saiba mais sobre o Aurora Serverless v2.

- Utilizar o machine learning

Saiba mais sobre o Aurora Machine Learning.

Using this service with an AWS SDK

Os kits de desenvolvimento de software (SDKs) da AWS estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ Exemplos de código
AWS CLI	AWS CLI Exemplos de código da
AWS SDK para Go	AWS SDK para Go Exemplos de código da
AWS SDK para Java	AWS SDK para Java Exemplos de código da
AWS SDK para JavaScript	AWS SDK para JavaScript Exemplos de código da
AWS SDK para Kotlin	AWS SDK para Kotlin Exemplos de código da
AWS SDK para .NET	AWS SDK para .NET Exemplos de código da
AWS SDK para PHP	AWS SDK para PHP Exemplos de código da
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell Exemplos de código da
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) Exemplos de código da
AWS SDK para Ruby	AWS SDK para Ruby Exemplos de código da
AWS SDK para Rust	AWS SDK para Rust Exemplos de código da
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP Exemplos de código da

Documentação do SDK	Exemplos de código
AWS SDK para Swift	AWS SDK para Swift Exemplos de código do

Para obter exemplos específicos deste serviço, consulte Exemplos de código para o Aurora usando AWS SDKs.

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Acesso programático ao Amazon Aurora

O Amazon RDS fornece as seguintes ferramentas para gerenciar seus recursos do Amazon Aurora de forma programática.

AWS Command Line Interface (AWS CLI)

Você pode criar e gerenciar seus recursos do Amazon Aurora usando a AWS CLI em um shell da linha de comandos. A AWS CLI fornece acesso direto às APIs para serviços da AWS, como o Amazon RDS. Consulte a sintaxe e exemplos dos comandos do Amazon RDS em rds na Referência de comandos da AWS CLI.

AWS CloudFormation

Com essa ferramenta de infraestrutura como código (IaC) da AWS, é possível criar modelos que descrevem todos os recursos do Amazon Aurora que você desejar, e o AWS CloudFormation provisiona e configura esses recursos para você. Para ter mais informações, consulte the section called “Criar recursos com o AWS CloudFormation”.

Kits de desenvolvimento de software (SDKs) da AWS

A AWS fornece SDKs para muitas tecnologias e linguagens de programação conhecidas. Eles facilitam a chamada de serviços da AWS de dentro de suas aplicações nessa linguagem ou tecnologia. Consulte mais informações sobre esses SDKs em Ferramentas para desenvolver e gerenciar aplicações na AWS.

API do Amazon RDS

Essa API é a interface em nível de protocolo para o Amazon RDS. Ao usar essa API, você deve formatar cada solicitação HTTPS corretamente e adicionar uma assinatura digital válida a cada solicitação. Para ter mais informações, consulte Referência da API do Amazon RDS.

Console-to-Code

Com essa ferramenta, é possível gerar código para ações que você executa no console do Amazon RDS e usar esse código em outras ferramentas, como o AWS CloudFormation. Para ter mais informações, consulte the section called “Console-to-Code”.

API de dados do RDS

A API de dados do RDS (API de dados) oferece acesso programático aos dados dos bancos de dados do Amazon Aurora Sem Servidor. Com a API de dados, é possível usar um endpoint

HTTP seguro para executar instruções SQL sem gerenciar conexões. Para ter mais informações, consulte Usar a API de dados do Amazon RDS.

Usar o Console-to-Code para gerar código para as suas ações no console do Amazon Aurora

O console oferece um caminho guiado para criar recursos e testar protótipos. Se quiser criar os mesmos recursos em escala, você precisará de um código de automação. O Console-to-Code é um atributo do Amazon Q Developer que pode ajudar você a começar a usar seu código de automação. O Console-to-Code registra as ações que você executa no console, incluindo os valores padrão e os valores de parâmetros fornecidos. Depois, ele usa IA generativa para sugerir código no idioma e formato de sua preferência para as ações desejadas. Como o fluxo de trabalho do console garante que os valores dos parâmetros que você especifica sejam válidos juntos, o código que você gera usando o Console-to-Code tem valores de parâmetros compatíveis. Você pode usar o código como ponto de partida e, depois, personalizá-lo para deixá-lo pronto para produção no seu caso de uso específico.

Por exemplo, com o Console-to-Code, é possível gravar a criação de um cluster de banco de dados do Aurora e optar por gerar código no formato JSON do AWS CloudFormation. Em seguida, você pode copiar esse código e personalizá-lo para uso em seu modelo do AWS CloudFormation.

Atualmente, o Console-to-Code pode gerar infraestrutura como código (IaC) nos seguintes formatos e linguagens:

- Java do CDK
- Python do CDK
- TypeScript do CDK
- JASON do CloudFormation
- YAML do CloudFormation

Para obter mais informações e instruções sobre como usar o Console-to-Code, consulte Automatizar serviços da AWS com o Console-to-Code do Amazon Q Developer no Guia do usuário do Amazon Q Developer.

Configurar um cluster de banco de dados do Amazon Aurora

Esta seção mostra como configurar o cluster de banco de dados Aurora. Antes de criar um cluster de banco de dados Aurora, é preciso decidir sobre a classe da instância de banco de dados que executará o cluster de banco de dados. Além disso, decida em que local o cluster de banco de dados será executado, escolhendo a região da AWS. Em seguida, crie o cluster de banco de dados. Se você tiver dados que se encontram fora do Aurora, poderá migrá-los para um cluster de banco de dados Aurora.

Tópicos

- Criar um cluster de bancos de dados do Amazon Aurora
- Criar recursos do Amazon Aurora com o AWS CloudFormation
- Como conectar-se a um cluster de bancos de dados Amazon Aurora
- Grupos de parâmetros para Amazon Aurora
- Migrar dados para um cluster de banco de dados do Amazon Aurora
- Criar um cache do Amazon ElastiCache usando as configurações do cluster de banco de dados do Aurora
- Migração automática de bancos de dados do EC2 para o Amazon Aurora usando o AWS Database Migration Service
- Tutorial: criar um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado

Criar um cluster de bancos de dados do Amazon Aurora

Um cluster de banco de dados do Amazon Aurora consiste em uma instância de banco de dados compatível com o MySQL ou o PostgreSQL e um volume de cluster que detém os dados do cluster de banco de dados copiados em três zonas de disponibilidade como um único volume virtual. Por padrão, um cluster de banco de dados Aurora contém uma instância de banco de dados primária que faz leituras e gravações e, como opção, até 15 réplicas do Aurora (instâncias de banco de dados de leitor). Para obter mais informações sobre clusters de bancos de dados Aurora, consulte Clusters de banco de dados do Amazon Aurora.

O Aurora tem dois tipos principais de cluster de banco de dados:

- Provisionado pelo Aurora: você escolhe a classe de instância de banco de dados para as instâncias do gravador e do leitor com base na workload prevista. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora. O provisionado pelo Aurora tem várias opções, incluindo bancos de dados globais do Aurora. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.
- Aurora Serverless: Aurora Serverless v2 é uma configuração de ajuste de escala automático sob demanda do Aurora. A capacidade é ajustada automaticamente com base na demanda da aplicação. Você será cobrado apenas pelos recursos que seu cluster de banco de dados consumir. Essa automação é especialmente útil para ambientes com workloads altamente variáveis e imprevisíveis. Para obter mais informações, consulte Usar o Aurora Serverless v2.

A seguir, é possível descobrir como criar um cluster de banco de dados Aurora. Para começar, primeiro consulte Pré-requisitos do cluster de banco de dados.

Para obter instruções sobre como se conectar ao cluster de bancos de dados Aurora, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Sumário

- Pré-requisitos do cluster de banco de dados
  - Configurar a rede para o cluster de banco de dados
    - Configurar a conectividade automática de rede com uma instância do EC2
    - Configurar a rede manualmente
  - Pré-requisitos adicionais
- Criar um cluster de banco de dados

- Criar uma instância de banco de dados primária (de gravador)
- Configurações de clusters de bancos de dados do Aurora
- Configurações não aplicáveis ao Amazon Aurora para clusters de banco de dados
- Configurações não aplicáveis a instâncias de banco de dados do Amazon Aurora

Pré-requisitos do cluster de banco de dados

> **Important**
>
> Antes de criar um cluster de bancos de dados do Aurora, você deve concluir as tarefas no Configuração de seu ambiente para Amazon Aurora.

As etapas a seguir são pré-requisitos que devem ser concluídos antes da criação de um cluster de banco de dados.

Tópicos

- Configurar a rede para o cluster de banco de dados
- Pré-requisitos adicionais

Configurar a rede para o cluster de banco de dados

Um cluster de banco de dados do Amazon Aurora pode ser criado apenas em uma nuvem privada virtual (VPC) baseada no serviço Amazon VPC em uma região Região da AWS que tenha pelo menos duas zonas de disponibilidade. O grupo de sub-redes do banco de dados escolhido para o cluster de banco de dados deve incluir pelo menos duas zonas de disponibilidade. Essa configuração garante que o cluster de banco de dados tenha sempre pelo menos uma instância de banco de dados disponível para failover, no caso improvável de ocorrer uma falha em uma zona de disponibilidade.

Se você planeja configurar a conectividade entre o novo cluster de banco de dados e uma instância do EC2 na mesma VPC, pode fazer isso durante a criação do cluster. Se você planeja se conectar ao cluster de banco de dados usando recursos que não sejam instâncias do EC2 na mesma VPC, pode configurar as conexões de rede manualmente.

Tópicos

- Configurar a conectividade automática de rede com uma instância do EC2
- Configurar a rede manualmente

Configurar a conectividade automática de rede com uma instância do EC2

Ao criar um cluster de banco de dados do Aurora, use o Console de gerenciamento da AWS para configurar a conectividade entre uma instância do Amazon EC2 e o novo cluster. Quando você faz isso, o RDS configura automaticamente as definições de VPC e rede. O cluster de banco de dados é criado na mesma VPC da instância do EC2 para que a instância do EC2 possa acessar o cluster de banco de dados.

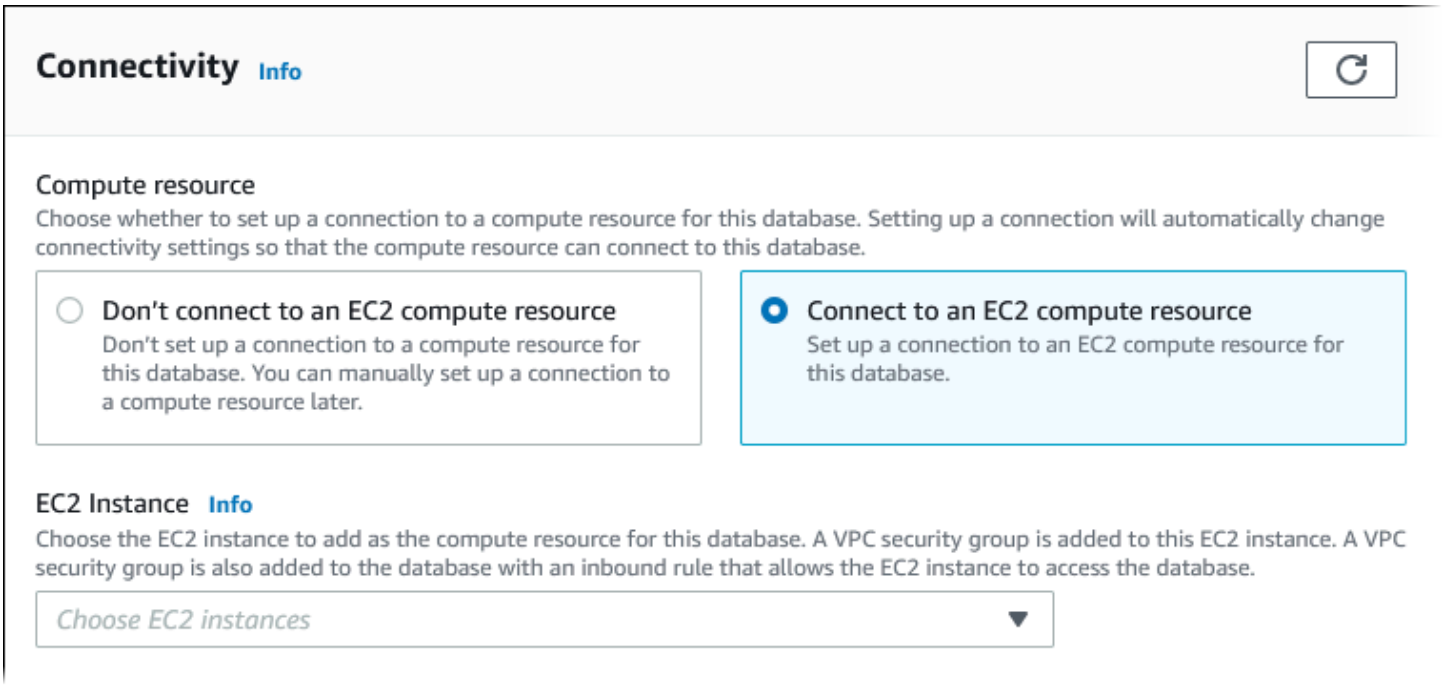
Confira a seguir os requisitos para conectar uma instância do EC2 ao cluster de banco de dados:

- A instância do EC2 deve existir na Região da AWS antes de criar o cluster de banco de dados.

  Se não houver nenhuma instância do EC2 na Região da AWS, o console fornecerá um link para que você crie uma.
- No momento, o cluster de banco de dados não pode ser um cluster de banco de dados do Aurora Serverless ou parte de um banco de dados global do Aurora.
- O usuário que está criando a instância de banco de dados deve ter permissões para realizar as seguintes operações:
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Usar essa opção cria um cluster de banco de dados privado. O cluster de banco de dados usa um grupo de sub-redes de banco de dados somente com sub-redes privadas para restringir o acesso aos recursos da VPC.

Para conectar uma instância do EC2 ao cluster de banco de dados, escolha Connect to an EC2 compute resource (Conectar-se a um recurso de computação do EC2) na seção Connectivity (Conectividade) da página Create database (Criar banco de dados).



Quando você escolhe Connect to an EC2 compute resource (Conectar-se a um recurso de computação do EC2), o RDS define as opções a seguir automaticamente. Você não pode alterar essas configurações, a menos que opte por não configurar a conectividade com uma instância do EC2 escolhendo Don't connect to an EC2 compute resource (Não conectar a um recurso de computação do EC2).

Opção do console	Configuração automática
Tipo de rede	O RDS define o tipo de rede como IPv4. No momento, o modo de pilha dupla não é compatível quando você configura uma conexão entre uma instância do EC2 e o cluster de banco de dados.
Nuvem privada virtual (VPC)	O RDS define a VPC como aquela associada à instância do EC2.

Opção do console	Configuração automática
Grupo de sub-rede de banco de dados	O RDS requer um grupo de sub-redes de banco de dados com uma sub-rede privada na mesma zona de disponibilidade da instância do EC2. Se existir um grupo de sub-redes de banco de dados que atenda a esse requisito, o RDS usará o grupo de sub-redes de banco de dados existente. Por padrão, essa opção está definida como Automatic setup (Configuração automática). Quando você escolhe Automatic setup (Configuração automátic a) e não há nenhum grupo de sub-redes de banco de dados que atenda a esse requisito, ocorre a ação a seguir. O RDS usa três sub-redes privadas disponíveis em três zonas de disponibi lidade, das quais uma é a mesma da instância do EC2. Se não houver uma sub-rede privada disponível em uma zona de disponibilidade, o RDS criará uma sub-rede privada na zona de disponibilidade. O RDS cria o grupo de sub-redes de banco de dados. Quando houver uma sub-rede privada disponível, o RDS usará a tabela de rotas associada a ela e adicionará todas as sub-redes que criar a essa tabela de rotas. Quando não houver nenhuma sub-rede privada disponível, o RDS criará uma tabela de rotas sem acesso ao gateway da Internet e adicionará as sub-redes que criar à tabela de rotas. O RDS também permite que você use grupos de sub-redes de banco de dados existentes. Selecione Choose existing (Selecion ar existente) se quiser usar um grupo de sub-redes de banco de dados existente de sua escolha.
Acesso público	O RDS escolhe No (Não) para que o cluster de banco de dados não fique publicamente acessível. Por motivos de segurança, é uma prática recomendada manter o banco de dados privado e garantir que ele não possa ser acessado pela Internet.

Opção do console	Configuração automática
VPC security group (firewall) [Grupo de segurança da VPC (firewall)	O RDS cria um grupo de segurança associado ao cluster de banco de dados. O grupo de segurança é chamado de `rds-ec2-`*n*, em que *n* é um número. Esse grupo de segurança inclui uma regra de entrada com o grupo de segurança da VPC do EC2 (firewall) como origem. Esse grupo de segurança associado ao cluster de banco de dados permite que a instância do EC2 acesse o cluster de banco de dados. O RDS também cria um grupo de segurança associado à instância de banco de dados. O grupo de segurança é chamado de `ec2-rds-`*n*, em que *n* é um número. Esse grupo de segurança inclui uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem. Esse grupo de segurança permite que a instância do EC2 envie tráfego ao cluster de banco de dados. Para adicionar outro novo grupo de segurança, escolha Create new (Criar novo) e digite o nome do novo grupo de segurança. Para adicionar grupos de segurança existentes, escolha Choose existing (Escolher existente) e selecione os grupos de segurança que deseja adicionar.

Opção do console	Configuração automática
Zona de disponibilidade	Quando você não cria uma réplica do Aurora em Availability & durability (Disponibilidade e durabilidade) durante a criação do cluster de banco de dados (implantação single-AZ), o RDS escolhe a zona de disponibilidade da instância do EC2. Quando você cria uma réplica do Aurora durante a criação do cluster de banco de dados (implantação multi-AZ), o RDS escolhe a zona de disponibilidade da instância do EC2 para uma instância de banco de dados no cluster de banco de dados. O RDS escolhe aleatoriamente uma zona de disponibilidade diferente para a outra instância de banco de dados no cluster de banco de dados. A instância de banco de dados primária ou a réplica do Aurora é criada na mesma zona de disponibilidade da instância do EC2. Existe a possibilidade de custos entre zonas de disponibilidade se ocorrer um failover e a instância de banco de dados gravadora estiver em uma zona de disponibilidade diferente.

Para ter mais informações sobre essas configurações, consulte Configurações de clusters de bancos de dados do Aurora.

Se você fizer alguma alteração nessas configurações após a criação do cluster de banco de dados, as alterações poderão afetar a conexão entre a instância do EC2 e o cluster de banco de dados.

Configurar a rede manualmente

Se você planeja se conectar ao cluster de banco de dados usando recursos que não sejam instâncias do EC2 na mesma VPC, pode configurar as conexões de rede manualmente. Se você usar o Console de gerenciamento da AWS para criar o cluster de banco de dados, o Amazon RDS poderá criar automaticamente uma VPC para você. Ou você poderá usar a VPC existente ou criar uma nova VPC para o cluster de bancos de dados do Aurora. Seja qual for a abordagem escolhida, a VPC deve ter pelo menos uma sub-rede em pelo menos duas das zonas de disponibilidade para você usá-la com um cluster de banco de dados do Amazon Aurora.

Por padrão, o Amazon RDS cria automaticamente a instância de banco de dados primária e a réplica do Aurora nas zonas de disponibilidade para você. Para escolher uma zona de disponibilidade

específica, é necessário alterar a configuração de implantação multi-AZ Availability & durability (Disponibilidade e durabilidade) para Don't create an Aurora Replica (Não criar uma réplica do Aurora). Essa ação exibe uma configuração de Availability Zone (Zona de disponibilidade) que permite escolher entre as zonas de disponibilidade em sua VPC. No entanto, recomendamos manter a configuração padrão e permitir que o Amazon RDS crie uma implantação multi-AZ e escolha as zonas de disponibilidade para você. Ao fazer isso, o cluster de banco de dados do Aurora é criado com os recursos de failover rápido e alta disponibilidade, que são dois dos principais benefícios do Aurora.

Se você não tiver uma VPC padrão, ou não tiver criado uma VPC, o Amazon RDS poderá criar automaticamente uma VPC para você ao criar um cluster de bancos de dados usando o console. Caso contrário, você deverá fazer o seguinte:

- Crie uma VPC com com no mínimo uma sub-rede em pelo menos duas das zonas de disponibilidade na Região da AWS em que você deseja implantar o cluster de banco de dados. Para obter mais informações, consulte Trabalhar com um cluster de banco de dados em uma VPC e Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).
- Especifique um grupo de segurança da VPC que autorize conexões ao seu cluster de banco de dados. Para obter mais informações, consulte Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança e Controlar acesso com grupos de segurança.
- Especifique um grupo de sub-redes de banco de dados do RDS que defina pelo menos duas sub-redes na VPC que possam ser usadas pelo cluster de banco de dados. Para obter mais informações, consulte Trabalhar com grupos de sub-redes de banco de dados.

Para obter informações sobre as VPCs, consulte Amazon VPC e Amazon Aurora. Para obter um tutorial que configura a rede para um cluster de banco de dados privado, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

Se você quiser se conectar a um recurso que não esteja na mesma VPC do cluster de banco de dados Aurora, veja os cenários apropriados em Cenários para acessar um cluster de banco de dados em uma VPC.

Pré-requisitos adicionais

Antes de criar o cluster de banco de dados multi-AZ, considere os seguintes pré-requisitos adicionais:

- Se estiver se conectando à AWS usando credenciais do AWS Identity and Access Management (IAM), sua conta da AWS deverá ter políticas do IAM que concedam as permissões necessárias para executar operações do Amazon RDS. Para obter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

  Se você estiver usando o IAM para acessar o console do Amazon RDS, primeiro será necessário fazer login no Console de gerenciamento da AWS com suas credenciais de usuário. Depois, acesse o console do Amazon RDS em https://console.aws.amazon.com/rds/.
- Se quiser personalizar os parâmetros de configuração do cluster de banco de dados, você deverá especificar um grupo de parâmetros do cluster de banco de dados e um grupo de parâmetros de banco de dados com as configurações de parâmetro obrigatórias. Para obter informações sobre como criar ou modificar um grupo de parâmetros do cluster de banco de dados ou um grupo de parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.
- Determine o número de porta de TCP/IP a ser especificado para seu cluster de banco de dados. Em algumas empresas, os firewalls bloqueiam conexões com as portas padrão (3306 para MySQL, 5432 para PostgreSQL) do Aurora. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o cluster de banco de dados. Todas as instâncias em um cluster de banco de dados usam a mesma porta.
- Se a versão principal do mecanismo do banco de dados tiver atingido a data de término do suporte padrão do RDS, você deverá usar a opção Suporte estendido da CLI ou o parâmetro de API do RDS. Consulte mais informações em “RDS Extended Support” no Configurações de clusters de bancos de dados do Aurora.

Criar um cluster de banco de dados

Você pode criar um novo cluster de bancos de dados Aurora usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

É possível criar um cluster de banco de dados usando o Console de gerenciamento da AWS com a opção Fácil de criar habilitada ou não. Com a Easy create (Criação fácil) habilitada, você especifica apenas o tipo de mecanismo de banco de dados, o tamanho da instância de banco de dados e o identificador da instância de banco de dados. A Easy create (Criação fácil) usa a configuração padrão para outras opções de configuração. Com a Easy create (Criação fácil) desabilitada, você especifica mais opções de configuração ao criar um banco de dados, incluindo as de disponibilidade, segurança, backups e manutenção.

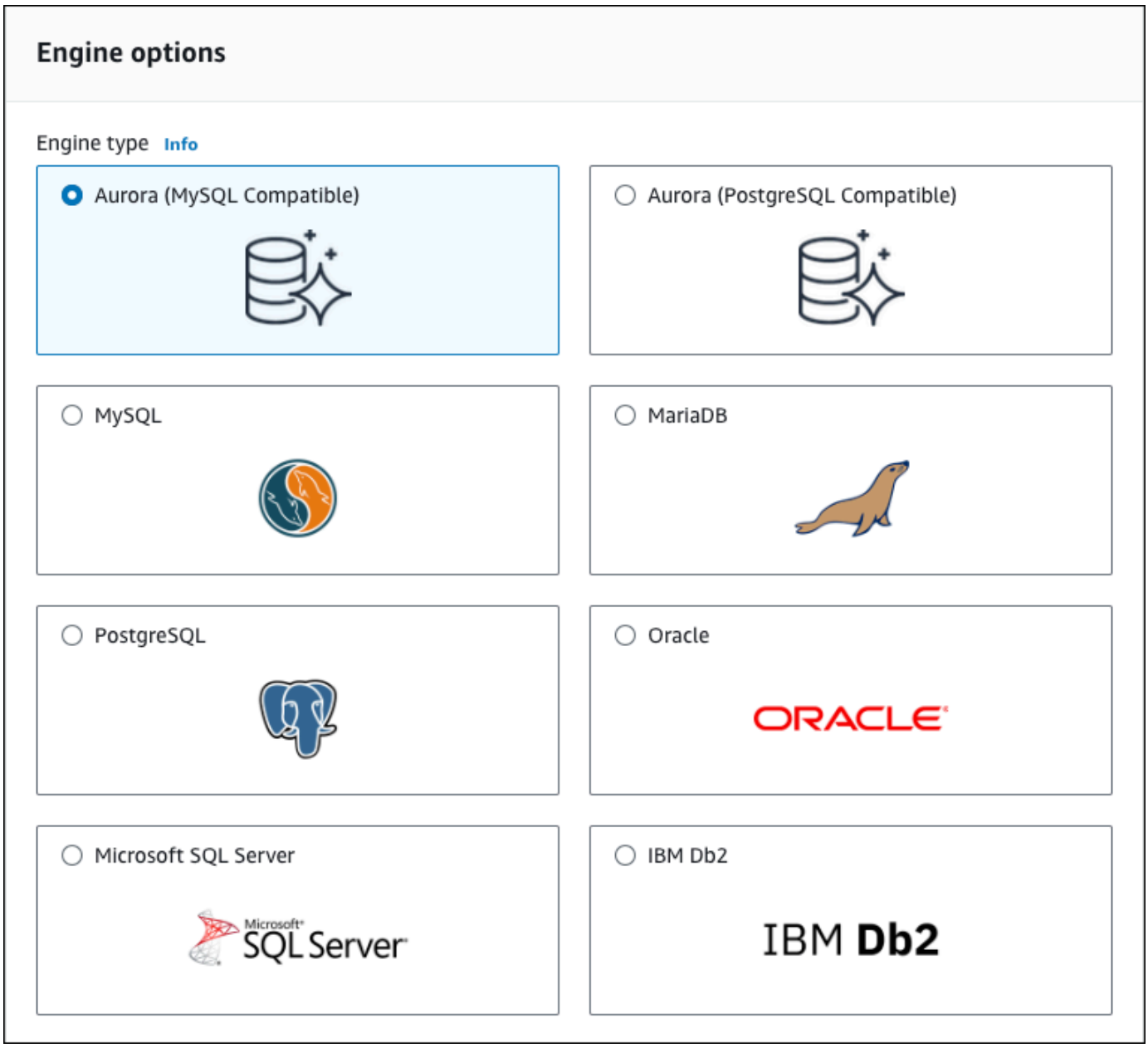
> **Note**
>
> Para este exemplo, a opção Standard Create (Criação padrão) está habilitada e a Easy Create (Criação fácil) não está habilitada. Para ter informações sobre como criar um cluster de banco de dados com a opção Fácil de criar habilitada, consulte Conceitos básicos do Amazon Aurora.

Como criar um cluster de bancos de dados do Aurora usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região da AWS em que você deseja criar o cluster de banco de dados.

   O Aurora não está disponível em todas as regiões da AWS. Para obter uma lista das regiões da AWS em que o Aurora esteja disponível, consulte Disponibilidade de regiões.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Selecione Criar banco de dados.
5. Em Escolher um método de criação de banco de dados, escolha Criação padrão.
6. Em Tipo de mecanismo, selecione uma das seguintes opções:

   - Aurora (compatível com MySQL)
   - Aurora (compatível com PostgreSQL)



7. Escolha a Versão do mecanismo.

   Para obter mais informações, consulte Versões do Amazon Aurora. Você pode usar os filtros para escolher versões compatíveis com os recursos que você deseja, como Aurora Serverless v2. Para obter mais informações, consulte Usar o Aurora Serverless v2.

8. Em Templates (Modelos), escolha o modelo que corresponde ao seu caso de uso.
9. Para inserir sua senha mestre, faça o seguinte:

   a. Na seção Configurações, abra Configurações de credencial.

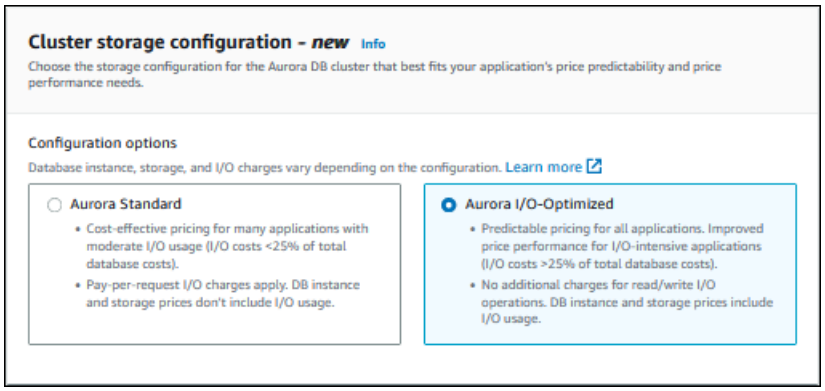
b. Desmarque a caixa de seleção Auto generate a password (Gerar uma senha automaticamente).

c. (Opcional) Altere o valor Master username (Nome do usuário principal) e insira a mesma senha em Master password (Senha mestre) e Confirm password (Confirmar senha).

Por padrão, a nova instância de banco de dados usa uma senha gerada automaticamente para o usuário mestre.

10. Na seção Conectividade em Grupo de segurança da VPC (firewall), se você selecionar Criar, um grupo de segurança da VPC será criado com uma regra de entrada que permite que o endereço IP do computador local acesse o banco de dados.

11. Para Configuração de armazenamento do cluster, escolha Aurora I/O-Optimized ou Aurora Standard. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.



12. (Opcional) Configure uma conexão com um recurso de computação para esse cluster de banco de dados.

Você pode configurar a conectividade entre uma instância do Amazon EC2 e o novo cluster de banco de dados durante a criação do cluster de banco de dados. Para obter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2.

13. Nas seções restantes, especifique suas configurações de cluster de banco de dados. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

14. Escolha Create database (Criar banco de dados).

Se você optar por usar uma senha gerada automaticamente, o botão View credential details (Visualizar detalhes da credencial) será exibido na página Databases (Bancos de dados).

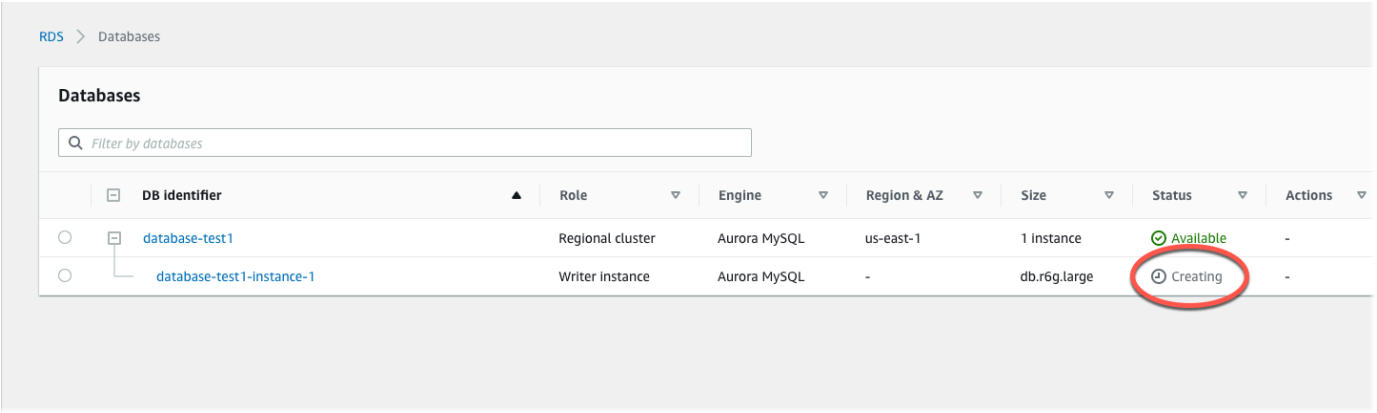
Para visualizar o nome do usuário principal e a senha do cluster de banco de dados, escolha View credential details (Ver detalhes da credencial).

Para se conectar à instância de banco de dados como o usuário mestre, use o nome de usuário e a senha exibidos.

> **Important**
>
> Você não pode visualizar a senha do usuário principal novamente. Caso você não a registre, talvez seja necessário alterá-la. Se for necessário alterar a senha do usuário mestre depois que a instância de banco de dados estiver disponível, será possível modificar a instância de banco de dados para fazer isso. Para obter mais informações sobre a modificação de uma instância de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

15. Em Databases (Bancos de dados), escolha o nome do novo cluster de bancos de dados do Aurora.

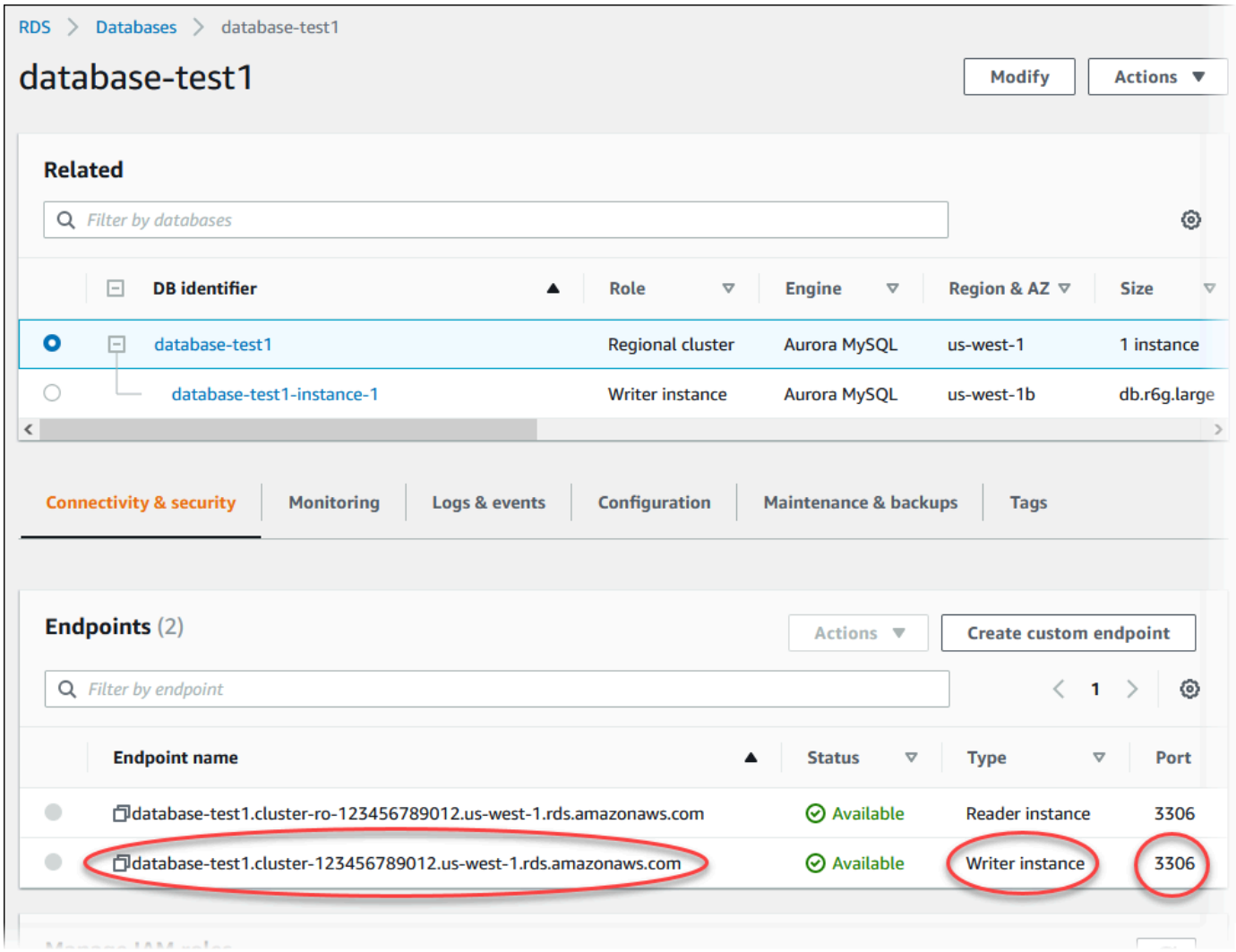
No console do RDS, os detalhes do novo cluster de banco de dados são exibidos. O cluster de banco de dados e sua instância de banco de dados têm um status creating (criando) até que o cluster de banco de dados esteja pronto para uso.



Quando o status muda para available (disponível) para ambos, você pode se conectar ao cluster de banco de dados. Dependendo da classe da instância de banco de dados e da quantidade de armazenamento, pode levar até 20 minutos para que o novo cluster de banco de dados esteja disponível.

Para visualizar o cluster recém-criado, selecione Databases (Bancos de dados) no painel de navegação no console do Amazon RDS. Depois, escolha o cluster de banco de dados para

mostrar os detalhes dele. Para obter mais informações, consulte Visualizar um cluster de bancos de dados Amazon Aurora.



Na guia Connectivity & security (Conectividade e segurança), observe a porta e o endpoint da instância de banco de dados do gravador. Use o endpoint e a porta do cluster nas suas strings de conexão JDBC e ODBC para qualquer aplicação que realize operações de gravação ou leitura.

AWS CLI

> **Note**
>
> Antes de criar um cluster de bancos de dados do Aurora usando a AWS CLI, é necessário cumprir os pré-requisitos necessários, como criar uma VPC e um grupo de sub-redes de

banco de dados do RDS. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.

Você também pode usar a AWS CLI para criar um cluster de bancos de dados Aurora MySQL ou do Aurora PostgreSQL.

Como criar um cluster de bancos de dados do Aurora MySQL usando a AWS CLI

Ao criar um cluster ou uma instância de banco de dados compatível com o Aurora MySQL 8.0 ou 5.7, especifique `aurora-mysql` para a opção `--engine`.

Execute as etapas a seguir:

1. Identifique o grupo de sub-redes do banco de dados e o ID do grupo de segurança da VPC de seu novo cluster de banco de dados e chame o comando create-db-cluster da AWS CLI para criar o cluster de bancos de dados do Aurora MySQL.

   Por exemplo, o seguinte comando cria um novo cluster de banco de dados compatível com o MySQL 8.0 chamado `sample-cluster`. O cluster usa a versão padrão do mecanismo e o tipo de armazenamento Aurora I/O-Optimized.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-mysql --engine-version 8.0 \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Para Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-mysql --engine-version 8.0 ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

O comando a seguir cria um novo cluster de banco de dados compatível com o MySQL 5.7 chamado `sample-cluster`. O cluster usa a versão padrão do mecanismo e o tipo de armazenamento Aurora Standard.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster  \
    --engine aurora-mysql --engine-version 5.7 \
    --storage-type aurora \
    --master-username user-name --manage-master-user-password \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster sample-cluster  ^
    --engine aurora-mysql --engine-version 5.7 ^
    --storage-type aurora ^
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Se você usar o console para criar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a AWS CLI para criar um cluster de banco de dados, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados. Até você criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `Creating`.

   Chame o comando create-db-instance da AWS CLI para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados como o valor da opção `--db-cluster-identifier`.

   > **Note**
   >
   > Não é possível definir a opção `--storage-type` para instâncias de banco de dados. Você pode configurá-la somente para clusters de banco de dados.

   Por exemplo, o comando a seguir cria uma nova instância de banco de dados compatível com o MySQL 5.7 ou o MySQL 8.0 chamada `sample-instance`.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Para criar um cluster de bancos de dados do Aurora PostgreSQL usando a AWS CLI

1. Identifique o grupo de sub-redes do banco de dados e o ID do grupo de segurança da VPC de seu novo cluster de banco de dados e chame o comando create-db-cluster da AWS CLI para criar o cluster de bancos de dados do Aurora PostgreSQL.

   Por exemplo, o seguinte comando cria um novo cluster de banco de dados chamado `sample-cluster`. O cluster usa a versão padrão do mecanismo e o tipo de armazenamento Aurora I/O-Optimized.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-postgresql \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Para Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-postgresql ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

2. Se você usar o console para criar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a AWS CLI para criar um cluster de banco de dados, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados. Até você criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `Creating`.

   Chame o comando create-db-instance da AWS CLI para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados como o valor da opção `--db-cluster-identifier`.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance \
        --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
   instance-class db.r5.large
   ```

   Para Windows:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance ^
        --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
   instance-class db.r5.large
   ```

Estes exemplos especificam a opção `--manage-master-user-password` para gerar a senha mestra do usuário e gerenciá-la no Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

API do RDS

> **Note**
>
> Antes de criar um cluster de bancos de dados Aurora usando a AWS CLI, é necessário cumprir com os pré-requisitos necessários, como criar uma VPC e um grupo de sub-redes de banco de dados do RDS. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.

Identifique o grupo de sub-redes do banco de dados e o ID do grupo de segurança da VPC de seu novo cluster de banco de dados. Depois, chame a operação CreateDBInstance para criar o cluster de banco de dados.

Ao criar um cluster ou uma instância de banco de dados do Aurora MySQL versão 2 ou 3, especifique `aurora-mysql` para o parâmetro `Engine`.

Ao criar um cluster de banco de dados ou instância de bancos de dados do Aurora PostgreSQL, especifique `aurora-postgresql` para o parâmetro `Engine`.

Se você usar o console para criar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a API do RDS para restaurar um cluster de banco de dados, será necessário criar explicitamente a instância primária para o cluster de banco de dados usando CreateDBInstance. A instância primária é a primeira instância criada em um cluster de banco de dados. Até você criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `Creating`.

Criar uma instância de banco de dados primária (de gravador)

Se você usar o Console de gerenciamento da AWS para criar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância primária (de gravador) para o cluster de banco de dados. Se você usar a AWS CLI ou a API do RDS para criar um cluster de banco de dados, deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados. Até você criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `Creating`.

Para obter mais informações, consulte Criar um cluster de banco de dados.

> **Note**
>
> Se você tiver um cluster de banco de dados sem uma instância de banco de dados de gravador, também chamado de cluster headless, não poderá usar o console para criar uma instância de gravador. Use a AWS CLI ou a API do RDS.

O exemplo a seguir usa o comando create-db-instance da AWS CLI para criar uma instância de gravador para um cluster de banco de dados do Aurora PostgreSQL chamado `headless-test`.

```
aws rds create-db-instance \
    --db-instance-identifier no-longer-headless \
```

```
    --db-cluster-identifier headless-test \
    --engine aurora-postgresql \
    --db-instance-class db.t4g.medium
```

Configurações de clusters de bancos de dados do Aurora

A tabela a seguir contém detalhes sobre as configurações que você escolhe quando cria um cluster de banco de dados do Aurora.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Atualização da versão secundária automática	Escolha Enable auto minor version upgrade (Habilitar atualização automática de versão secundária) se você quiser permitir que o cluster de bancos de dados do Aurora receba automaticamente as atualizações de versão secundária preferida no mecanismo de banco de dados quando disponíveis. A configuração Atualização automática de versão secundária se aplica aos clusters de bancos de dados do Aurora PostgreSQL e do Aurora MySQL. Para obter mais informações sobre atualizações de mecanismos para o Aurora PostgreSQL, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL. Para obter mais informações sobre atualizações de mecanismos	Defina esse valor para cada instância de banco de dados em seu cluster do Aurora. Se qualquer instância de banco de dados em seu cluster tiver essa configuração desativada, o cluster não será atualizado automaticamente. Usando a AWS CLI, execute `create-db-instance` e defina a opção `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `AutoMinorVersionUpgrade`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
	para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.	
AWS KMS key	Disponível apenas quando Encryption (Criptografia) estiver definido como Enable encryption (Habilitar criptografia). Escolha a AWS KMS key a ser usada para criptografar esse cluster de banco de dados. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--kms-key-id`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `KmsKeyId`.
Retrocesso	Aplica-se somente ao Aurora MySQL. Escolha Enable Backtrack (Habilitar retrocesso) para habilitar o retrocesso ou Disable Backtrack (Desabilitar retrocesso) para desabilitá-lo. Usando o retrocesso, você pode retroceder um cluster de banco de dados a um período específico, sem criar um novo cluster de banco de dados. O recurso é desabilitado por padrão. Se você habilitar o retrocesso, especifique também a quantidade de tempo que deseja retroceder o cluster de banco de dados (a janela do retrocesso de destino). Para obter mais informações, consulte Retroceder um cluster de banco de dados Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--backtrack-window`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `BacktrackWindow`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Autoridade certificadora	A autoridade de certificação (CA) para o certificado do servidor usado pelas instâncias de banco de dados no cluster de banco de dados. Para obter mais informações, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.	Usando a AWS CLI, execute `create-db-instance` e defina a opção `--ca-certificate-identifier`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `CACertificateIdentifier`.
Configuração de armazenamento do cluster	O tipo de armazenamento do cluster de banco de dados: Aurora I/O-Optimized ou Aurora Standard. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--storage-type`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `StorageType`.
Copiar tags para snapshots	Escolha esta opção para copiar qualquer tag da instância de banco de dados para um snapshot de banco de dados quando você cria um snapshot. Para obter mais informações, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--copy-tags-to-snapshot | --no-copy-tags-to-snapshot`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `CopyTagsToSnapshot`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Autenticação de banco de dados	A autenticação de banco de dados que você deseja usar. Para MySQL: • Selecione Password authentication (Autenticação de senha) para autenticar usuários de banco de dados somente com senhas de banco de dados. • Selecione Password and IAM database authentication (Senha e autenticação de banco de dados do IAM) para autenticar usuários de banco de dados com senhas de banco de dados e credenciais de usuário por meio de funções e usuários do IAM. Para obter mais informações, consulte Autenticação do banco de dados do IAM. Para PostgreSQL: • Selecione IAM database authentication (Autenticação de banco de dados do IAM) para autenticar usuários de banco de dados com senhas de banco de dados e credenciais de usuário por meio de usuários e perfis. Para obter mais informações, consulte Autenticação do banco de dados do IAM.	Para usar a autenticação de banco de dados do IAM com a AWS CLI, execute `create-db-cluster` e defina a opção `--enable-iam-database-authentication | --no-enable-iam-database-authentication`. Para usar a autenticação de banco de dados do IAM com a API do RDS, chame `CreateDBCluster` e defina o parâmetro `EnableIAMDatabaseAuthentication`. Para usar a autenticação Kerberos com a AWS CLI, execute `create-db-cluster` e defina as opções `--domain` e `--domain-iam-role-name`. Para usar a autenticação Kerberos com a API do RDS, chame `CreateDBCluster` e defina os parâmetros `Domain` e `DomainIAMRoleName`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
	• Escolha Kerberos authentication (Autenticação do Kerberos) para autenticar senhas de banco de dados e credenciais de usuário usando a autenticação do Kerberos. Para obter mais informações, consulte Usar a autenticação Kerberos com o Aurora PostgreSQL.	
Porta de banco de dados	Especifique a porta que as aplicações e os utilitários usam para acessar o banco de dados. Os clusters de bancos de dados do Aurora MySQL assumem como padrão a porta do MySQL padrão, 3306. Os clusters de bancos de dados do Aurora PostgreSQL assumem como padrão a porta do PostgreSQL padrão, 5432. Em algumas empresas, os firewalls bloqueiam conexões com essas portas padrão. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o novo cluster de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--port`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `Port`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Identificador do cluster de banco de dados	Insira um nome para o cluster de banco de dados exclusivo da sua conta na região da AWS escolhida. Esse identificador é usado no endereço do endpoint do cluster para o seu cluster de banco de dados. Para obter informaçõ es sobre o endpoint do cluster, consulte Conexões de endpoints do Amazon Aurora. O identificador do cluster de banco de dados tem as seguintes restrições: • Deve conter de 1 a 63 caractere s alfanuméricos ou hífens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todos os clusters de banco de dados por conta da AWS, por região da AWS.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--db-cluster-identifier` . Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `DBClusterIdentifier` .

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Grupo de parâmetros do cluster de banco de dados	Escolha um grupo de parâmetros de cluster de banco de dados. O Aurora conta com um grupo de parâmetros de cluster de banco de dados que você pode usar, ou é possível criar seu próprio grupo de parâmetros. Para obter mais informações sobre os grupos de parâmetros do cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--db-cluster-parameter-group-name`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `DBClusterParameterGroupName`.
Classe de instância de banco de dados	Aplica-se apenas ao tipo de capacidade provisionada. Selecione uma classe de instância de banco de dados que defina os requisitos de processamento e de memória de cada instância no cluster de banco de dados. Para obter mais informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.	Defina esse valor para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina a opção `--db-instance-class`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `DBInstanceClass`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Grupo de parâmetros de banco de dados	Escolha um grupo de parâmetros. O Aurora conta com um grupo de parâmetros padrão que você pode usar, ou você pode criar seu próprio grupo de parâmetros. Para obter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.	Defina esse valor para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina a opção `--db-parameter-group-name`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `DBParameterGroupName`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Grupo de sub-rede de banco de dados	O grupo de sub-redes de banco de dados que você deseja usar para o cluster de banco de dados. Selecione Choose existing (Selecionar existente) para usar um grupo de sub-redes de banco de dados existente . Depois, escolha o grupo de sub-redes necessário na lista suspensa Existing DB subnet groups (Grupos de sub-redes de banco de dados existentes). Escolha Automatic setup (Configuração automática) para permitir que o RDS selecione um grupo de sub-redes de banco de dados compatível. Se não existir nenhum, o RDS criará um grupo de sub-redes para o cluster. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--db-subnet-group-name` . Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `DBSubnetGroupName` .

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Habilitar proteção contra exclusão	Escolha Enable deletion protection (Habilitar proteção contra exclusão) para impedir que seu cluster de banco de dados seja excluído. Por padrão, se você criar um cluster de banco de dados de produção com o console, a proteção contra exclusão será habilitada.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--deletion-protection | --no-deletion-protection`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `DeletionProtection`.
Habilitar criptografia	Selecione `Enable encryption` para habilitar a criptografia em repouso para esse cluster de banco de dados. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--storage-encrypted | --no-storage-encrypted`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `StorageEncrypted`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Habilitar monitoramento avançado	Selecione Enable enhanced monitoring (Habilitar monitoramento aprimorado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.	Defina esses valores para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina as opções `--monitoring-interval` e `--monitoring-role-arn`. Usando a API do RDS, chame `CreateDBInstance` e defina os parâmetros `MonitoringInterval` e `MonitoringRoleArn`.
Habilitar a API de dados do RDS	Selecione Habilitar a API de dados do RDS para habilitar a API de dados do RDS (API de dados). A API de dados fornece um endpoint HTTP seguro para executar declarações SQL sem gerenciar conexões. Para obter mais informações, consulte Usar a API de dados do Amazon RDS.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--enable-http-endpoint | --no-enable-http-endpoint`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `EnableHttpEndpoint`.
Tipo de mecanismo	Escolha o mecanismo de banco de dados a ser usado para este cluster de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--engine`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `Engine`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Versão do mecanismo	Aplica-se apenas ao tipo de capacidade provisionada. Escolha o número da versão de seu mecanismo de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--engine-version`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `EngineVersion`.
Prioridade de failover	Escolha uma prioridade de failover para a instância. Se você não selecionar um valor, o padrão será tier-1. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Para obter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.	Defina esse valor para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina a opção `--promotion-tier`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `PromotionTier`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Nome do banco de dados inicial	Insira um nome para seu banco de dados padrão. Se não fornecer um nome para um cluster de bancos de dados do Aurora MySQL, Amazon RDS não criará um banco de dados no cluster de banco de dados que estiver criando. Se você não fornecer um nome para um cluster de bancos de dados Aurora PostgreSQL, Amazon RDS criar um banco de dados chamado `postgres`. Para o Aurora MySQL, o nome do banco de dados padrão tem estas restrições: • Deve conter de 1 a 64 caracteres alfanuméricos. • Não pode ser uma palavra reservada pelo mecanismo de banco de dados. Para o Aurora PostgreSQL, o nome do banco de dados padrão tem estas restrições: • Deve conter de 1 a 63 caracteres alfanuméricos. • Deve começar com uma letra. Os caracteres subsequentes podem ser letras, sublinhados ou dígitos (0 a 9).	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--database-name` . Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `DatabaseName` .

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
	• Não pode ser uma palavra reservada pelo mecanismo de banco de dados. Para criar bancos de dados adicionais, conecte-se ao cluster de banco de dados e use o comando CREATE DATABASE do SQL. Para obter mais informações sobre como se conectar ao cluster de banco de dados, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.	
Exportações de log	Na seção Log exports (Exportações de log), escolha os logs que deseja começar a publicar no Amazon CloudWatch Logs. Para obter mais informações sobre publicação de logs do Aurora MySQL no CloudWatch Logs, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs. Para obter mais informações sobre publicação de logs do Aurora PostgreSQL no CloudWatch Logs, consulte Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--enable-cloudwatch-logs-exports`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `EnableCloudwatchLogsExports`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Janela de manutenção	Escolha Select window (Selecionar janela) e especifique o período semanal durante o qual a manutenção do sistema pode ser realizada. Ou selecione No preference (Sem preferência) para que o Amazon RDS atribua um período aleatoriamente.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--preferred-maintenance-window`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `PreferredMaintenanceWindow`.
Gerenciar credenciais principais no AWS Secrets Manager	Selecione Gerenciar credenciais principais no AWS Secrets Manager para gerenciar a senha do usuário principal em um segredo no Secrets Manager. Opcionalmente, selecione uma chave do KMS a ser usada para proteger o segredo. Escolha entre uma das chaves do KMS da sua conta ou insira a chave de uma conta distinta. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.	Usando a AWS CLI, execute `create-db-cluster` e defina as opções `--manage-master-user-password | --no-manage-master-user-password` e `--master-user-secret-kms-key-id`. Usando a API do RDS, chame `CreateDBCluster` e defina os parâmetros `ManageMasterUserPassword` e `MasterUserSecretKmsKeyId`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Senha mestre	Insira uma senha para fazer login no cluster de banco de dados: • Para o Aurora MySQL, a senha deve conter 8 a 41 caracteres ASCII imprimíveis. • Para o Aurora PostgreSQL, ela deve conter de 8 a 99 caracteres ASCII imprimíveis. • Não pode conter /, ", @ ou um espaço.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--master-user-password`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `MasterUserPassword`.
Nome do usuário principal	Insira um nome a ser usado como o nome do usuário mestre para fazer login no cluster de banco de dados: • Para o Aurora MySQL, o nome deve conter 1 a 16 caracteres alfanuméricos. • Para o Aurora PostgreSQL, deve conter de 1 a 63 caracteres alfanuméricos. • O primeiro caractere deve ser uma letra. • O nome não pode ser uma palavra reservada pelo mecanismo de banco de dados. Você não pode alterar o nome do usuário principal após a criação do cluster de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--master-username`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `MasterUsername`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Implantação multi-AZ	Aplica-se apenas ao tipo de capacidade provisionada. Determine se você deseja criar réplicas do Aurora em outras Zonas de disponibilidade para suporte a failover. Se você escolher Create Replica in Different Zone (Criar réplica em outra zona), o Amazon RDS criará uma réplica do Aurora para você no cluster de banco de dados, em uma zona de disponibilidade diferente da instância primária do cluster. Para obter mais informações sobre várias Zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--availability-zones`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `AvailabilityZones`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Tipo de rede	Os protocolos de endereçamento IP compatíveis com o cluster de banco de dados. IPv4. para especificar que os recursos podem se comunicar com o cluster de banco de dados somente por meio do protocolo de endereçamento IPv4. Dual-stack mode (Modo de pilha dupla), para especificar que os recursos podem se comunicar com o cluster de banco de dados por IPv4, IPv6 ou ambos. Use o modo de pilha dupla se você tiver algum recurso que precise se comunicar com o cluster de banco de dados pelo protocolo de endereçamento IPv6. Para usar o modo de pilha dupla, pelo menos duas sub-redes devem abranger duas zonas de disponibilidade compatíveis com os protocolos de rede IPv4 e IPv6. Além disso, associe um bloco CIDR IPv6 às sub-redes no grupo de sub-redes de banco de dados especificado. Para obter mais informações, consulte Endereçamento IP do Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `-network-type` . Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `NetworkType` .

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Acesso público	Escolha Publicly accessible (Acessível ao público geral) para fornecer ao cluster de banco de dados um endereço IP público ou escolha Not publicly accessible (Não acessível ao público geral). As instâncias no seu cluster de banco de dados podem ser uma combinação de instâncias de banco de dados públicas e privadas. Para obter mais informações sobre como ocultar instâncias do acesso público, consulte Ocultar um cluster de banco de dados em uma VPC da Internet. Para se conectar a uma instância de banco de dados de fora de sua Amazon VPC, a instância de banco de dados deve ser acessível ao público geral, o acesso deve ser concedido usando as regras de entrada do grupo de segurança da instância de banco de dados e outros requisitos devem ser atendidos. Para obter mais informações, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS. Se sua instância de banco de dados não estiver acessível	Defina esse valor para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina a opção `--publicly-accessible | --no-publicly-accessible`. Usando a API do RDS, chame `CreateDBInstance` e defina o parâmetro `PubliclyAccessible`.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
	publicamente, também será possível usar uma conexão AWS Site-to-Site VPN ou uma conexão do Direct Connect para acessá-la de uma rede privada. Para obter mais informações, consulte Privacidade do tráfego entre redes.	

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Suporte estendido do RDS	Selecione Habilitar Suporte estendido do RDS para permitir que as versões principais do mecanismo compatíveis continuem funcionando após a data de término do suporte padrão do Aurora. Quando você cria um cluster de banco de dados, o Amazon Aurora usa como padrão o Suporte estendido do RDS. Para evitar a criação de um cluster de banco de dados após a data de fim do suporte padrão do Aurora e para evitar cobranças pelo Suporte estendido do RDS, desabilite essa configuração. Os clusters de banco de dados existentes não incorrerão em cobranças até a data de início dos preços do Suporte estendido do RDS. Para obter mais informações, consulte Suporte estendido do Amazon RDS com Amazon Aurora.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--engine-lifecycle-support` . Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `EngineLifecycleSupport` .

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
RDS Proxy	Selecione Create an RDS Proxy (Criar um proxy RDS) para criar um proxy para seu cluster de banco de dados. O Amazon RDS cria automaticamente um perfil do IAM e um segredo do Secrets Manager para o proxy. Para obter mais informações, consulte Amazon RDS Proxypara Aurora.	Não disponível ao criar um cluster de banco de dados.
Período de retenção	Escolha o tempo, de 1 a 35 dias, que o Aurora retém cópias de backup do banco de dados. As cópias de backup podem ser usadas para restaurações point-in-time (PITR) do banco de dados, contabilizando até os segundos.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--backup-retention-period`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `BackupRetentionPeriod`.
Ativar DevOps Guru	Escolha Turn on DevOps Guru (Ativar DevOps Guru) para ativar o Amazon DevOps Guru para seu banco de dados do Aurora. Para que o DevOps Guru para RDS forneça uma análise detalhada de anomalias de performance, o Performance Insights deve estar ativado. Para obter mais informações, consulte Configurar o DevOps Guru para RDS.	Você pode ativar o DevOps Guru para RDS no console do RDS, mas não use a API nem a CLI do RDS. Para obter mais informações sobre como executar o DevOps Guru, consulte o Guia do usuário do Amazon DevOps Guru.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Ativar o Performance Insights	Escolha Turn on Performance Insights (Ativar Performance Insights) para ativar o Amazon RDS Performance Insights. Para obter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.	Defina esses valores para cada instância de banco de dados em seu cluster do Aurora. Usando a AWS CLI, execute `create-db-instance` e defina as opções `--enable-performance-insights` | `--no-enable-performance-insights`, `--performance-insights-kms-key-id` e `--performance-insights-retention-period`. Usando a API do RDS, chame `CreateDBInstance` e defina os parâmetros `EnablePerformanceInsights`, `PerformanceInsightsKMSKeyId` e `PerformanceInsightsRetentionPeriod`.
Virtual Private Cloud (VPC)	Escolha a VPC que hospeda o cluster de banco de dados. Selecione Create a New VPC (Criar uma nova VPC) para fazer o Amazon RDS criar uma VPC para você. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.	Para a AWS CLI e a API, especifique os IDs do grupo de segurança da VPC.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
Grupo de segurança da VPC (firewall)	Escolha Create new (Criar novo) para fazer o Amazon RDS criar um grupo de segurança da VPC para você. Ou selecione Choose existing (Escolher existente) e especifique um ou mais grupos de segurança de VPC para proteger o acesso de rede ao cluster de banco de dados. Quando você escolhe Create new (Criar novo) no console do RDS, um novo grupo de segurança é criado com uma regra de entrada que permite acesso à instância de banco de dados pelo endereço IP detectado no navegador. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.	Usando a AWS CLI, execute `create-db-cluster` e defina a opção `--vpc-security-group-ids`. Usando a API do RDS, chame `CreateDBCluster` e defina o parâmetro `VpcSecurityGroupIds`.

Configurações não aplicáveis ao Amazon Aurora para clusters de banco de dados

As configurações a seguir no comando `create-db-cluster` da AWS CLI e na operação de API `CreateDBCluster` do RDS não se aplicam a clusters de banco de dados do Amazon Aurora.

> **Note**
>
> O Console de gerenciamento da AWS não mostra essas configurações para clusters de banco de dados do Aurora.

Configuração da AWS CLI	Configuração da API do RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`
`--publicly-accessible | --no-publicly-accessible`	`PubliclyAccessible`

Configurações não aplicáveis a instâncias de banco de dados do Amazon Aurora

As configurações a seguir no comando `create-db-instance` da AWS CLI e na operação de API `CreateDBInstance` do RDS não se aplicam a instâncias de banco de dados e ao cluster de banco de dados do Amazon Aurora.

> **Note**
>
> O Console de gerenciamento da AWS não mostra essas configurações para instâncias de banco de dados Aurora.

Configuração da AWS CLI	Configuração da API do RDS
`--allocated-storage`	`AllocatedStorage`
`--availability-zone`	`AvailabilityZone`
`--backup-retention-period`	`BackupRetentionPeriod`
`--backup-target`	`BackupTarget`
`--character-set-name`	`CharacterSetName`
`--character-set-name`	`CharacterSetName`
`--custom-iam-instance-profile`	`CustomIamInstanceProfile`
`--db-security-groups`	`DBSecurityGroups`
`--deletion-protection | --no-deletion-protection`	`DeletionProtection`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-cloudwatch-logs-exports`	`EnableCloudwatchLogsExports`
`--enable-customer-owned-ip | --no-enable-customer-owned-ip`	`EnableCustomerOwnedIp`
`--enable-iam-database-authentication | --no-enable-iam-database-authentication`	`EnableIAMDatabaseAuthentication`

Configuração da AWS CLI	Configuração da API do RDS
`--engine-version`	`EngineVersion`
`--iops`	`Iops`
`--kms-key-id`	`KmsKeyId`
`--master-username`	`MasterUsername`
`--master-user-password`	`MasterUserPassword`
`--max-allocated-storage`	`MaxAllocatedStorage`
`--multi-az | --no-multi-az`	`MultiAZ`
`--nchar-character-set-name`	`NcharCharacterSetName`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--preferred-backup-window`	`PreferredBackupWindow`
`--processor-features`	`ProcessorFeatures`
`--storage-encrypted | --no-storage-encrypted`	`StorageEncrypted`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--timezone`	`Timezone`
`--vpc-security-group-ids`	`VpcSecurityGroupIds`

Criar recursos do Amazon Aurora com o AWS CloudFormation

O Amazon Aurora é integrado ao AWS CloudFormation, um serviço que ajuda você a modelar e configurar os recursos da AWS para que você possa passar menos tempo criando e gerenciando os recursos e a infraestrutura. Você cria um modelo que descreve todos os recursos AWS desejados , (como clusters de banco de dados e grupos de parâmetros de cluster de banco de dados) e o CloudFormation provisiona e configura esses recursos para você.

Ao usar o CloudFormation, você poderá reutilizar o modelo para configurar os recursos do Aurora de forma repetida e consistente. Descreva seus recursos uma vez e, depois, provisione os mesmos recursos repetidamente em várias contas e regiões da AWS.

Aurora e modelos do CloudFormation

Modelos do CloudFormation são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os atributos que você deseja provisionar nas suas pilhas CloudFormation. Se não estiver familiarizado com o JSON ou o YAML, é possível usar o CloudFormation Designer para ajudar a começar a usar os modelos do CloudFormation. Para obter mais informações, consulte O que é o CloudFormation Designer? no Guia do usuário do AWS CloudFormation.

O Aurora é compatível com a criação de recursos no CloudFormation. Para obter mais informações, incluindo exemplos de modelos JSON e YAML para esses recursos, consulte Referência de tipo de recurso do RDS no Guia do usuário do AWS CloudFormation.

Saiba mais sobre o CloudFormation

Para saber mais sobre o CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS CloudFormation Guia do usuário do
- CloudFormation Referência de API do
- AWS CloudFormation Guia do Usuário da Interface de Linha de Comando

Como conectar-se a um cluster de bancos de dados Amazon Aurora

É possível conectar-se a um cluster de bancos de dados Aurora usando as mesmas ferramentas usadas para se conectar a um banco de dados MySQL ou PostgreSQL. Especifique uma string de conexão com qualquer script, utilitário ou aplicativo que conecte-se a uma instância de banco de dados MySQL ou PostgreSQL. É possível usar a mesma chave pública para conexões Secure Sockets Layer (SSL).

Na string de conexão, use tipicamente as informações de host e porta de endpoints especiais associados ao cluster de banco de dados. Com esses endpoints, é possível usar os mesmos parâmetros de conexão, independentemente do número de instâncias de banco de dados no cluster. É possível usar as informações de host e porta de uma instância de bancos de dados Aurora para tarefas específicas, como solução de problemas.

> **Note**
>
> Para clusters de banco de dados do Aurora Serverless, você se conecta ao endpoint do banco de dados em vez da instância de banco de dados. Você pode encontrar o endpoint do banco de dados para um cluster de banco de dados de Aurora Serverless na guia Conectividade e segurança do Console de gerenciamento da AWS. Para obter mais informações, consulte Usar o Amazon Aurora Serverless v1.

Independentemente do mecanismo de banco de dados de Aurora e das ferramentas específicas que você usa para trabalhar com o cluster de banco de dados ou instância, o endpoint deve estar acessível. Um cluster de bancos de dados do Aurora pode ser criado apenas em uma nuvem privada virtual (VPC) com base no serviço da Amazon VPC. Isso significa que você acessa o endpoint de dentro da VPC ou fora da VPC usando uma das seguintes abordagens.

- Acessar o cluster de bancos de dados do Aurora dentro da VPC: habilite o acesso ao cluster de banco de dados do Aurora por meio da VPC. Você faz isso editando as regras de entrada no grupo de segurança para VPC para permitir o acesso ao cluster de banco de dados de Aurora específico. Para saber mais, incluindo como configurar sua VPC para diferentes cenários de cluster de banco de dados de Aurora, consulte Amazon Virtual Private Cloud VPCs e Amazon Aurora.
- Acessar o cluster de banco de dados do Aurora fora da VPC: para acessar um cluster de banco de dados de Aurora fora da VPC, use o endereço de endpoint público do cluster de bancos de dados.

Para obter mais informações, consulte Solução de problemas de falha de conexão do Aurora.

Sumário

- Conectar-se aos clusters de banco de dados do Aurora com os drivers da AWS
- Como conectar-se a um cluster de bancos de dados Amazon Aurora MySQL
  - Utilitários de conexão do Aurora MySQL
  - Conectar-se ao Aurora MySQL com o utilitário MySQL
  - Conectar-se ao Aurora MySQL com o driver JDBC da Amazon Web Services (AWS)
  - Conectar-se ao Aurora MySQL com o driver Python da Amazon Web Services (AWS)
  - Conectar-se ao Aurora MySQL com o driver ODBC da Amazon Web Services (AWS) para MySQL
  - Conexão ao Aurora MySQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)
  - Conectar-se ao Aurora MySQL usando SSL
- Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL
  - Utilitários de conexão do Aurora PostgreSQL
  - Conectar-se ao Aurora PostgreSQL com o driver JDBC da Amazon Web Services (AWS)
  - Conectar-se ao Aurora PostgreSQL com o driver Python da Amazon Web Services (AWS)
  - Conexão ao Aurora PostgreSQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)
- Solução de problemas de falha de conexão do Aurora

Conectar-se aos clusters de banco de dados do Aurora com os drivers da AWS

O pacote de drivers da AWS foram projetados para comportar tempos mais rápidos de transição e de failover, além de autenticação com o AWS Secrets Manager, o AWS Identity and Access Management (IAM) e identidades federadas. Os drivers da AWS dependem do monitoramento do status do cluster de banco de dados e do conhecimento da topologia do cluster para determinar o novo gravador. Essa abordagem reduz os tempos de transição e de failover para segundos de um dígito, em comparação com dezenas de segundos para drivers de código aberto.

A tabela a seguir lista os recursos aceitos para cada um dos drivers. Como novos recursos do serviço são introduzidos, o objetivo do pacote de drivers da AWS é ter suporte integrado para esses recursos do serviço.

Recurso	Driver JDBC da AWS	Driver Python da AWS	AWS Driver ODBC da para MySQL	Wrapper avançado de NodeJS da AWS
Suporte a failover	Sim	Sim	Sim	Sim
Monitoramento aprimorado de failover	Sim	Sim	Sim	Sim
Divisão de leituras/ gravações	Sim	Sim	Não	Sim
Gerenciamento de conexões do Aurora	Sim	Sim	Não	Sim
Conexão de metadados do driver	Sim	N/D	N/D	N/D
Telemetria	Sim	Sim	Não	Sim
Secrets Manager	Sim	Sim	Sim	Sim
Autenticação do IAM	Sim	Sim	Sim	Sim
Identidades federadas (AD FS)	Sim	Sim	Não	Sim

Recurso	Driver JDBC da AWS	Driver Python da AWS	AWS Driver ODBC da para MySQL	Wrapper avançado de NodeJS da AWS
Identidade federada (Okta)	Sim	Sim	Sim	Sim
Aurora PostgreSQL Limitless Database	Sim (apenas Aurora PostgreSQL)	Não	Não	Sim (apenas Aurora PostgreSQL)

Para ter mais informações sobre os drivers da AWS, consulte o driver de linguagem correspondente para o cluster de banco de dados do Aurora MySQL ou do Aurora PostgreSQL.

Como conectar-se a um cluster de bancos de dados Amazon Aurora MySQL

Para fazer a autenticação no cluster de bancos de dados Aurora MySQL, você pode usar o nome de usuário e a autenticação de senha do MySQL ou a autenticação de banco de dados do AWS Identity and Access Management (IAM). Para obter mais informações sobre como usar o nome de usuário e a autenticação de senha no MySQL, consulte Gerenciamento de contas e controle de acesso na documentação do MySQL. Para obter mais informações sobre como usar autenticação de banco de dados do IAM, consulte Autenticação do banco de dados do IAM.

Quando você tiver uma conexão com um cluster de bancos de dados Amazon Aurora com compatibilidade do MySQL 8.0, será possível executar os comandos SQL que forem compatíveis com o MySQL versão 8.0. O MySQL 8.0.23 é a versão mínima compatível. Para obter mais informações sobre a sintaxe SQL do MySQL 8.0, consulte o Manual de referência do MySQL 8.0. Para obter informações sobre as limitações aplicadas ao Aurora MySQL versão 3, consulte Comparar o Aurora MySQL versão 3 e o MySQL 8.0 Community Edition.

Quando você tiver uma conexão com um cluster de bancos de dados Amazon Aurora com compatibilidade do MySQL 5.7, será possível executar os comandos SQL que forem compatíveis com o MySQL versão 5.7. Para obter mais informações sobre a sintaxe de SQL do MySQL 5.7, consulte o Manual de referência do MySQL 5.7. Para obter informações sobre as limitações aplicadas ao Aurora MySQL 5.7, consulte Aurora MySQL versão 2 compatível com o MySQL 5.7.

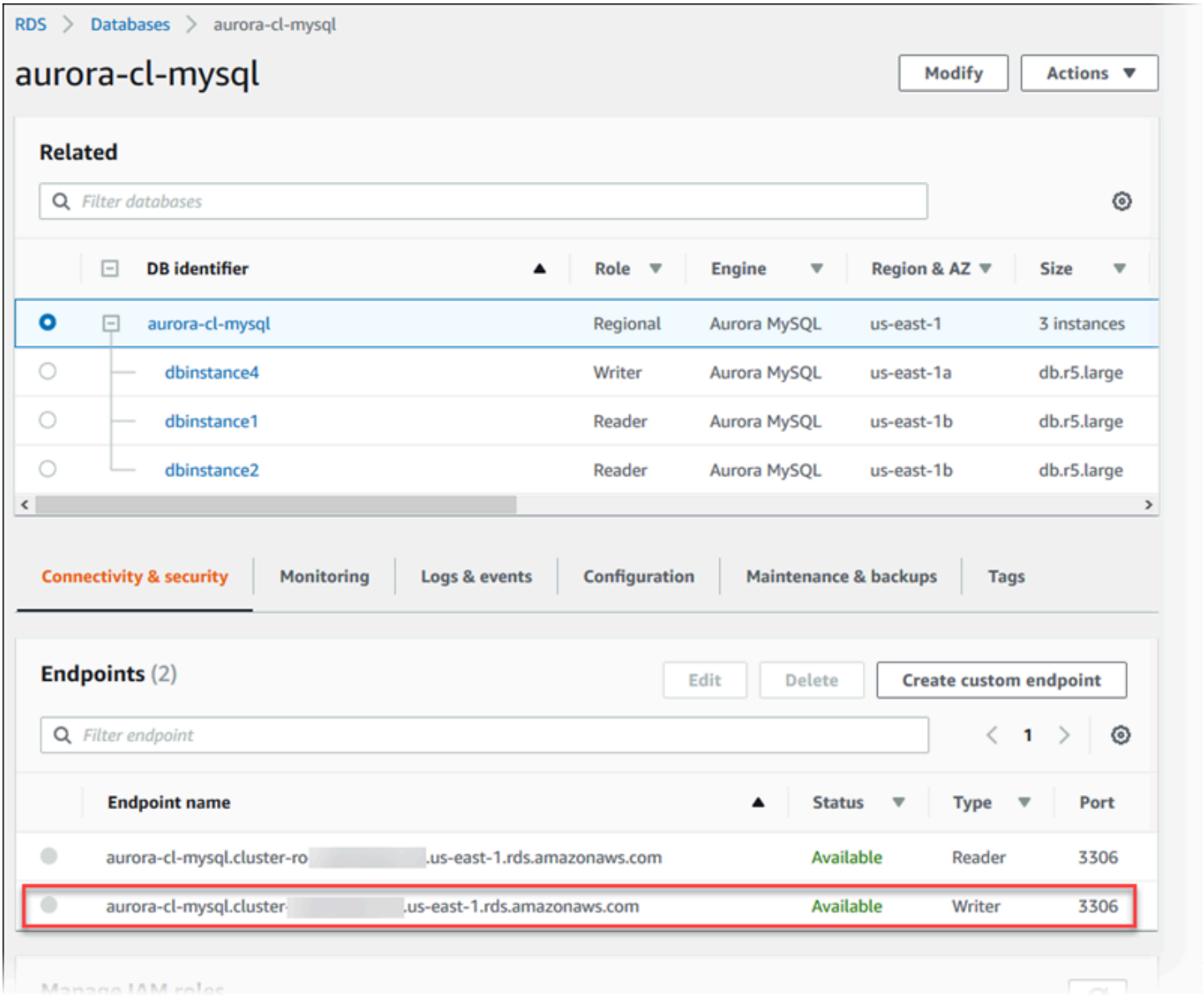
> **Note**
>
> Para obter um guia útil e detalhado sobre como se conectar a um cluster de banco de dados do Amazon Aurora MySQL, você pode consultar o guia Gerenciamento de conexões do Aurora.

Na visualização de detalhes de seu cluster de banco de dados, você pode encontrar o endpoint do cluster, o qual você pode usar na string de conexão do MySQL. O endpoint é composto do nome de domínio e da porta de seu cluster de banco de dados. Por exemplo, se um valor de endpoint for `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, você especificará os seguintes valores em uma string de conexão do MySQL:

- Para o host ou o nome de host, especifique `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Para a porta, especifique `3306` ou o valor da porta que você usou quando criou o cluster de banco de dados

O endpoint de cluster lhe conecta à instância primária do cluster de banco de dados. Você pode realizar as operações de leitura e gravação usando o endpoint do cluster. Seu cluster de banco de dados também pode ter até 15 réplicas do Aurora que oferecem suporte para acesso somente leitura aos dados em seu cluster de banco de dados. A instância primária e cada réplica do Aurora contam com um endpoint exclusivo, independente do endpoint do cluster, e permitem que você se conecte diretamente a uma instância de banco de dados específica no cluster. O endpoint de cluster sempre aponta para a instância primária. Se a instância primária falhar e for substituída, o endpoint do cluster apontará para a nova instância primária.

Para visualizar o endpoint do cluster (endpoint de gravador), selecione Databases (Bancos de dados) no console do Amazon RDS e selecione o nome do cluster de banco de dados para mostrar os detalhes dele.



Tópicos

- Utilitários de conexão do Aurora MySQL
- Conectar-se ao Aurora MySQL com o utilitário MySQL
- Conectar-se ao Aurora MySQL com o driver JDBC da Amazon Web Services (AWS)
- Conectar-se ao Aurora MySQL com o driver Python da Amazon Web Services (AWS)
- Conectar-se ao Aurora MySQL com o driver ODBC da Amazon Web Services (AWS) para MySQL
- Conexão ao Aurora MySQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)
- Conectar-se ao Aurora MySQL usando SSL

Utilitários de conexão do Aurora MySQL

Veja a seguir alguns utilitários de conexão que você pode usar:

- Linha de comando – você pode se conectar a um cluster de bancos de dados Amazon Aurora usando ferramentas, como o utilitário da linha de comando do MySQL. Para obter mais informações sobre como usar o utilitário do MySQL, consulte mysql: o cliente de linha de comando MySQL na documentação do MySQL.
- GUI – você pode usar o utilitário MySQL Workbench para conectar-se usando uma interface de IU. Para obter mais informações, consulte a página Download MySQL Workbench.
- AWS Drivers da:
  - Conectar-se ao Aurora MySQL com o driver JDBC da Amazon Web Services (AWS)
  - Conectar-se ao Aurora MySQL com o driver Python da Amazon Web Services (AWS)
  - Conectar-se ao Aurora MySQL com o driver ODBC da Amazon Web Services (AWS) para MySQL
  - Conexão ao Aurora MySQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)

Conectar-se ao Aurora MySQL com o utilitário MySQL

Use o procedimento a seguir. Nele é pressuposto que você tenha configurado o cluster de banco de dados em uma sub-rede privada na VPC. Você se conecta usando uma instância do Amazon EC2 configurada de acordo com os tutoriais em Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora.

> **Note**
>
> Esse procedimento não exige a instalação do servidor web no tutorial, mas requer a instalação do MariaDB 10.5.

Como conectar-se a um cluster de banco de dados usando o utilitário do MySQL

1. Faça login na instância do EC2 que você está usando para se conectar ao cluster de banco de dados.

   Você deve ver saída semelhante ao seguinte:

```
Last login: Thu Jun 23 13:32:52 2022 from xxx.xxx.xxx.xxx

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-xxx.xxx ~]$
```

2. Digite o comando a seguir em um prompt de comando para se conectar à instância de banco de dados primária de seu cluster de banco de dados.

   Para o parâmetro `-h`, substitua o nome de DNS do endpoint para sua instância primária. Para o parâmetro `-u`, substitua o ID de usuário de uma conta de usuário de banco de dados.

   ```
   mysql -h primary-instance-endpoint.AWS_account.AWS_Region.rds.amazonaws.com -P 3306
    -u database_user -p
   ```

   Por exemplo:

   ```
   mysql -h my-aurora-cluster-instance.c1xy5example.123456789012.eu-
   central-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. Digite a senha do usuário do banco de dados.

   Você deve ver saída semelhante ao seguinte:

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1770
   Server version: 8.0.23 Source distribution

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MySQL [(none)]>
   ```

4. Insira os comandos do SQL.

Conectar-se ao Aurora MySQL com o driver JDBC da Amazon Web Services (AWS)

O driver JDBC da Amazon Web Services (AWS) foi projetado como um wrapper JDBC avançado. Esse wrapper complementa e amplia a funcionalidade de um driver JDBC existente para ajudar as aplicações a aproveitar os recursos de bancos de dados em clusters, como o Aurora MySQL. O driver é compatível de forma intercambiável com os drivers MySQL Connector/J e MariaDB Connector/J da comunidade.

Para instalar o driver JDBC da AWS, anexe o arquivo .jar do driver JDBC da AWS (localizado na aplicação `CLASSPATH`) e mantenha referências ao respectivo driver da comunidade. Atualize o respectivo prefixo do URL de conexão da seguinte forma:

- `jdbc:mysql://` para `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` para `jdbc:aws-wrapper:mariadb://`

Consulte mais informações sobre o driver JDBC da AWS e siga as instruções para usá-lo em Amazon Web Services (AWS) JDBC Driver GitHub repository.

> **Note**
>
> A versão 3.0.3 do utilitário MariaDB Connector/J elimina a compatibilidade com clusters de banco de dados do Aurora, por isso é altamente recomendável mudar para o driver JDBC da AWS.

Conectar-se ao Aurora MySQL com o driver Python da Amazon Web Services (AWS)

O driver Python da Amazon Web Services (AWS) foi projetado como um wrapper Python avançado. Esse wrapper é complementar e amplia a funcionalidade do driver Psycopg de código aberto. O driver Python da AWS é compatível com as versões 3.8 e posterior do Python. É possível instalar o pacote `aws-advanced-python-wrapper` usando o comando `pip`, bem como os pacotes de código aberto `psycopg`.

Para ter mais informações sobre o driver Python da AWS e instruções completas para usá-lo, consulte Amazon Web Services (AWS) Python Driver GitHub repository.

Conectar-se ao Aurora MySQL com o driver ODBC da Amazon Web Services (AWS) para MySQL

O Driver ODBC da AWS para MySQL é um driver de cliente projetado para a alta disponibilidade do Aurora MySQL. O driver pode existir junto com o driver MySQL Connector/ODBC e é compatível com os mesmos fluxos de trabalho.

Para ter mais informações sobre o driver ODBC da AWS para MySQL e instruções completas de uso e instalação, consulte o diretório do GitHub Amazon Web Services (AWS) ODBC Driver for MySQL.

Conexão ao Aurora MySQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)

O wrapper avançado de NodeJS da AWS é complementar e amplia a funcionalidade do driver NodeJS existente. Ele ajuda as aplicações a aproveitarem os recursos de bancos de dados em cluster, como o Aurora MySQL.

Para obter mais informações sobre o Wrapper avançado de NodeJS da AWS e instruções de uso completas, consulte o repositório do GitHub do Wrapper avançado de NodeJS da Amazon Web Services (AWS).

Conectar-se ao Aurora MySQL usando SSL

Você pode usar a criptografia SSL em conexões de uma instância de banco de dados Aurora MySQL. Para mais informações, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

Para conectar-se usando SSL, use o utilitário do MySQL conforme descrito no seguinte procedimento. Se estiver usando uma autenticação de banco de dados do IAM, você deverá usar uma conexão SSL. Para obter mais informações, consulte Autenticação do banco de dados do IAM.

> **Note**
>
> Para conectar-se ao endpoint do cluster usando SSL, o utilitário de conexão do cliente deve oferecer suporte a Subject Alternative Names (SAN). Se o utilitário de conexão de cliente não oferecer suporte a SAN, é possível conectar-se diretamente a instâncias em seu cluster de banco de dados Aurora. Para obter mais informações sobre endpoints do Aurora, consulte Conexões de endpoints do Amazon Aurora.

Para conectar-se a um cluster de banco de dados com SSL usando o utilitário do MySQL

1. Baixe a chave pública do certificado de assinatura do Amazon RDS.

   Para obter informações sobre como baixar certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

2. Digite o seguinte comando em um prompt de comando para se conectar à instância primária de um cluster de banco de dados com SSL usando o utilitário do MySQL. Para o parâmetro `-h`, substitua o nome de DNS do endpoint para sua instância primária. Para o parâmetro `-u`, substitua o ID de usuário de uma conta de usuário de banco de dados. Para o parâmetro `--ssl-ca`, substitua o nome do arquivo de certificado SSL, conforme apropriado. Digite a senha do usuário mestre quando solicitado.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com -u admin_user -p --ssl-ca=[full path]global-bundle.pem --ssl-verify-server-cert
   ```

Você deve ver saída semelhante a.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 8.0.26-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Para obter instruções gerais sobre como criar strings de conexão do RDS for MySQL e localizar a chave pública para as conexões SSL, consulte Conexão a uma instância de banco de dados executando o mecanismo de banco de dados do MySQL.

Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL

Você pode conectar-se a uma instância de banco de dados no cluster de banco de dados Amazon Aurora PostgreSQL usando as mesmas ferramentas que você usa para se conectar a um banco de dados PostgreSQL. Como parte disso, você pode usar a mesma chave pública para conexões Secure Sockets Layer (SSL). Você pode usar as informações de endpoint e da porta da instância

primária ou réplicas do Aurora em seu cluster de banco de dados PostgreSQL do Aurora na string de conexão de qualquer script, utilitário ou aplicativo que se conecta a uma instância de banco de dados PostgreSQL. Na string de conexão, especifique o endereço DNS da instância primária ou do endpoint da réplica do Aurora como o parâmetro de host. Especifique o número da porta do endpoint como o parâmetro da porta.

Quando você tiver uma conexão com uma instância de banco de dados no cluster de bancos de dados Amazon Aurora PostgreSQL, será possível executar qualquer comando SQL compatível com o PostgreSQL.

Na exibição de detalhes do seu cluster de bancos de dados Aurora PostgreSQL, é possível encontrar o nome, o status, o tipo e o número de porta do endpoint do cluster. Utilize esse endpoint e esse número de porta na sua string de conexão do PostgreSQL. Por exemplo, se um valor de endpoint for `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`, você especificará os seguintes valores em uma string de conexão do PostgreSQL:

- Para o host ou o nome de host, especifique `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Para a porta, especifique `5432` ou o valor da porta que você usou quando criou o cluster de banco de dados

O endpoint de cluster lhe conecta à instância primária do cluster de banco de dados. Você pode realizar as operações de leitura e gravação usando o endpoint do cluster. Seu cluster de banco de dados também pode ter até 15 réplicas do Aurora que oferecem suporte para acesso somente leitura aos dados em seu cluster de banco de dados. Cada instância de banco de dados no cluster do Aurora (ou seja, a instância primária e cada réplica do Aurora) conta com um endpoint exclusivo, independente do endpoint do cluster. Esse endpoint exclusivo permite que você conecte a uma instância de banco de dados específica no cluster diretamente. O endpoint de cluster sempre aponta para a instância primária. Se a instância primária falhar e for substituída, o endpoint do cluster apontará para a nova instância primária.

Para visualizar o endpoint do cluster (endpoint de gravador), selecione Databases (Bancos de dados) no console do Amazon RDS e selecione o nome do cluster de banco de dados para mostrar os detalhes dele.
Endpoint do gravador do Aurora PostgreSQL.

Utilitários de conexão do Aurora PostgreSQL

Veja a seguir alguns utilitários de conexão que você pode usar:

- Linha de comando: você pode se conectar a clusters de bancos de dados do Aurora PostgreSQL usando ferramentas, como psql, o terminal interativo do PostgreSQL. Para obter mais informações sobre como usar o terminal interativo do PostgreSQL, consulte psql na documentação do PostgreSQL.
- GUI: você pode usar o utilitário pgAdmin para se conectar a clusters de banco de dados do Aurora PostgreSQL usando uma interface do usuário. Para obter mais informações, consulte a página Download do site pgAdmin.
- AWS Drivers da:
  - Conectar-se ao Aurora PostgreSQL com o driver JDBC da Amazon Web Services (AWS)
  - Conectar-se ao Aurora PostgreSQL com o driver Python da Amazon Web Services (AWS)
  - Conexão ao Aurora PostgreSQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)

Conectar-se ao Aurora PostgreSQL com o driver JDBC da Amazon Web Services (AWS)

O driver JDBC da Amazon Web Services (AWS) foi projetado como um wrapper JDBC avançado. Esse wrapper complementa e amplia a funcionalidade de um driver JDBC existente para ajudar as aplicações a aproveitar os recursos de bancos de dados em clusters, como o Aurora PostgreSQL. O driver é compatível de forma intercambiável com o driver pgJDBC.

Para instalar o driver JDBC da AWS, anexe o arquivo .jar do driver JDBC da AWS (localizado na aplicação `CLASSPATH`) e mantenha referências ao driver pgJDBC da comunidade. Atualize o prefixo do URL de conexão de `jdbc:postgresql://` para `jdbc:aws-wrapper:postgresql://`.

Consulte mais informações sobre o driver JDBC da AWS e siga as instruções para usá-lo em Amazon Web Services (AWS) JDBC Driver GitHub repository.

Conectar-se ao Aurora PostgreSQL com o driver Python da Amazon Web Services (AWS)

O driver Python da Amazon Web Services (AWS) foi projetado como um wrapper Python avançado. Esse wrapper é complementar e amplia a funcionalidade do driver Psycopg de código aberto. O

driver Python da AWS é compatível com as versões 3.8 e posterior do Python. É possível instalar o pacote `aws-advanced-python-wrapper` usando o comando `pip`, bem como os pacotes de código aberto `psycopg`.

Para ter mais informações sobre o driver Python da AWS e instruções completas para usá-lo, consulte Amazon Web Services (AWS) Python Driver GitHub repository.

Conexão ao Aurora PostgreSQL com o Wrapper avançado de NodeJS da Amazon Web Services (AWS)

O wrapper avançado de NodeJS da AWS é complementar e amplia a funcionalidade do driver NodeJS existente. Ele ajuda as aplicações a aproveitarem os recursos de bancos de dados em cluster, como o Aurora PostgreSQL.

Para obter mais informações sobre o Wrapper avançado de NodeJS da AWS e instruções de uso completas, consulte o repositório do GitHub do Wrapper avançado de NodeJS da Amazon Web Services (AWS).

Solução de problemas de falha de conexão do Aurora

As causas comuns de falhas de conexão a um novo cluster de bancos de dados Aurora são as seguintes:

- O grupo de segurança na VPC não permite o acesso: sua VPC precisa permitir conexões de seu dispositivo ou de uma instância do Amazon EC2 por meio da configuração adequada do grupo de segurança na VPC. Para resolver, modifique as regras de entrada do grupo de segurança da VPC para permitir conexões. Para ver um exemplo, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).
- Porta bloqueada por regras de firewall – Verifique o valor da porta configurada para o cluster de bancos de dados Aurora. Se uma regra de firewall bloquear essa porta, você poderá recriar a instância usando uma porta diferente.
- IAMConfiguração incompleta ou incorreta – Se você criou sua instância de bancos de dados Aurora para usar a autenticação baseada em IAM–, verifique se ela está configurada corretamente. Para obter mais informações, consulte Autenticação do banco de dados do IAM.

Para obter mais informações sobre como solucionar problemas Aurora com a conexão, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS.

Grupos de parâmetros para Amazon Aurora

Database parameters (Parâmetros do banco de dados) especifica como o banco de dados é configurado. Por exemplo, os parâmetros do banco de dados podem especificar a quantidade de recursos, como memória, a serem alocados para um banco de dados.

Você gerencia a configuração do banco de dados associando suas instâncias e seus clusters de banco de dados Aurora com grupos de parâmetros. O Aurora define grupos de parâmetros com configurações padrão. Você também pode definir seus próprios grupos de parâmetros com configurações personalizadas.

Tópicos

- Visão geral dos grupos de parâmetros
- Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora
- Grupos de parâmetros de banco de dados para instâncias de banco de dados do Amazon Aurora
- Comparação de grupos de parâmetros de banco de dados
- Especificação de parâmetros de banco de dados

Visão geral dos grupos de parâmetros

Um grupo de parâmetros de cluster de banco de dados atua como um contêiner de valores de configuração de mecanismo que se aplicam a cada instância de banco de dados em um cluster de bancos de dados do Aurora. Por exemplo, o modelo de armazenamento compartilhado do Aurora requer que cada instância de banco de dados em m um cluster do Aurora use a mesma configuração de parâmetros, como `innodb_file_per_table`. Dessa forma, os parâmetros que afetam o layout de armazenamento físico fazem parte do grupo de parâmetros do cluster. O grupo de parâmetros de cluster de banco de dados também inclui valores padrão para todos os parâmetros em nível de instância.

Um grupo de parâmetros de banco de dados atua como um contêiner para valores de configuração de mecanismo que são aplicados a uma ou mais instâncias de bancos de dados. Grupos de parâmetros de banco de dados aplicam-se a instâncias de banco de dados no Amazon RDS e no Aurora. Essas definições de configuração aplicam-se a propriedades que podem variar entre as instâncias de banco de dados em um cluster do Aurora, como os tamanhos dos buffers de memória.

Tópicos

- Grupos de parâmetros padrão e personalizados
- Parâmetros estáticos e dinâmicos de cluster de banco de dados
- Parâmetros estáticos e dinâmicos de instância de banco de dados
- Parâmetros de conjunto de caracteres
- Parâmetros e valores de parâmetros compatíveis

Grupos de parâmetros padrão e personalizados

Se você criar uma instância de banco de dados sem especificar um grupo de parâmetros de banco de dados, essa instância usará o grupo de parâmetros de banco de dados padrão. Da mesma forma, se você criar um cluster de banco de dados Aurora sem especificar um grupo de parâmetros de cluster de banco de dados, esse cluster usará um grupo de parâmetros de cluster de banco de dados padrão. Cada grupo de parâmetros de banco de dados padrão contém padrões de mecanismo de banco de dados e do sistema Amazon RDS com base no mecanismo, na classe de computação e no armazenamento alocado da instância.

Não é possível modificar as configurações de parâmetros de um grupo de parâmetros padrão. Em vez disso, você pode fazer o seguinte:

1. Crie um novo grupo de parâmetros.
2. Altere as configurações dos parâmetros desejados. Nem todos os parâmetros de mecanismo de banco de dados em um grupo de parâmetros podem ser modificados.
3. Modifique a instância de banco de dados ou o cluster de banco de dados para associar o novo grupo de parâmetros.

   Para obter informações sobre como modificar um cluster de banco de dados ou uma instância de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

> **Note**
>
> Se você modificou sua instância de banco de dados para usar um grupo de parâmetros personalizado, e iniciar a instância de banco de dados, o RDS reinicializará automaticamente a instância de banco de dados como parte do processo de inicialização. Para instâncias multi-AZ do RDS para SQL Server com a opção “Sempre ativa” ou

> “Espelhamento” habilitada, um failover é esperado quando a instância é reiniciada após o processo de inicialização.

O RDS aplica os parâmetros estáticos e dinâmicos modificados a um grupo de parâmetros recém-associado somente depois que a instância de banco de dados é reinicializada. No entanto, se você modificar parâmetros dinâmicos no grupo de parâmetros de banco de dados depois de associá-lo à instância de banco de dados, essas alterações serão aplicadas imediatamente sem uma reinicialização. Para obter mais informações sobre como alterar o grupo de parâmetros de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Se você atualizar parâmetros dentro de um grupo de parâmetros de banco de dados, as alterações se aplicarão a todas as instâncias de banco de dados associadas a esse grupo de parâmetros. Da mesma forma, se você atualizar os parâmetros dentro de um grupo de parâmetros de cluster de banco de dados do Aurora, as alterações se aplicarão a todos os clusters do Aurora que estiverem associados a esse grupo.

Se você não quiser criar um grupo de parâmetros do zero, poderá copiar um grupo de parâmetros existente com o comando copy-db-parameter-group ou o comando copy-db-cluster-parameter-group da AWS CLI. Você pode perceber que copiar um grupo de parâmetros é útil em alguns casos. Por exemplo, talvez você queira incluir a maioria dos valores e parâmetros personalizados de um grupo de parâmetros existente em um novo grupo de parâmetros.

Parâmetros estáticos e dinâmicos de cluster de banco de dados

Os parâmetros de cluster de banco de dados são estáticos ou dinâmicos. As diferenças são as seguintes:

- Quando você altera um parâmetro estático e salva o grupo de parâmetros de cluster de banco de dados, a alteração entra em vigor depois que você reinicializa manualmente as instâncias de banco de dados em cada cluster de banco de dados associado. Quando você usa o Console de gerenciamento da AWS para alterar os valores dos parâmetros estáticos do cluster de banco de dados, ele sempre usa `pending-reboot` para `ApplyMethod`.
- Quando você altera um parâmetro dinâmico, por padrão, a alteração de parâmetro entra em vigor imediatamente, sem a necessidade de reinicialização. Quando você usa o console, ele sempre usa `immediate` para `ApplyMethod`. Para adiar a alteração do parâmetro até a finalização da reinicialização das instâncias de banco de dados em um cluster de banco de dados associado, use

a AWS CLI ou a API do RDS. Defina o `ApplyMethod` como `pending-reboot` para a alteração do parâmetro.

Para obter mais informações sobre como usar a AWS CLI para alterar um valor de parâmetro, consulte modify-db-cluster-parameter-group. Para obter mais informações sobre como usar a API do RDS para alterar um valor de parâmetro, consulte ModifyDBClusterParameterGroup.

Se você alterar o grupo de parâmetros de cluster de banco de dados associado a um cluster de banco de dados, reinicialize as instâncias de banco de dados no cluster. A reinicialização aplica as alterações a todas as instâncias de banco de dados no cluster de banco de dados. Para determinar se as instâncias de banco de dados de um cluster de banco de dados deve ser reinicializada para as alterações serem aplicadas, execute o comando da AWS CLI a seguir.

```
aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier
```

Verifique o valor `DBClusterParameterGroupStatus` da instância do banco de dados primário na saída. Se o valor for `pending-reboot`, reinicialize as instâncias de banco de dados do cluster de banco de dados.

Parâmetros estáticos e dinâmicos de instância de banco de dados

Os parâmetros de instância de banco de dados são estáticos ou dinâmicos. As diferenças são as seguintes:

- Quando você altera um parâmetro estático e salva o grupo de parâmetros de banco de dados, a alteração entra em vigor depois que você reinicializa as instâncias de banco de dados manualmente. Para parâmetros estáticos, o console sempre usa `pending-reboot` para o `ApplyMethod`.
- Quando você altera um parâmetro dinâmico, por padrão, a alteração de parâmetro entra em vigor imediatamente, sem a necessidade de reinicialização. Quando você usa o Console de gerenciamento da AWS para alterar os valores dos parâmetros da instância de banco de dados, ele sempre usa `immediate` para `ApplyMethod` para parâmetros dinâmicos. Para adiar a alteração do parâmetro até a finalização da reinicialização de uma instância de banco de dados associada, use a AWS CLI ou a API do RDS. Defina o `ApplyMethod` como `pending-reboot` para a alteração do parâmetro.

Para obter mais informações sobre como usar a AWS CLI para alterar um valor de parâmetro, consulte modify-db-parameter-group. Para obter mais informações sobre como usar a API do RDS para alterar um valor de parâmetro, consulte ModifyDBParameterGroup.

Se uma instância de banco de dados não estiver usando as últimas alterações de seu grupo de parâmetros de banco de dados associado, o console mostrará o status pending-reboot para o grupo de parâmetros de banco de dados. Esse status não ocasionará uma reinicialização automática durante a próxima janela de manutenção. Para aplicar as alterações de parâmetro mais recentes a essa instância de banco de dados, reinicialize-a manualmente.

Parâmetros de conjunto de caracteres

Antes de criar um cluster de banco de dados, defina todos os parâmetros relacionados ao conjunto de caracteres ou ao agrupamento do seu banco de dados no grupo de parâmetros. Também faça isso antes de criar um banco de dados nele. Dessa forma, você garante que o banco de dados padrão e os novos bancos de dados usem o conjunto de caracteres e os valores de agrupamento que você especificar. Se você alterar o conjunto de caracteres ou os parâmetros de agrupamento, as alterações de parâmetros não serão aplicadas aos bancos de dados existentes.

Em mecanismos de bancos de dados, você pode alterar o conjunto de caracteres ou os valores de agrupamento para um banco de dados existente usando o comando `ALTER DATABASE`. Por exemplo:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

Para obter mais informações sobre como alterar o conjunto de caracteres ou valores de agrupamento de um banco de dados, consulte a documentação referente ao seu mecanismo de banco de dados.

Parâmetros e valores de parâmetros compatíveis

Para determinar os parâmetros compatíveis com seu mecanismo de banco de dados, visualize os parâmetros no grupo de parâmetros de banco de dados e no grupo de parâmetros do cluster de banco de dados utilizados pela instância ou pelo cluster de banco de dados. Para obter mais informações, consulte Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados no Amazon Aurora e Visualizar valores de parâmetros de um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Em muitos casos, é possível especificar valores de parâmetros inteiros e boolianos usando expressões, fórmulas e funções. As funções podem incluir uma expressão matemática de log.

No entanto, nem todos os parâmetros são compatíveis com expressões, fórmulas e funções para valores de parâmetros. Para obter mais informações, consulte Especificação de parâmetros de banco de dados.

Para um banco de dados global Aurora, especifique definições de configuração diferentes para os clusters do Aurora individuais. Verifique se as configurações são suficientemente semelhantes para produzir um comportamento consistente caso você promova um cluster secundário para ser o cluster primário. Por exemplo, use as mesmas configurações para os fusos horários e os conjuntos de caracteres em todos os clusters de um banco de dados global Aurora.

Definir incorretamente os parâmetros em um grupo de parâmetros pode causar efeitos adversos não intencionais, inclusive diminuição da performance e instabilidade no sistema. Sempre tenha cuidado ao modificar parâmetros de bancos de dados e faça backup dos dados antes de modificar um grupo de parâmetros. Faça testes com alterações de configuração de grupos de parâmetros em uma instância ou um cluster de banco de dados de teste antes de aplicar essas alterações de grupos de parâmetros a uma instância ou um cluster de banco de dados de produção.

Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora

Clusters de banco de dados Amazon Aurora usam grupos de parâmetros de cluster de banco de dados As seções a seguir descrevem a configuração e o gerenciamento de grupos de parâmetros de cluster de banco de dados.

Tópicos

- Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora
- Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora
- Associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados no Amazon Aurora
- Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora
- Redefinir parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora
- Copiar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora
- Listar grupos de parâmetros de cluster de banco de dadosno Amazon Aurora
- Visualizar valores de parâmetros de um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

- Excluir um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora

O Aurora usa um sistema de configurações de dois níveis:

- Os parâmetros em um grupo de parâmetros de cluster de banco de dados aplicam-se a cada instância de banco de dados em um cluster de banco de dados. Seus dados são armazenados no subsistema de armazenamento compartilhado do Aurora. Por causa disso, todos os parâmetros relacionados ao layout físico de dados de tabelas devem ser os mesmos para todas as instâncias de banco de dados em um cluster do Aurora. Da mesma forma, como as instâncias de bancos de dados do Aurora são conectadas por replicação, todos os parâmetros para configurações de replicação devem ser idênticos por todo um cluster do Aurora.
- Os parâmetros em um grupo de parâmetros de banco de dados aplicam-se a uma única instância de banco de dados em um cluster de bancos de dados Aurora. Esses parâmetros estão relacionados a aspectos como o uso da memória, que você pode variar entre instâncias de bancos de dados no mesmo cluster do Aurora. Por exemplo, um cluster geralmente contém instâncias de banco de dados com diferentes classes de instância da AWS.

Cada cluster do Aurora é associado a um grupo de parâmetros de cluster de banco de dados. Esse grupo de parâmetros atribui valores padrão para cada valor de configuração para o mecanismo de banco de dados correspondente. O grupo de parâmetros de cluster também inclui valores padrão para os parâmetros em nível de instância e de cluster. Cada instância de banco de dados em um cluster provisionado ou do Aurora Serverless v2 herda as configurações desse grupo de parâmetros de cluster de banco de dados.

Cada instância de banco de dados também é associada a um grupo de parâmetros de banco de dados. Os valores no grupo de parâmetros de banco de dados podem substituir os valores padrão do grupo de parâmetros do cluster. Por exemplo, se uma instância em um cluster tiver problemas, você poderá atribuir um grupo de parâmetros de banco de dados personalizado a essa instância. O grupo de parâmetros personalizado pode ter configurações específicas para parâmetros relacionados à depuração ou ao ajuste de performance.

O Aurora atribui grupos de parâmetros padrão quando você cria um cluster ou uma nova instância de banco de dados, com base no mecanismo de banco de dados e na versão especificados. Em vez disso, você pode especificar grupos de parâmetros personalizados. Você mesmo cria esses

grupos de parâmetros e pode editar os valores dos parâmetros. Você pode especificar esses grupos de parâmetros personalizados no momento da criação. Também é possível modificar um cluster de banco de dados ou instância posteriormente para usar um grupo de parâmetros personalizado.

Para instâncias do Aurora Serverless v2 e provisionadas, quaisquer valores de configuração que você modificar no grupo de parâmetros de cluster de banco de dados substituirão os valores padrão no grupo de parâmetros de banco de dados. Se você editar os valores correspondentes no grupo de parâmetros de banco de dados, eles substituirão as configurações do grupo de parâmetros de cluster de banco de dados.

Qualquer configuração de parâmetro de banco de dados que você modificar terá precedência sobre os valores do grupo de parâmetros de cluster de banco de dados, mesmo que você altere os parâmetros de configuração de volta para seus valores padrão. É possível ver quais parâmetros são substituídos usando o comando describe-db-parameters AWS CLI ou a operação DescribeDBParameters da API do RDS. O campo `Source` conterá o valor `user` se você tiver modificado esse parâmetro. Para redefinir um ou mais parâmetros de forma que o valor do grupo de parâmetros de cluster de banco de dados tenha precedência, use o comando reset-db-parameter-group AWS CLI ou a operação ResetDBParameterGroup da API do RDS.

Os parâmetros do cluster e da instância de banco de dados disponíveis no Aurora variam de acordo com a compatibilidade do mecanismo de banco de dados.

Mecanismo do banco de dados	Parâmetros
Aurora MySQL	Consulte Parâmetros de configuração do Aurora MySQL. Para clusters do Aurora Serverless, consulte detalhes adicionais em Trabalhar com grupos de parâmetros para o Aurora Serverless v2 e Grupos de parâmetros para Aurora Serverless v1.
Aurora PostgreSQL	Consulte Amazon Aurora PostgreSQL parameters. Para clusters do Aurora Serverless, consulte detalhes adicionais em Trabalhar com grupos de parâmetros para o Aurora Serverless v2 e Grupos de parâmetros para Aurora Serverless v1.

> **Note**
>
> Aurora Serverless v1Os clusters do têm apenas grupos de parâmetros de cluster de banco de dados associados, não grupos de parâmetros de banco de dados. Para clusters do Aurora Serverless v2, você faz todas as alterações nos parâmetros personalizados no grupo de parâmetros de cluster de banco de dados.
> Aurora Serverless v2O usa grupos de parâmetros de cluster de banco de dados e grupos de parâmetros de banco de dados Com o Aurora Serverless v2, você pode modificar quase todos os parâmetros de configuração. O Aurora Serverless v2 substitui as configurações de alguns parâmetros de configuração relacionados à capacidade para que sua workload não seja interrompida quando houver redução de escala vertical das instâncias do Aurora Serverless v2.
> Para saber mais sobre as definições de configuração do Aurora Serverless e quais configurações você pode modificar, consulte Trabalhar com grupos de parâmetros para o Aurora Serverless v2 e Grupos de parâmetros para Aurora Serverless v1.

Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Você pode criar um novo grupo de parâmetros de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Depois de criar um grupo de parâmetros de cluster de banco de dados, você deve esperar pelo menos 5 minutos antes de criar seu primeiro cluster que usa esse grupo de parâmetros de cluster de banco de dados. Isso permite que o Amazon RDS conclua totalmente a criação do grupo de parâmetros antes que seja usado pelo novo cluster de banco de dados. É possível usar a página Parameter Groups (Grupos de parâmetros) do console do Amazon RDS ou o comando describe-db-cluster-parameters para verificar se o seu grupo de parâmetros de cluster de banco de dados foi criado.

As seguintes limitações se aplicam ao nome do grupo de parâmetros do cluster de banco de dados:

- O nome deve ter de 1 a 255 letras, números ou hifens.

  Os nomes de grupos de parâmetros padrão podem incluir um ponto, como `default.aurora-mysql5.7`. No entanto, nomes de grupos de parâmetros personalizados não podem incluir um ponto.
- O primeiro caractere deve ser uma letra.

- O nome não pode terminar com hífen nem conter dois hífens consecutivos.

Console

Para criar um grupo de parâmetros de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Parameter groups (Grupos de parâmetros).
3. Escolha Criar grupo de parâmetros.
4. Em Nome do grupo de parâmetros, insira o nome do novo grupo de parâmetros do cluster de banco de dados.
5. Em Descrição, insira uma descrição para o novo grupo de parâmetros do cluster de banco de dados.
6. Em Tipo de mecanismo, escolha o mecanismo de banco de dados.
7. Em Família de grupos de parâmetros, selecione uma família de grupos de parâmetros de banco de dados.
8. Escolha Criar.

AWS CLI

Para criar um grupo de parâmetros de cluster de banco de dados, use o comando da AWS CLI `create-db-cluster-parameter-group`.

O exemplo a seguir cria um grupo de parâmetros de cluster de banco de dados chamado mydbclusterparametergroup para o Aurora MySQL versão 5.7 com a descrição "My new parameter group" (Meu novo grupo de parâmetros).

Inclua os seguintes parâmetros necessários:

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Para listar todas as famílias de grupos de parâmetros disponíveis, use o comando a seguir:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> A saída contém duplicatas.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new cluster parameter group"
```

Para Windows:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new cluster parameter group"
```

O comando gerará uma saída semelhante à seguinte:

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
    }
}
```

API do RDS

Para criar um grupo de parâmetros de cluster de banco de dados, use a ação da API do RDS `CreateDBClusterParameterGroup`.

Inclua os seguintes parâmetros necessários:

- `DBClusterParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados no Amazon Aurora

Você pode criar seus próprios grupos de parâmetros de cluster de banco de dados com configurações personalizadas. Você pode associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados usando o Console de gerenciamento da AWS ou a API RDSAWS CLI. Você pode fazer isso ao criar ou modificar um cluster de banco de dados.

Para obter informações sobre como criar um grupo de parâmetros de cluster de banco de dados, consulte Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora. Para obter informações sobre como criar um cluster de banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora. Para obter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

> **Note**
>
> Para o Aurora PostgreSQL 15.2, 14.7, 13.10, 12.14 e todas as versões 11, ao alterar o grupo de parâmetros de cluster de banco de dados associado a um cluster de banco de dados, reinicialize cada instância de réplica para aplicar as alterações.
> Para determinar se a instância de banco de dados principal de um cluster de banco de dados deve ser reinicializada para aplicar alterações, execute o seguinte comando AWS CLI: `aws rds describe-db-clusters --db-cluster-identifier` *`db_cluster_identifier`*
> Verifique o valor `DBClusterParameterGroupStatus` da instância do banco de dados primário na saída. Se o valor for `pending-reboot`, reinicie a instância de banco de dados principal do cluster de banco de dados.

Console

Para associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha o cluster de banco de dados que você deseja modificar.
3. Escolha Modify (Modificar). A página Modify DB cluster (Modificar cluster de banco de dados) é exibida.
4. Altere a configuração do grupo de parâmetros do cluster de banco de dados.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.

   A alteração é aplicada imediatamente, independentemente da configuração Agendamento de modificações.
6. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify cluster (Modificar cluster) para salvar as alterações.

   Como alternativa, escolha Back (Voltar) para editar suas alterações ou escolha Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados, use o comando da AWS CLI `modify-db-cluster` com as seguintes opções:

- `--db-cluster-name`
- `--db-cluster-parameter-group-name`

O exemplo a seguir associa o `mydbclpg` grupo de parâmetros de banco de dados ao cluster de banco de dados`mydbcluster`.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --db-cluster-parameter-group-name mydbclpg
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-parameter-group-name mydbclpg
```

API do RDS

Para associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados, use a operação da API do RDS `ModifyDBCluster` com os seguintes parâmetros:

- `DBClusterIdentifier`
- `DBClusterParameterGroupName`

Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

É possível modificar valores de parâmetros em um grupo de parâmetros do cluster de banco de dados criado pelo cliente. Não é possível alterar os valores de parâmetros em um grupo de parâmetros do cluster de banco de dados padrão. As alterações dos parâmetros em um grupo de parâmetros de cluster de banco de dados criado pelo cliente são aplicadas a todos os clusters de bancos de dados que estão associados a esse grupo de parâmetros de cluster de banco de dados.

Console

Para modificar um grupo de parâmetros de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros que você deseja modificar.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Edit (Editar).
5. Altere os valores dos parâmetros que você deseja modificar. Você pode percorrer os parâmetros usando as teclas de seta no canto superior direito da caixa de diálogo.

Não altere valores em um grupo de parâmetros padrão.

6. Selecione Save changes.
7. Reinicialize a instância de banco de dados primária (de gravador) no cluster para aplicar as alterações a ela.
8. Depois, reinicialize as instâncias de banco de dados de leitor para aplicar as alterações a elas.

   Se você não reiniciar as instâncias de banco de dados, uma operação de failover poderá levar mais tempo do que o normal.

AWS CLI

Para modificar um grupo de parâmetros de cluster de banco de dados, use o comando da AWS CLI `modify-db-cluster-parameter-group` com os seguintes parâmetros obrigatórios:

- `--db-cluster-parameter-group-name`
- `--parameters`

O exemplo a seguir modifica os valores `server_audit_logging` e `server_audit_logs_upload` no grupo de parâmetros de cluster de banco de dados chamado mydbclusterparametergroup.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^
```

```
"ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

O comando produz uma saída como a seguinte:

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

API do RDS

Para modificar um grupo de parâmetros de cluster de banco de dados, use o comando da API do RDS `ModifyDBClusterParameterGroup` com os seguintes parâmetros obrigatórios:

- `DBClusterParameterGroupName`
- `Parameters`

Redefinir parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Você pode redefinir parâmetros para seus valores padrão em um grupo de parâmetro de cluster de banco de dados criado pelo cliente. As alterações dos parâmetros em um grupo de parâmetros de cluster de banco de dados criado pelo cliente são aplicadas a todos os clusters de bancos de dados que estão associados a esse grupo de parâmetros de cluster de banco de dados.

> **Note**
>
> Em um grupo de parâmetro de cluster de banco de dados padrão, os parâmetros são sempre definidos como seus valores padrão.

Console

Para redefinir parâmetros em um grupo de parâmetros de cluster de banco de dados para seus valores padrão

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

3. Na lista, escolha o grupo de parâmetros.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Edit (Editar).
5. Escolha os parâmetros que você deseja redefinir para os valores padrão. Você pode percorrer os parâmetros usando as teclas de seta no canto superior direito da caixa de diálogo.

   Não redefina os valores em um grupo de parâmetros padrão.
6. Escolha Reset (Redefinir) e, em seguida, confirme escolhendo Reset parameters (Redefinir parâmetros).
7. Reinicialize a instância de banco de dados primária no cluster de banco de dados para aplicar as alterações a todas as instâncias de banco de dados no cluster do banco de dados.

AWS CLI

Para redefinir parâmetros em um grupo de parâmetros de cluster de banco de dados para seus valores padrão, use o comando da AWS CLI `reset-db-cluster-parameter-group` com a seguinte opção obrigatória: `--db-cluster-parameter-group-name`.

Para redefinir todos os parâmetros no grupo de parâmetro do cluster de banco de dados, especifique a opção `--reset-all-parameters`. Para redefinir parâmetros específicos, especifique a opção `--parameters`.

O exemplo a seguir redefine todos os parâmetros no grupo de parâmetros de banco de dados chamado mydbparametergroup para seus valores padrão.

Example

Para Linux, macOS ou Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

O exemplo a seguir redefine os valores padrão `server_audit_logging` e `server_audit_logs_upload` no grupo de parâmetro de cluster de banco de dados chamado mydbclusterparametergroup.

Example

Para Linux, macOS ou Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

O comando produz uma saída como a seguinte:

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

API do RDS

Para redefinir parâmetros em um grupo de parâmetros de cluster de banco de dados para seus valores padrão, use o comando da API do RDS de `ResetDBClusterParameterGroup` com o seguinte parâmetro necessário: `DBClusterParameterGroupName`.

Para redefinir todos os parâmetros no grupo de parâmetros do cluster de banco de dados, defina o parâmetro `ResetAllParameters` para `true`. Para redefinir parâmetros específicos, especifique o parâmetro `Parameters`.

Copiar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Você pode copiar os grupos de parâmetros de cluster de banco de dados personalizados que criou. Copiar um grupo de parâmetros é uma solução conveniente quando você já criou um grupo

de parâmetros de cluster de banco de dados e deseja incluir a maioria dos parâmetros e valores personalizados desse grupo em um novo grupo de parâmetros de cluster de banco de dados. É possível copiar um grupo de parâmetros de cluster de banco de dados usando o comando AWS CLI copy-db-cluster-parameter-group ou a operação da API do RDS CopyDBClusterParameterGroup.

Depois de copiar um grupo de parâmetros de cluster de banco de dados, você deve esperar pelo menos 5 minutos antes de criar seu primeiro cluster que usa esse grupo de parâmetros de cluster de banco de dados. Isso permite que o Amazon RDS conclua totalmente a cópia do grupo de parâmetros antes que seja usado pelo novo cluster de banco de dados. É possível usar a página Parameter Groups (Grupos de parâmetros) do console do Amazon RDS ou o comando describe-db-cluster-parameters para verificar se o seu grupo de parâmetros de cluster de banco de dados foi criado.

> **Note**
>
> Não é possível copiar um grupo de parâmetros padrão. No entanto, é possível criar um grupo de parâmetros com base em um grupo de parâmetros padrão.
> Não é possível copiar um grupo de parâmetros de cluster de banco de dados para uma Conta da AWS ou Região da AWS diferente.

Console

Para copiar um grupo de parâmetros de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros personalizado que você deseja copiar.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Copy (Copiar).
5. Em New DB parameter group identifier (Novo identificador do grupo de parâmetros do banco de dados), digite um nome para o novo grupo de parâmetros.
6. Em Description (Descrição), digite uma descrição para o novo grupo de parâmetros.
7. Escolha Copiar.

AWS CLI

Para copiar um grupo de parâmetros de cluster de banco de dados, use o comando da AWS CLI `copy-db-cluster-parameter-group` com os seguintes parâmetros obrigatórios:

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

O exemplo a seguir cria um novo grupo de parâmetros de cluster de banco de dados chamado `mygroup2` que é uma cópia do grupo de parâmetros de cluster de banco de dados `mygroup1`.

Example

Para Linux, macOS ou Unix:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Para Windows:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

API do RDS

Para copiar um grupo de parâmetros de cluster de banco de dados, use a operação da API do RDS `CopyDBClusterParameterGroup` com os seguintes parâmetros obrigatórios:

- `SourceDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

Listar grupos de parâmetros de cluster de banco de dadosno Amazon Aurora

Você pode listar os grupos de parâmetros do cluster de banco de dados que criou para sua conta da AWS.

> **Note**
>
> Grupos de parâmetros padrão são criados automaticamente a partir de um modelo de parâmetro padrão quando você cria um cluster de banco de dados para um mecanismo e uma versão de banco de dados específicos. Esses grupos de parâmetros padrão contêm configurações de parâmetros preferenciais e não podem ser modificados. Quando você cria um grupo de parâmetros personalizado, pode modificar as configurações desses parâmetros.

Console

Para listar todos os grupos de parâmetros do cluster de banco de dados para uma conta da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

   Os parameter groups de cluster de banco de dados são exibidos na lista com DB cluster parameter group (Parameter group de cluster de banco de dados) para o Type (Tipo).

AWS CLI

Para listar todos os grupos de parâmetros de cluster de banco de dados de uma conta da AWS, use o comando da AWS CLI `describe-db-cluster-parameter-groups`.

Example

O exemplo a seguir lista todos os grupos de parâmetros de cluster de banco de dados de uma conta da AWS.

```
aws rds describe-db-cluster-parameter-groups
```

O exemplo a seguir descreve o grupo de parâmetros mydbclusterparametergroup.

Para Linux, macOS ou Unix:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Para Windows:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

O comando retorna uma resposta como a seguinte:

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

API do RDS

Para listar todos os grupos de parâmetros de cluster de banco de dados de uma conta da AWS, use a ação da API do RDS `DescribeDBClusterParameterGroups`.

Visualizar valores de parâmetros de um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Você pode obter uma lista de todos os parâmetros em um grupo de parâmetros de cluster de banco de dados e seus valores.

Console

Para visualizar os valores de parâmetros para um grupo de parâmetros de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

Os grupos de parâmetros de cluster de banco de dados são exibidos na lista com DB cluster parameter group (Grupo de parâmetros de cluster de banco de dados) para o Type (Tipo).

3. Escolha o nome do grupo de parâmetros de cluster de banco de dados para ver sua lista de parâmetros.

AWS CLI

Para visualizar os valores de parâmetros para um grupo de parâmetros de cluster de banco de dados, use o comando da AWS CLI `describe-db-cluster-parameters` com o parâmetro obrigatório a seguir.

- `--db-cluster-parameter-group-name`

Example

O exemplo a seguir lista os parâmetros e os valores de parâmetros para um grupo de parâmetros de cluster de banco de dados chamado mydbclusterparametergroup, no formato JSON.

O comando retorna uma resposta como a seguinte:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
```

```
        {
            "ParameterName": "aurora_binlog_read_buffer_size",
            "ParameterValue": "5242880",
            "Description": "Read buffer size used by master dump thread when the switch
 aurora_binlog_use_large_read_buffer is ON.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "8192-536870912",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },

...
```

API do RDS

Para visualizar os valores de parâmetros para um grupo de parâmetros de cluster de banco de dados, use o comando `DescribeDBClusterParameters` da API do RDS com o seguinte parâmetro necessário.

- `DBClusterParameterGroupName`

Em alguns casos, os valores permitidos para um parâmetro não são mostrados. São sempre parâmetros em que a origem é o padrão do mecanismo de banco de dados.

Para visualizar os valores desses parâmetros, você pode executar as seguintes instruções SQL:

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
```

```
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

Excluir um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

É possível excluir um grupo de parâmetros de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Um grupo de parâmetros do cluster de banco de dados será elegível para exclusão somente se não estiver associado a um cluster de banco de dados.

Console

Para excluir um grupo de parâmetros

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Parameter groups (Grupos de parâmetros).

   Os grupos de parâmetros aparecem em uma lista.
3. Escolha o nome dos grupos de parâmetros do cluster de banco de dados a serem excluídos.
4. Escolha Ações e então Excluir.
5. Revise os nomes dos grupos de parâmetros e escolha Excluir.

AWS CLI

Para excluir um grupo de parâmetros de cluster de banco de dados, use o comando `delete-db-cluster-parameter-group` da AWS CLI com os parâmetros obrigatórios a seguir.

- `--db-parameter-group-name`

Example

O exemplo a seguir exclui um grupo de parâmetros do cluster de banco de dados chamado mydbparametergroup.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

API do RDS

Para excluir um grupo de parâmetros de cluster de banco de dados, use o comando `DeleteDBClusterParameterGroup` da API do RDS com os parâmetros obrigatórios a seguir.

- `DBParameterGroupName`

Grupos de parâmetros de banco de dados para instâncias de banco de dados do Amazon Aurora

As instâncias de banco de dados usam grupos de parâmetros de banco de dados. As seções a seguir descrevem a configuração e o gerenciamento de grupos de parâmetros de instância de banco de dados.

Tópicos

- Criar um grupo de parâmetros de banco de dados no Amazon Aurora
- Associar um grupo de parâmetros de banco de dados a uma instância de banco de dados no Amazon Aurora
- Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora
- Redefinir parâmetros em um grupo de parâmetros de banco de dados como seus valores padrão no Amazon Aurora
- Copiar um grupo de parâmetros de banco de dados no Amazon Aurora
- Listar grupos de parâmetros de banco de dados no Amazon Aurora
- Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados no Amazon Aurora
- Excluir um grupo de parâmetros de banco de dados no Amazon Aurora

Criar um grupo de parâmetros de banco de dados no Amazon Aurora

Você pode criar um novo grupo de parâmetros de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

As seguintes limitações se aplicam ao nome do grupo de parâmetros de banco de dados:

- O nome deve ter de 1 a 255 letras, números ou hifens.

Os nomes de grupos de parâmetros padrão podem incluir um ponto, como `default.mysql8.0`. No entanto, nomes de grupos de parâmetros personalizados não podem incluir um ponto.

- O primeiro caractere deve ser uma letra.
- O nome não pode terminar com hífen nem conter dois hífens consecutivos.

Console

Para criar um grupo de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Parameter groups (Grupos de parâmetros).
3. Escolha Create parameter group (Criar grupo de parâmetros).
4. Em Nome do grupo de parâmetros, digite o nome do novo grupo de parâmetros de banco de dados.
5. Em Descrição, digite uma descrição para o novo grupo de parâmetros de banco de dados.
6. Em Tipo de mecanismo, escolha o mecanismo de banco de dados.
7. Em Família de grupos de parâmetros, selecione uma família de grupos de parâmetros de banco de dados.
8. Em Tipo, se aplicável, selecione Grupo de parâmetros de banco de dados.
9. Escolha Criar.

AWS CLI

Para criar um grupo de parâmetros de banco de dados, use o comando AWS CLI `create-db-parameter-group`da . O exemplo a seguir cria um grupo de parâmetros de banco de dados chamado mydbparametergroup para o MySQL versão 8.0 com a descrição "My new parameter group (Meu novo grupo de parâmetros)."

Inclua os seguintes parâmetros necessários:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Para listar todas as famílias de grupos de parâmetros disponíveis, use o comando a seguir:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> A saída contém duplicatas.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new parameter group"
```

Para Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new parameter group"
```

O comando gerará uma saída semelhante à seguinte:

```
DBPARAMETERGROUP  mydbparametergroup  aurora-mysql5.7  My new parameter group
```

API do RDS

Para criar um grupo de parâmetros de banco de dados, use a operação da API `CreateDBParameterGroup` do RDS.

Inclua os seguintes parâmetros necessários:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Associar um grupo de parâmetros de banco de dados a uma instância de banco de dados no Amazon Aurora

Você pode criar seus próprios grupos de parâmetros de banco de dados com configurações personalizadas. Você pode associar um grupo de parâmetros de banco de dados a uma instância de banco de dados usando a Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Você pode fazer isso ao criar ou modificar uma instância de banco de dados.

Para obter mais informações sobre como criar um grupo de parâmetros de banco de dados, consulte Criar um grupo de parâmetros de banco de dados no Amazon Aurora. Para ter mais informações sobre como modificar uma instância de banco de dados, consulte Modificar uma instância de banco de dados em um cluster de banco de dados.

> **Note**
>
> Ao associar um novo grupo de parâmetros de banco de dados a uma instância de banco de dados, os parâmetros estáticos e dinâmicos modificados serão aplicados somente após a reinicialização da instância de banco de dados. No entanto, se você modificar parâmetros dinâmicos no grupo de parâmetros de banco de dados depois de associá-lo à instância de banco de dados, essas alterações serão aplicadas imediatamente sem uma reinicialização.

Console

Para associar um grupo de parâmetros de banco de dados a uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e a instância de banco de dados que você deseja modificar.
3. Selecione Modify. A página Modify DB instance (Modificar instância de banco de dados) será exibida.
4. Altere a configuração do grupo de parâmetros de banco de dados.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. (Opcional) Escolha Apply immediately (Aplicar imediatamente) para aplicar as alterações imediatamente. Escolher essa opção pode causar uma interrupção em alguns casos.

7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify DB Instance (Modificar instância de banco de dados) para salvar suas alterações.

   Ou escolha Back (Voltar) para editar as alterações ou Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para associar um grupo de parâmetros de banco de dados a uma instância de banco de dados, use o comando da AWS CLI `modify-db-instance` com as seguintes opções:

- `--db-instance-identifier`
- `--db-parameter-group-name`

O exemplo a seguir associa o grupo de parâmetros de banco de dados `mydbpg` à instância de banco de dados `database-1`. As alterações são aplicadas imediatamente usando `--apply-immediately`. Use `--no-apply-immediately` para aplicar alterações durante a próxima janela de manutenção.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

API do RDS

Para associar um grupo de parâmetros de banco de dados a uma instância de banco de dados, use a operação da API `ModifyDBInstance` do RDS com os seguintes parâmetros:

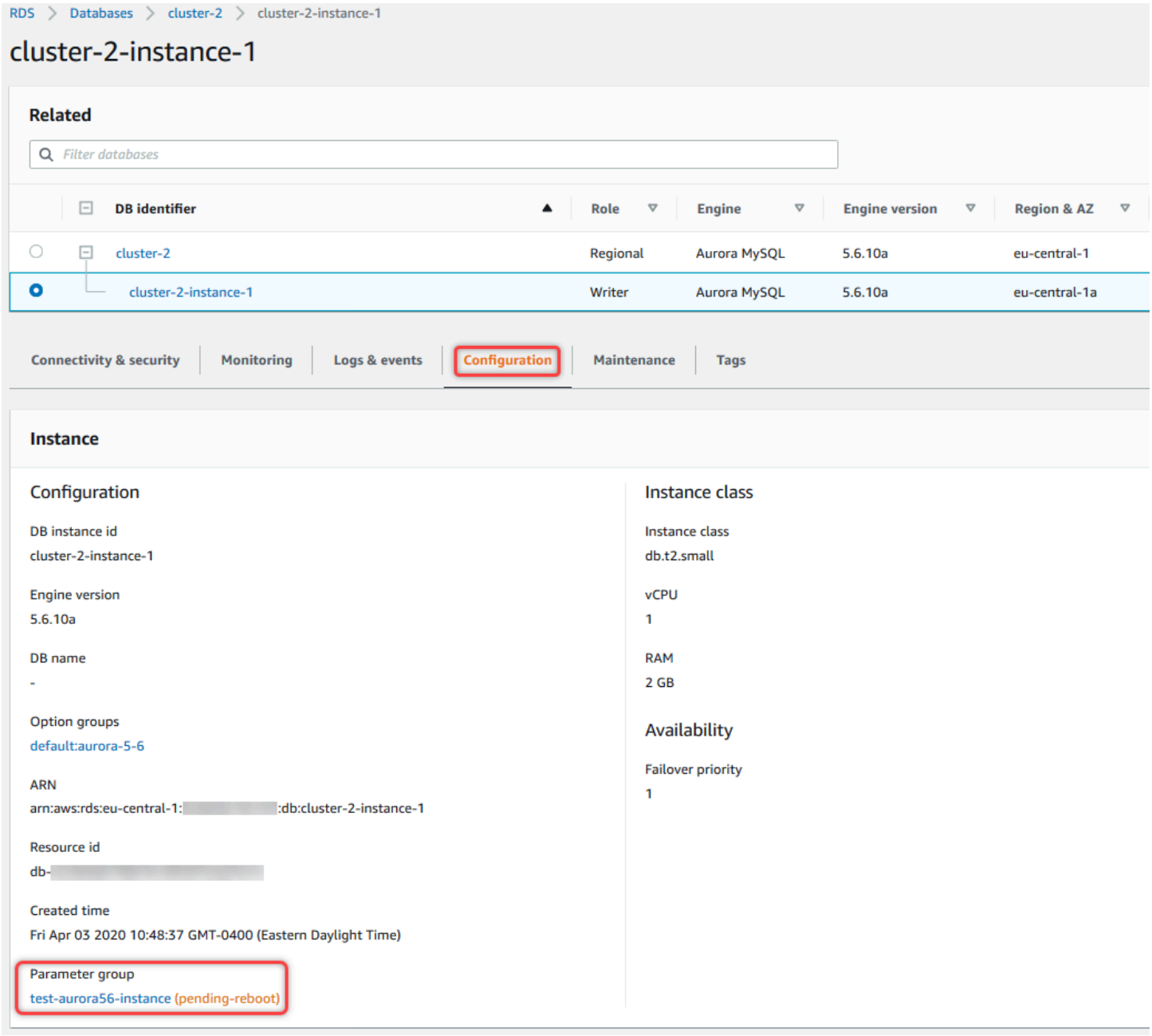
- `DBInstanceName`
- `DBParameterGroupName`

Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora

Você pode modificar valores de parâmetros em um grupo de parâmetros de banco de dados criado pelo cliente, mas não pode alterar os valores dos parâmetros em um grupo de parâmetros de banco de dados padrão. As alterações em parâmetros em um grupo de parâmetros de banco de dados criado pelo cliente são aplicadas a todas as instâncias de bancos de dados que estão associadas a esse grupo de parâmetros.

Existem dois tipos de parâmetros: dinâmicos e estáticos. As alterações em parâmetros dinâmicos são aplicadas à instância de banco de dados imediatamente sem uma reinicialização. As alterações feitas em parâmetros estáticos só serão aplicadas depois que a instância de banco de dados for reinicializada.

O console do RDS mostra o status do grupo de parâmetros de banco de dados associado a uma instância de banco de dados na guia Configuration (Configuração). Por exemplo, se a instância de banco de dados não estiver usando as últimas alterações do grupo de parâmetros de banco de dados associado, o console do RDS mostrará o grupo de parâmetros de banco de dados com um status de pending-reboot (reinicialização pendente). Para aplicar as alterações de parâmetro mais recentes a essa instância de banco de dados, reinicialize-a manualmente.



Console

Como modificar os parâmetros em um grupo de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Parameter groups (Grupos de parâmetros).
3. Na lista, selecione o grupo de parâmetros que você deseja modificar.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Edit (Editar).

5. Altere os valores dos parâmetros que você deseja modificar. Você pode percorrer os parâmetros usando as teclas de seta no canto superior direito da caixa de diálogo.

   Não altere valores em um grupo de parâmetros padrão.

6. Escolha Save changes (Salvar alterações).

AWS CLI

Para copiar um grupo de parâmetros de banco de dados, use o comando AWS CLI`modify-db-parameter-group` com as seguintes opções obrigatórias:

- `--db-parameter-group-name`
- `--parameters`

O exemplo a seguir modifica os valores `max_connections` e `max_allowed_packet` no grupo de parâmetros de banco de dados chamado mydbparametergroup.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

O comando produz uma saída como a seguinte:

```
DBPARAMETERGROUP  mydbparametergroup
```

API do RDS

Para copiar um grupo de parâmetros de banco de dados, use a operação da API `ModifyDBParameterGroup` do RDS com os seguintes parâmetros obrigatórios:

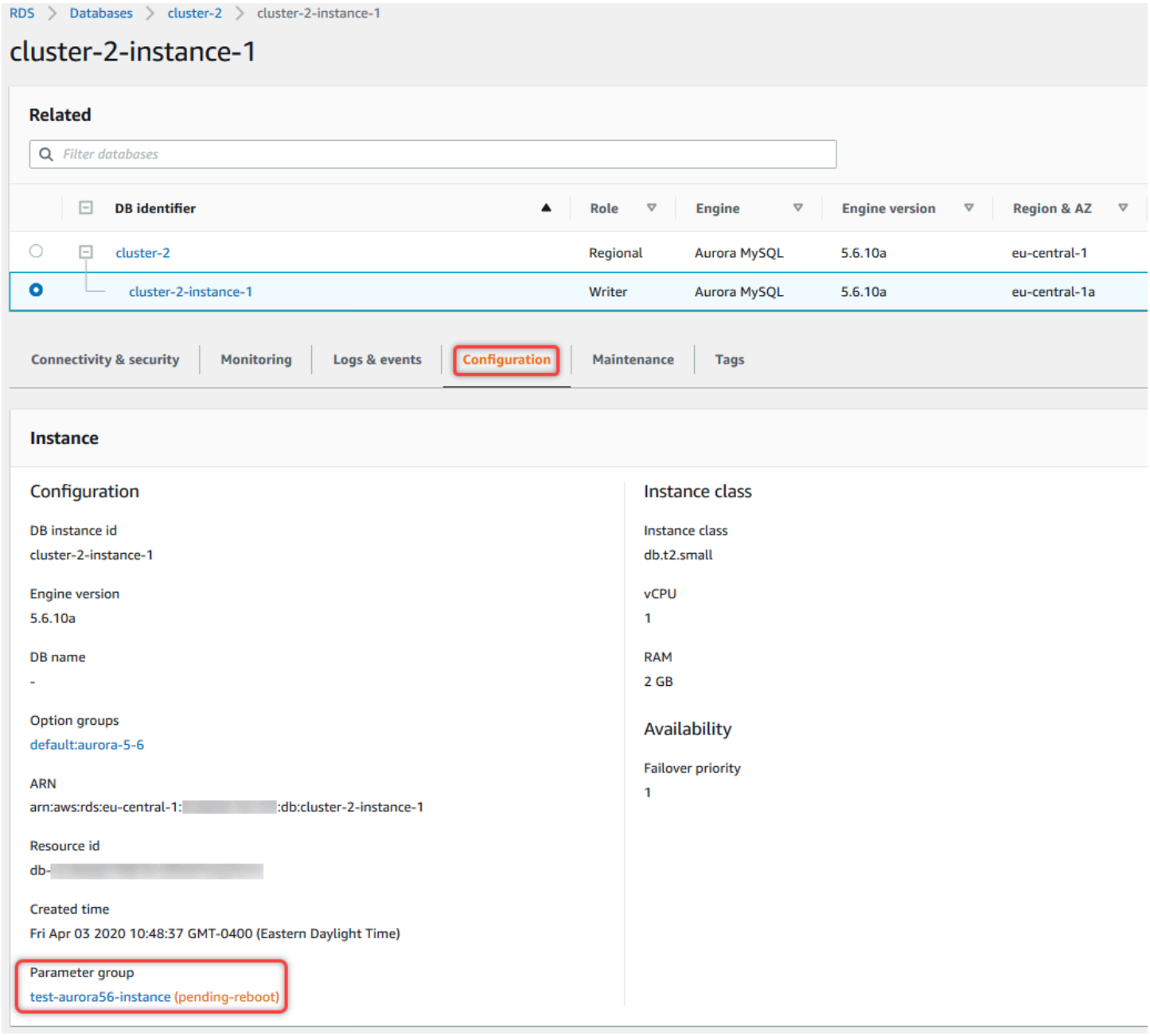
- `DBParameterGroupName`
- `Parameters`

Redefinir parâmetros em um grupo de parâmetros de banco de dados como seus valores padrão no Amazon Aurora

Você pode redefinir valores de parâmetro em um grupo de parâmetros de banco de dados criado pelo cliente para seus valores padrão. As alterações em parâmetros em um grupo de parâmetros de banco de dados criado pelo cliente são aplicadas a todas as instâncias de bancos de dados que estão associadas a esse grupo de parâmetros.

Ao usar o console, você pode redefinir parâmetros específicos como seus valores padrão. No entanto, não é possível redefinir facilmente todos os parâmetros no grupo de parâmetros de banco de dados de uma só vez. Ao usar A AWS CLI ou a API do RDS, você pode redefinir parâmetros específicos como seus valores padrão. Você também pode redefinir todos os parâmetros no grupo de parâmetros de banco de dados de uma só vez.

As alterações em alguns parâmetros são aplicadas à instância de banco de dados imediatamente sem uma reinicialização. As alterações feitas em outros parâmetros só serão aplicadas depois que a instância de banco de dados no cluster de banco de dados for reinicializada. O console do RDS mostra o status do grupo de parâmetros de banco de dados associado a uma instância de banco de dados na guia Configuration (Configuração). Por exemplo, suponha que a instância de banco de dados não esteja usando as últimas alterações do grupo de parâmetros de banco de dados associado. Se sim, o console do RDS mostrará o grupo de parâmetros de banco de dados com o status pending-reboot. Para aplicar as alterações de parâmetro mais recentes a essa instância de banco de dados, reinicialize-a manualmente.



> **Note**
>
> Em um grupo de parâmetros de banco de dados padrão, os parâmetros são sempre definidos para seus valores padrão.

Console

Para redefinir parâmetros em um grupo de parâmetros de banco de dados para seus valores padrão

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Edit (Editar).
5. Escolha os parâmetros que você deseja redefinir para os valores padrão. Você pode percorrer os parâmetros usando as teclas de seta no canto superior direito da caixa de diálogo.

   Não redefina os valores em um grupo de parâmetros padrão.
6. Escolha Reset (Redefinir) e, em seguida, confirme escolhendo Reset parameters (Redefinir parâmetros).

AWS CLI

Para redefinir alguns ou todos os parâmetros em um grupo de parâmetros de banco de dados, use o comando AWS CLI da `reset-db-parameter-group` com a seguinte opção necessária: `--db-parameter-group-name`.

Para redefinir todos os parâmetros no grupo de parâmetros de banco de dados, especifique a opção `--reset-all-parameters`. Para redefinir parâmetros específicos, especifique a opção `--parameters`.

O exemplo a seguir redefine todos os parâmetros no grupo de parâmetros de banco de dados chamado mydbparametergroup para seus valores padrão.

Example

Para Linux, macOS ou Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Para Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

O exemplo a seguir redefine as pções `max_connections` e `max_allowed_packet` para seus valores padrão no grupo de parâmetros de banco de dados chamado mydbparametergroup.

Example

Para Linux, macOS ou Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

O comando produz uma saída como a seguinte:

```
DBParameterGroupName  mydbparametergroup
```

API do RDS

Para redefinir parâmetros em um grupo de parâmetros de banco de dados para seus valores padrão, use o comando `ResetDBParameterGroup` da API do RDS com o seguinte parâmetro necessário: `DBParameterGroupName`.

Para redefinir todos os parâmetros no grupo de parâmetro de banco de dados, defina o parâmetro `ResetAllParameters` como `true`. Para redefinir parâmetros específicos, especifique o parâmetro `Parameters`.

Copiar um grupo de parâmetros de banco de dados no Amazon Aurora

Você pode copiar os grupos de parâmetros de banco de dados personalizados que criou. Copiar um grupo de parâmetros pode ser uma solução conveniente. Um exemplo é quando você criou

um grupo de parâmetros de banco de dados e quer incluir a maioria de seus parâmetros e valores personalizados em um novo grupo de parâmetros de banco de dados. Você pode copiar um grupo de parâmetros de banco de dados usando o Console de gerenciamento da AWS. Também é possível usar o comando copy-db-parameter-group da AWS CLI ou a operação da API CopyDBParameterGroup do RDS.

Depois de copiar um grupo de parâmetros de banco de dados, você deve esperar pelo menos 5 minutos antes de criar sua primeira instância de banco de dados que usa esse grupo de parâmetros de banco de dados como o padrão. Isso permite que o Amazon RDS conclua completamente a ação de cópia antes que o grupo de parâmetros seja usado. Isso é especialmente importante para parâmetros que são críticos ao criar o banco de dados padrão para uma instância de banco de dados. Um exemplo é o conjunto de caracteres para o banco de dados padrão definido pelo parâmetro `character_set_database`. Use opção Parameter Groups (Grupos de parâmetros) do console do Amazon RDS ou o comando describe-db-parameters para verificar se o seu grupo de parâmetros de banco de dados foi criado.

> **Note**
>
> Não é possível copiar um grupo de parâmetros padrão. No entanto, é possível criar um grupo de parâmetros com base em um grupo de parâmetros padrão.
> Não é possível copiar um grupo de parâmetros de banco de dados para uma Conta da AWS ou Região da AWS diferente.

Console

Para copiar um grupo de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros personalizado que você deseja copiar.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Copy (Copiar).
5. Em New DB parameter group identifier (Novo identificador do grupo de parâmetros do banco de dados), digite um nome para o novo grupo de parâmetros.
6. Em Description (Descrição), digite uma descrição para o novo grupo de parâmetros.
7. Escolha Copiar.

AWS CLI

Para copiar um grupo de parâmetros de banco de dados, use o comando da AWS CLI `copy-db-parameter-group` com as seguintes opções obrigatórias:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

O exemplo a seguir cria um novo grupo de parâmetros de banco de dados chamado `mygroup2` que é uma cópia do grupo de parâmetros de banco de dados `mygroup1`.

Example

Para Linux, macOS ou Unix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Para Windows:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

API do RDS

Para copiar um grupo de parâmetros de banco de dados, use a operação da API do RDS `CopyDBParameterGroup` com os seguintes parâmetros obrigatórios:

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Listar grupos de parâmetros de banco de dados no Amazon Aurora

Você pode listar os grupos de parâmetros de banco de dados que criou para sua conta da AWS.

> **Note**
>
> Grupos de parâmetros padrão são criados automaticamente a partir de um template de parâmetro padrão quando você cria uma instância de banco de dados para um mecanismo e uma versão de banco de dados específicos. Esses grupos de parâmetros padrão contêm configurações de parâmetros preferenciais e não podem ser modificados. Quando você cria um grupo de parâmetros personalizado, pode modificar as configurações desses parâmetros.

Console

Para listar todos os grupos de parâmetros de banco de dados para uma conta da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

   Os grupos de parâmetros de banco de dados aparecem em uma lista.

AWS CLI

Para listar todos os grupos de parâmetros de banco de dados de uma conta da AWS, use o comando AWS CLI da `describe-db-parameter-groups`.

Example

O exemplo a seguir lista todos os grupos de parâmetros de banco de dados para uma conta da AWS.

```
aws rds describe-db-parameter-groups
```

O comando retorna uma resposta como a seguinte:

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
```

```
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

O exemplo a seguir descreve o grupo de parâmetros mydbparamgroup1.

Para Linux, macOS ou Unix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Para Windows:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

O comando retorna uma resposta como a seguinte:

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

API do RDS

Para listar todos os grupos de parâmetros de banco de dados para uma conta da AWS, use a operação da API do RDS `DescribeDBParameterGroups`.

Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados no Amazon Aurora

Você pode obter uma lista de todos os parâmetros em um grupo de parâmetros de banco de dados e seus valores.

Console

Para visualizar os valores de parâmetros para um grupo de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

   Os grupos de parâmetros de banco de dados aparecem em uma lista.
3. Escolha o nome do grupo de parâmetros para ver sua lista de parâmetros.

AWS CLI

Para visualizar os valores de parâmetros para um grupo de parâmetros de banco de dados, use o comando AWS CLI da `describe-db-parameters` com o seguinte parâmetro obrigatório.

- `--db-parameter-group-name`

Example

O exemplo a seguir lista os parâmetros e os valores de parâmetros para um grupo de parâmetros de banco de dados chamado mydbparametergroup.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

O comando retorna uma resposta como a seguinte:

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                   engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768            system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string
 static      false
```

API do RDS

Para visualizar os valores de parâmetros para um grupo de parâmetros de banco de dados, use o comando `DescribeDBParameters` da API do RDS com o seguinte parâmetro necessário.

- `DBParameterGroupName`

Excluir um grupo de parâmetros de banco de dados no Amazon Aurora

É possível excluir um grupo de parâmetros de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Um grupo de parâmetros será elegível para exclusão somente se não estiver associado a uma instância de banco de dados.

Console

Como excluir um grupo de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.

   Os grupos de parâmetros de banco de dados aparecem em uma lista.
3. Escolha o nome dos grupos de parâmetros a serem excluídos.
4. Escolha Ações e então Excluir.
5. Revise os nomes dos grupos de parâmetros e escolha Excluir.

AWS CLI

Para excluir um grupo de parâmetros de banco de dados, use o comando `delete-db-parameter-group` da AWS CLI com o parâmetro exigido a seguir.

- `--db-parameter-group-name`

Example

O exemplo a seguir exclui um grupo de parâmetros de banco de dados chamado mydbparametergroup.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

API do RDS

Para excluir um grupo de parâmetros de banco de dados, use o comando `DeleteDBParameterGroup` da API do RDS com o parâmetro exigido a seguir.

- `DBParameterGroupName`

Comparação de grupos de parâmetros de banco de dados

É possível usar o Console de gerenciamento da AWS para visualizar as diferenças entre dois grupos de parâmetros de banco de dados.

Os grupos de parâmetros especificados devem ser grupos de parâmetros de banco de dados ou devem ser grupos de parâmetros de cluster de banco de dados. Isso se aplica mesmo se o mecanismo de banco de dados e a versão forem os mesmos. Por exemplo, não é possível comparar um grupo de parâmetros do banco de dados do `aurora-mysql8.0` (Aurora MySQL versão 3) e um grupo de parâmetros do cluster de banco de dados do `aurora-mysql8.0`.

Você pode comparar grupos de parâmetros de banco de dados do Aurora MySQL e do RDS para MySQL, mesmo para versões diferentes, mas não pode comparar grupos de parâmetros de banco de dados do Aurora PostgreSQL e do RDS para PostgreSQL.

Para comparar dois grupos de parâmetros de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha os dois grupos de parâmetros que você deseja comparar.

   > **Note**
   >
   > Para comparar um grupo de parâmetros padrão com um grupo de parâmetros personalizado, primeiro escolha o grupo de parâmetros padrão na guia Padrão e, depois, selecione o grupo de parâmetros personalizado na guia Personalizado.

4. Em Ações, selecione Comparar.

Especificação de parâmetros de banco de dados

Os tipos de parâmetros de banco de dados incluem os seguintes:

- Inteiro
- Booliano
- String
- Longo
- Double
- Timestamp
- Objeto de outros tipos de dados definidos

- Matriz de valores do tipo integer, boolean, string, long, double, timestamp ou object

Você também pode especificar parâmetros inteiros e booleanos usando expressões, fórmulas e funções.

Sumário

- Fórmulas de parâmetros de banco de dados
  - Variáveis de fórmulas de parâmetros de banco de dados
  - Operadores de fórmulas de parâmetros de banco de dados
- Funções de parâmetros de banco de dados
- Expressões de log de parâmetros de banco de dados
- Exemplos de valores de parâmetros de banco de dados

Fórmulas de parâmetros de banco de dados

Uma fórmula de parâmetro de banco de dados é uma expressão resolvida como um valor inteiro ou booliano. A expressão é delimitada por chaves: {}. você pode usar uma fórmula para um valor de parâmetro de banco de dados ou como um argumento para uma função de parâmetro de banco de dados.

Sintaxe

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

Variáveis de fórmulas de parâmetros de banco de dados

Cada variável de fórmula retorna um inteiro ou um valor booliano. Os nomes das variáveis diferenciam maiúsculas de minúsculas.

AllocatedStorage

Retorna um inteiro que representa o tamanho, em bytes, do volume de dados.

DBInstanceClassMemory

Retorna um número inteiro para o número de bytes de memória disponíveis para o processo do banco de dados. Esse número é calculado internamente começando com a quantidade total de memória para a classe de instância de banco de dados. A partir disso, o cálculo subtrai a memória reservada para o sistema operacional e os processos do RDS que gerenciam a instância. Portanto, o número é sempre um pouco menor do que as figuras de memória mostradas nas tabelas de classe de instância em Classes de instâncias de banco de dados do Amazon Aurora. O valor exato depende de uma combinação de fatores. São eles classe de instância, mecanismo de banco de dados e se ele se aplica a uma instância do RDS ou a uma instância que faça parte de um cluster do Aurora.

DBInstanceVCPU

Retorna um número inteiro representando a quantidade de unidades de processamento centrais virtuais (vCPUs) usadas pelo Amazon RDS para gerenciar a instância.

EndPointPort

Retorna um inteiro representando a porta usada ao se conectar à instância de banco de dados.

Réplica TrueIF

Retorna `1` se a instância de banco de dados é uma réplica de leitura e `0` se não é. Esse é o valor padrão do `read_only` parâmetro no Aurora MySQL.

Operadores de fórmulas de parâmetros de banco de dados

Fórmulas de parâmetros de banco de dados que oferecem suporte a dois operadores: divisão e multiplicação.

Operador de divisão: /

Divide o dividendo pelo divisor, retornando um quociente inteiro. Casas decimais no quociente são truncadas, não arredondadas.

Sintaxe

```
dividend / divisor
```

Os argumentos de dividendo e divisor devem ser expressões de inteiros.

Operador de multiplicação: *

Multiplica as expressões, retornando seu produto. As casas decimais nas expressões são truncadas, e não arredondadas.

Sintaxe

```
expression * expression
```

Ambas as expressões devem ser números inteiros.

Funções de parâmetros de banco de dados

Você especifica os argumentos das funções de parâmetro de banco de dados como inteiros ou fórmulas. Cada função deve ter pelo menos um argumento. Especifique vários argumentos como uma lista separada por vírgulas. A lista não pode ter membros vazios, como argument1,,argument3. Nomes de função não diferenciam maiúsculas de minúsculas.

IF

Retorna um argumento.

Sintaxe

```
IF(argument1, argument2, argument3)
```

Retorna o segundo argumento se o primeiro argumento é avaliado como verdadeiro. Retorna o terceiro argumento, caso contrário.

GREATEST

Retorna o maior valor de uma lista de números inteiros ou fórmulas de parâmetros.

Sintaxe

```
GREATEST(argument1, argument2,...argumentn)
```

Retorna um número inteiro.

LEAST

Retorna o menor valor de uma lista de números inteiros ou fórmulas de parâmetros.

Sintaxe

```
LEAST(argument1, argument2,...argumentn)
```

Retorna um número inteiro.

SUM

Adiciona os valores dos números inteiros ou fórmulas de parâmetros especificados.

Sintaxe

```
SUM(argument1, argument2,...argumentn)
```

Retorna um número inteiro.

Expressões de log de parâmetros de banco de dados

Você pode definir um valor de parâmetro de banco de dados inteiro como uma expressão de log. A expressão é delimitada por chaves: {}. Por exemplo:

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

A função `log` representa a base de log 2. Esse exemplo também usa a variável de fórmula `DBInstanceClassMemory`. Consulte Variáveis de fórmulas de parâmetros de banco de dados.

Exemplos de valores de parâmetros de banco de dados

Esses exemplos mostram o uso de fórmulas, funções e expressões para os valores de parâmetros de banco de dados.

> **Warning**
>
> Definir parâmetros indevidamente em um grupo de parâmetros de banco de dados pode ter efeitos adversos não intencionais. Estes podem incluir performance degradada e instabilidade do sistema. Sempre tenha cuidado ao modificar os parâmetros do banco de dados e faça backup dos seus dados antes de modificar seu grupo de parâmetros de banco de dados. Faça testes com alterações de grupos de parâmetros em instâncias de bancos de

dados de teste, criadas usando restaurações pontuais, antes de aplicar essas alterações às suas instâncias de banco de dados de produção.

Example uso da função de parâmetro de banco de dados LEAST (Menor)

Você pode especificar a função `LEAST` em um valor do parâmetro do Aurora MySQL `table_definition_cache`. Use-a para definir a quantidade de definições de tabela que podem ser armazenadas no cache de definição para o menor entre `DBInstanceClassMemory`/393040 ou 20.000.

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Migrar dados para um cluster de banco de dados do Amazon Aurora

Você tem várias opções para migrar dados do seu banco de dados existente para um cluster de banco de dados do Amazon Aurora, dependendo da compatibilidade do mecanismo de banco de dados. Suas opções de migração também dependem do banco de dados do qual você está migrando e do tamanho dos dados que você está migrando.

Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL

Você pode migrar dados de uma das origens a seguir para um cluster de banco de dados do Amazon Aurora MySQL.

- Uma instância de banco de dados RDS for MySQL
- Um banco de dados MySQL externo para o Amazon RDS
- Um banco de dados não compatível com MySQL

Para ter mais informações, consulte Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL.

Migrar dados para um cluster de banco de dados do Amazon Aurora PostgreSQL

Você pode migrar dados de uma das origens a seguir para um cluster de banco de dados do Amazon Aurora PostgreSQL.

- Uma instância de banco de dados PostgreSQL do Amazon RDS
- Um banco de dados incompatível com PostgreSQL

Para ter mais informações, consulte Migrar dados para o Amazon Aurora com compatibilidade com o PostgreSQL.

Criar um cache do Amazon ElastiCache usando as configurações do cluster de banco de dados do Aurora

O ElastiCache é um serviço de armazenamento em cache em memória totalmente gerenciado que fornece latências de leitura e gravação em microssegundos que são compatíveis com casos de uso flexíveis e em tempo real. O ElastiCache pode ajudar você a acelerar a performance de aplicações e bancos de dados. Você pode usar o ElastiCache como armazenamento de dados primário para casos de uso que não exigem durabilidade de dados, como tabelas de classificação de jogos, streaming e análise de dados. O ElastiCache ajuda a remover a complexidade associada à implantação e ao gerenciamento de um ambiente de cache distribuído. Para ter mais informações, consulte Common ElastiCache Use Cases and How ElastiCache Can Help para Memcached e Common ElastiCache Use Cases and How ElastiCache Can Help para Redis OSS. É possível usar o console do Amazon RDS para criar o cache do ElastiCache.

Você pode operar o Amazon ElastiCache em dois formatos. Você pode começar com um cache sem servidor ou optar por criar seu próprio cluster de cache. Se você optar por desenvolver seu próprio cluster de cache, saiba que o ElastiCache funciona com os mecanismos do Redis OSS e do Memcached. Se você não tiver certeza de qual mecanismo deseja usar, consulte Comparing Memcached and Redis OSS. Para ter mais informações sobre o Amazon ElastiCache, consulte o Guia do usuário do Amazon ElastiCache.

Tópicos

- Visão geral da criação do cache do ElastiCache com as configurações do cluster de banco de dados do Aurora
- Criar um cache do ElastiCache com as configurações de um cluster de banco de dados do Aurora

Visão geral da criação do cache do ElastiCache com as configurações do cluster de banco de dados do Aurora

É possível criar um cache do ElastiCache pelo Amazon RDS usando as mesmas configurações de um cluster de banco de dados do RDS recém-criado ou existente.

Alguns casos de uso para associar um cache do ElastiCache ao cluster de banco de dados:

- Você pode economizar custos e melhorar a performance usando o ElastiCache com o RDS em vez de executar somente no RDS.

- É possível usar o cache do ElastiCache como um datastore primário para aplicações que não exigem durabilidade de dados. As aplicações que usam Redis OSS ou o Memcached podem usar o ElastiCache sem praticamente nenhuma modificação.

Ao criar um cache do ElastiCache pelo RDS, esse cache herda as seguintes configurações do cluster de banco de dados do Aurora associado:

- Configurações de conectividade do ElastiCache
- Configurações de segurança do ElastiCache

Também é possível definir as configurações de cache de acordo com seus requisitos.

Configurar o ElastiCache nas aplicações

As aplicações devem ser configuradas para usar o cache do ElastiCache. Também é possível otimizar e melhorar o desempenho do cache configurando as aplicações para usar estratégias de armazenamento em cache, dependendo dos requisitos.

- Para acessar o cache do ElastiCache e começar, consulte Getting started with ElastiCache (Redis OSS) e Getting started with ElastiCache (Memcached).
- Para ter mais informações sobre estratégias de armazenamento em cache, consulte Caching strategies and best practices para Memcached e Caching strategies and best practices para Redis OSS.
- Para ter mais informações sobre alta disponibilidade nos clusters do ElastiCache (Redis OSS), consulte High availability using replication groups.
- É possível incorrer em custos associados a armazenamento de backup, transferência de dados dentro ou entre regiões ou uso do AWS Outposts. Para obter detalhes de preço, consulte Preço do Amazon ElastiCache.

Criar um cache do ElastiCache com as configurações de um cluster de banco de dados do Aurora

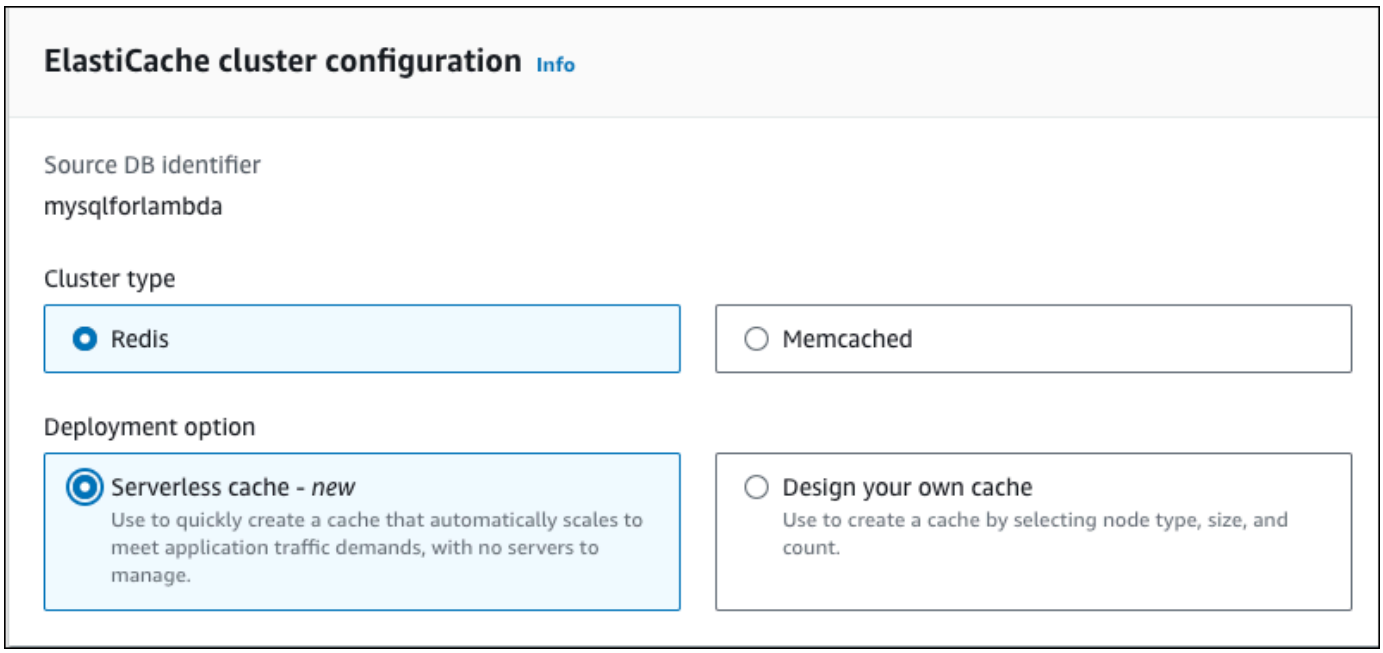
É possível criar um cache do ElastiCache para clusters de banco de dados do Aurora com configurações herdadas do cluster de banco de dados.

Criar um cache do ElastiCache com as configurações de um cluster de banco de dados

1. Para criar um cluster de banco de dados, siga as instruções em Criar um cluster de bancos de dados do Amazon Aurora.
2. Depois de criar um cluster de banco de dados do Aurora, o console exibe a janela Complementos sugeridos. Selecione Criar um cluster do ElastiCache do RDS usando suas configurações de banco de dados.

   Para um banco de dados existente, na página Bancos de dados, selecione o cluster de banco de dados. No menu suspenso Ações, selecione Criar cluster do ElastiCache para criar um cache do ElastiCache no RDS que tenha as mesmas configurações do cluster de banco de dados do Aurora existente.

   Na seção Configuração do ElastiCache, o Identificador de banco de dados de origem exibe de qual cluster de banco de dados o cache do ElastiCache herda as configurações.
3. Escolha se você deseja ou não criar um cluster do Redis OSS ou do Memcached. Para ter mais informações, consulte Comparing Memcached and Redis OSS.



4. Depois disso, escolha entre criar um Cache sem servidor ou Crie seu próprio cache. Consulte mais informações em Choosing between deployment options.

   Se você escolher Cache sem servidor:

   a. Em Configurações de cache, insira valores para Nome e Descrição.

b. Em Visualizar configurações padrão, deixe as configurações padrão para estabelecer a conexão entre o cache e o cluster de banco de dados.

c. Também é possível editar as configurações padrão escolhendo Personalizar as configurações padrão. Selecione as Configurações de conectividade do ElastiCache, as Configurações de segurança do ElastiCache e os Limites de uso máximos.

5. Se você escolher Crie seu próprio cache:

a. Se você selecionou Cluster do Redis OSS, escolha se deseja manter o modo de cluster Habilitado ou Desabilitado. Para ter mais informações, consulte Replication: Redis OSS (Cluster Mode Disabled) vs. Redis OSS (Cluster Mode Enabled).

b. Insira valores para Nome, Descrição e Versão do mecanismo.

Para Versão de mecanismo, o valor padrão recomendado é a versão mais recente do mecanismo. Você também pode escolher uma Versão do mecanismo para o cache do ElastiCache mais adequada para as seus requisitos.

c. Escolha o tipo de nó na opção Tipo de nó. Para ter mais informações, consulte Gerenciar nós.

Se você optar por criar um cluster do Redis OSS com o Modo de cluster definido como Habilitado, insira o número de fragmentos (partições/grupos de nós) na opção Número de fragmentos.

Insira o número de réplicas de cada fragmento em Número de réplicas.

> **Note**
>
> O tipo de nó selecionado, o número de fragmentos e o número de réplicas afetam o desempenho do cache e os custos de recursos. Essas configurações devem corresponder às necessidades de seu banco de dados. Para ter informações de preços, consulte Definição de preço do Amazon ElastiCache.

d. Selecione as Configurações de conectividade do ElastiCache e as Configurações de segurança do ElastiCache. É possível manter as configurações padrão ou personalizar essas configurações de acordo com seus requisitos.

6. Verifique as configurações padrão e herdadas do cache do ElastiCache. Algumas configurações não podem ser alteradas após a criação.

> **Note**
>
> O RDS pode ajustar a janela de backup do cache do ElastiCache para atender ao requisito mínimo de janela de 60 minutos. A janela de backup de seu banco de dados de origem permanece a mesma.

7. Quando estiver tudo pronto, selecione Criar cache do ElastiCache.

O console exibe um banner de confirmação para a criação do cache do ElastiCache. Siga o link no banner para o console do ElastiCache para ver os detalhes do cache. O console do ElastiCache exibe o cache do ElastiCache recém-criado.

Migração automática de bancos de dados do EC2 para o Amazon Aurora usando o AWS Database Migration Service

Você pode usar o console do Aurora para migrar um banco de dados do EC2 para o Aurora. O Aurora usa o AWS Database Migration Service (AWS DMS) para migrar seu banco de dados do EC2 de origem. O AWS DMS permite que você migre bancos de dados relacionais para a Nuvem AWS. Para obter mais informações sobre o AWS Database Migration Service, consulte O que é AWS Database Migration Service? no Guia do Usuário doAWS Database Migration Service.

Para começar a migração, você deve criar um cluster de banco de dados ou uma equivalente do Aurora para o qual migrar os dados. Depois de criar o banco de dados de destino, você pode importar seu banco de dados do EC2 para ele. Para bancos de dados de origem menores que 1 TiB, essa ação de migração reduz o tempo e os recursos necessários para migrar seus dados para o Aurora .

Visão geral

O console do Aurora permite migrar bancos de dados do EC2 para bancos de dados equivalentes do Aurora. Você deve criar um banco de dados do Aurora para habilitar a migração pelo console.

Você pode migrar bancos de dados do EC2 para os seguintes mecanismos de banco de dados:

- MySQL
- PostgreSQL

O processo de migração inclui as seguintes etapas:

- Crie um banco de dados equivalente no Aurora. Para que os bancos de dados sejam equivalentes, eles devem ter o mesmo mecanismo de banco de dados e versões de mecanismo compatíveis. Eles também devem estar na mesma VPC. Para obter instruções sobre como criar um banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora .
- Escolha o tipo de replicação para seu banco de dados:
  - Migração de carga máxima: o Aurora copia o banco de dados de origem completo para o banco de dados de destino, criando novas tabelas no destino quando necessário.

> **Note**
>
> Essa opção causa uma interrupção no banco de dados do Aurora.

- Migração de carga máxima e captura de dados de alteração (CDC): semelhante à migração de carga máxima, com essa opção, o Aurora copia todo o banco de dados de origem para o banco de dados de destino. No entanto, após a migração de carga máxima, o Aurora aplica todas as alterações capturadas na origem ao banco de dados de destino. A captura de dados de alteração coleta alterações nos logs de banco de dados utilizando a API nativa do mecanismo do banco de dados.

  > **Note**
  >
  > Essa opção causa uma interrupção no banco de dados do Aurora.

- Captura de dados de alteração (CDC): use essa opção para manter seu banco de dados de destino disponível durante a migração. O Aurora migra as alterações em andamento no banco de dados de origem para o banco de dados de destino.

- O Aurora cria os recursos de rede necessários para facilitar a migração. Depois que o Aurora cria os recursos necessários, ele notifica você sobre os recursos criados e permite que inicie a transferência de dados.

  O tempo necessário para concluir a migração depende do tipo de replicação e do tamanho do banco de dados de origem.

Pré-requisitos

MySQL

Antes de começar a trabalhar com um banco de dados MySQL como origem, confirme se você tem os pré-requisitos a seguir. Esses pré-requisitos se aplicam às gerenciadas pela AWS.

Você deve ter uma conta do AWS DMS com o perfil Replication Admin. O perfil precisa dos seguintes privilégios:

- REPLICATION CLIENT: este privilégio é necessário apenas para tarefas de CDC. Em outras palavras, as tarefas somente com carga completa não requerem este privilégio.

- REPLICATION SLAVE: este privilégio é necessário apenas para tarefas de CDC. Em outras palavras, as tarefas somente com carga completa não requerem este privilégio.

O usuário do AWS DMS deve ter privilégios SELECT para as tabelas de origem designadas para replicação.

Conceda os seguintes privilégios se você usar avaliações de pré-migração específicas do MySQL.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

PostgreSQL

Antes de migrar dados de um banco de dados de origem PostgreSQL gerenciado pela AWS, faça o seguinte:

- É recomendável utilizar uma conta de usuário da AWS com as permissões mínimas necessárias para a instância de banco de dados PostgreSQL como a conta de usuário do endpoint de origem do PostgreSQL do AWS DMS. A utilização de uma conta mestre não é recomendada. A conta deve ter o perfil `rds_superuser` e o perfil `rds_replication`. O perfil `rds_replication` concede permissões para gerenciar slots lógicos e transmitir dados utilizando slots lógicos.

> **Note**
>
> Algumas transações do AWS DMS ficam ociosas por algum tempo antes que o mecanismo do DMS as utilize novamente. Com a utilização do parâmetro `idle_in_transaction_session_timeout` no PostgreSQL versões 9.6 e superior é possível fazer com que as transações ociosas atinjam o tempo limite e falhem.

Limitações

As seguintes limitações se aplicam ao processo de migração automática:

- O status do banco de dados de destino deve ser Disponível para iniciar a migração do banco de dados de origem.
- Ao migrar de um banco de dados de origem MySQL, sua conta do Aurora deve ter a função de Administrador de Replicação. Você também deve ter os privilégios adequados aplicados para essa função.
- A instância do EC2 e o banco de dados de destino devem estar na mesma VPC.
- Você não pode migrar o banco de dados do EC2 para os seguintes bancos de dados de destino ao usar a ação Migrar dados do banco de dados do EC2:
  - Aurora global database
  - Aurora Limitless database
  - Aurora Serverless v1
  - Bancos de dados com versão do MySQL anterior a 5.7
  - Bancos de dados com versão do PostgreSQL anterior a 10.4

Criação de recursos do IAM para migrações homogêneas

O Aurora usa AWS DMS para migrar seus dados. Para acessar os bancos de dados e migrar os dados, o AWS DMS cria um ambiente com tecnologia sem servidor para migrações de dados homogêneas. Nesse ambiente, o AWS DMS exige acesso ao emparelhamento de VPC, a tabelas de rotas, a grupos de segurança e a outros recursos da AWS. Além disso, o AWS DMS armazena os logs, as métricas e o progresso de cada migração de dados no Amazon CloudWatch. Para criar um projeto de migração de dados, o AWS DMS precisa acessar esses serviços.

Além disso, o AWS DMS requer acesso aos segredos que representam um conjunto de credenciais de usuário para autenticar a conexão de banco de dados para as conexões de origem e de destino.

> **Note**
>
> Ao usar a ação Migrar dados da instância do EC2, você pode usar o console do Aurora para gerar esses recursos do IAM. Ignore esta etapa se você estiver usando os recursos do IAM gerados pelo console.

Para esse processo, é necessário ter estes recursos do IAM:

Tópicos

- Criação de uma política do IAM para migrações de dados homogêneas
- Criação de um perfil do IAM para migrações de dados homogêneas
- Criação de uma política de acesso secreto e um perfil
- Criação de um perfil do IAM para AWS DMS para gerenciar a Amazon VPC

Criação de uma política do IAM para migrações de dados homogêneas

Nesta etapa, você cria uma política do IAM que fornece acesso aos recursos do Amazon EC2 e do CloudWatch ao AWS DMS. Crie um perfil do IAM e anexe uma política.

Como criar uma política do IAM para migrações de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Políticas.
3. Selecione Criar política.
4. Na página Criar política, escolha a guia JSON.
5. Cole a política o JSON a seguir no editor.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
```

```
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. Selecione Avançar: tags e Avançar: revisar.
7. Insira **`HomogeneousDataMigrationsPolicy`** para Nome* e escolha Criar política.

Criação de um perfil do IAM para migrações de dados homogêneas

Nesta etapa, você cria um perfil do IAM que fornece acesso ao AWS Secrets Manager, ao Amazon EC2 e ao CloudWatch.

Como criar um perfil do IAM para migrações de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Perfis.
3. Escolha Criar Perfil.
4. Na página Selecionar entidade confiável, em Tipo de entidade confiável, escolha Serviço da AWS. Em Casos de uso para outros serviços da AWS, escolha DMS.
5. Marque a caixa de seleção DMS e escolha Próximo.
6. Na página Adicionar permissões, escolha HomogeneousDataMigrationsPolicy criada anteriormente. Escolha Próximo.
7. Na página Nomear, revisar e criar, insira **`HomogeneousDataMigrationsRole`** em Nome do perfil e escolha Criar função.
8. Na página Perfis, insira **`HomogeneousDataMigrationsRole`** em Nome do perfil. Escolha HomogeneousDataMigrationsRole.
9. Na página HomogeneousDataMigrationsRole, escolha a guia Relações de confiança. Selecione Edit trust policy (Editar política de confiança).
10. Na página Editar política de confiança, cole o seguinte JSON no editor, substituindo o texto existente.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
```

```
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

No exemplo anterior, substitua *your_region* pelo nome da sua Região da AWS.

A política anterior baseada em recursos fornece às entidades principais do AWS DMS as permissões para executar tarefas de acordo com a política HomogeneousDataMigrationsPolicy gerenciada pelo cliente.

11. Escolha Atualizar política.

Criação de uma política de acesso secreto e um perfil

Siga os procedimentos abaixo para criar sua política de acesso secreto e seu perfil que permitem que o DMS acesse as credenciais do usuário para os bancos de dados de origem e destino.

Como criar a política de acesso secreto e o perfil, que permite que o Amazon RDS acesse o AWS Secrets Manager para acessar o segredo apropriado

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS Identity and Access Management (IAM) em https://console.aws.amazon.com/iam/.
2. Escolha Políticas e Criar política.
3. Escolha JSON e insira a política a seguir para ativar o acesso e a descriptografia do segredo.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName-ABCDEF"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

Aqui, *secret_arn* é o ARN do segredo, que é possível obter de `SecretsManagerSecretId` conforme apropriado, e *kms_key_arn* é o ARN da chave do AWS KMS que você está utilizando para criptografar o segredo, como no exemplo a seguir.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                       "kms:Decrypt",
                       "kms:DescribeKey"
                     ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> Se você utilizar a chave de criptografia padrão criada pelo AWS Secrets Manager, não precisará especificar as permissões do AWS KMS para *kms_key_arn*.
> Se quiser que a política forneça acesso aos dois segredos, basta especificar um objeto de recurso JSON adicional para o outro *secret_arn*.

4. Revise e crie a política com um nome amigável e uma descrição opcional.
5. Escolha Funções e Criar função.
6. Escolha Serviço da AWS como o tipo de entidade confiável.
7. Escolha DMS na lista de serviços como o serviço confiável e escolha Próximo: Permissões.
8. Pesquise e anexe a política que criada na etapa 4 e continue para adicionar quaisquer tags e revisar o perfil. Nesse ponto, edite as relações de confiança do perfil para utilizar a entidade principal do serviço Amazon RDS regional como a entidade confiável. Essa entidade principal tem o seguinte formato.

```
dms.region-name.amazonaws.com
```

Aqui, *region-name* é nome da sua região, como `us-east-1`. Desse modo, segue uma entidade principal do serviço Amazon RDS regional para essa região.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Criação de um perfil do IAM para AWS DMS para gerenciar a Amazon VPC

Você deve criar um perfil do IAM para AWS DMS para gerenciar as configurações de VPC para seus recursos. Esse perfil deve estar disponível para uma migração bem-sucedida.

Criação de **`dms-vpc-role`** para migração de banco de dados

<result>

Isso cria o perfil para o DMS para gerenciar as configurações de VPC para a migração.
</result>

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console, escolha Roles (Perfis) e, em seguida, clique em Create role (Criar perfil).
3. Selecione a opção AWS service (Serviço da AWS) para Selecionar entidade confiável.

   Em Caso de uso, selecione DMS.
4. Na etapa Adicionar permissões, selecione `AmazonDMSVPCManagementRole` e escolha Próximo.
5. Na página Nomear, revisar e criar, defina o Nome do perfil como `dms-vpc-role` e escolha Criar perfil.

Configurar a migração de dados para o banco de dados do EC2

Para começar a migrar dados do banco de dados de origem do EC2, é necessário criar um banco de dados do Aurora equivalente. Para obter instruções sobre como criar um banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora .

Depois de criar o banco de dados de destino, use as etapas a seguir para configurar a migração de dados:

Configurar o projeto de migração de dados

1. Selecione o banco de dados de destino na página Bancos de dados no console do RDS.
2. Escolha o menu suspenso Ações e selecione a opção Migrar dados do banco de dados do EC2. Para ver os bancos de dados de destino compatíveis, consulte Limitações.
3. Na seção Selecionar banco de dados de origem do EC2:

   1. Verifique o Tipo de mecanismo e certifique-se de que seja o mesmo do banco de dados de origem.

      Além disso, verifique se as versões do mecanismo são compatíveis.
   2. Em Instância do EC2, escolha a instância do EC2 onde o banco de dados de origem reside.
   3. Em Porta, insira a porta na qual o banco de dados de origem permite tráfego.
   4. Em Segredo, escolha Criar e usar um novo segredo se você não tiver um segredo existente. Insira o Nome de usuário e a Senha do banco de dados de origem. Escolha também a chave do KMS com a qual criptografar seu segredo.

Para você já tiver um segredo, selecione Usar um segredo existente e escolha um segredo no menu suspenso.

5. Em Perfil do IAM para segredo, se você tiver um perfil do IAM existente, selecione Usar um perfil do IAM existente e escolha um perfil do IAM no menu suspenso que possa acessar o ID do segredo da etapa anterior.

   Se você não tem um perfil do IAM, escolha Criar e usar novo perfil do IAM. Insira o novo nome para o perfil no campo Nome do perfil do IAM. Você pode ver as permissões associadas a essa função no link abaixo.

4. Na seção Visualizar banco de dados de destino do RDS:

   1. Confirme as configurações do banco de dados de destino na parte superior da seção.
   2. Em Segredo, escolha Criar e usar um novo segredo se você não tiver um segredo existente que contenha as credenciais do banco de dados de destino.

      Para você já tiver um segredo, selecione o segredo no menu suspenso.
   3. Em Perfil do IAM para segredo, selecione um perfil do IAM que possa acessar o segredo da etapa anterior. Você também pode criar um perfil do IAM se já não tiver um.

      Se o menu suspenso não preencher os perfis do IAM, especifique o ARN do perfil do IAM no formato `arn:aws:iam:`*`account_id`*`:role/`*`roleName`*.

5. Na seção Configurar migração de dados:

   1. Selecione o tipo de migração de dados escolhendo entre Carga máxima, Captura de dados de alteração e carga máxima (CDC) ou Captura de dados de alteração (CDC). Para obter mais informações sobre essas opções, consulte Visão geral.

      Não é possível modificar o tipo de migração após o início da migração.
   2. Em Perfil do IAM para migração de dados, se você já tiver um perfil do IAM, selecione Usar um perfil do IAM existente e escolha um perfil do IAM no menu suspenso que conceda ao DMS as permissões para criar os recursos necessários para a migração. Se você não tem um perfil do IAM, escolha Criar e usar novo perfil do IAM.

6. Confirme se a guia Exibir configurações de migração mostra as configurações necessárias para que a migração de dados seja configurada com sucesso.
7. Selecione Migrar para concluir a configuração da migração.

Depois de concluir essas etapas, você pode ver os recursos que estão sendo configurados para a migração de dados escolhendo Visualizar detalhes no banner de progresso no console. Depois que os recursos necessários estiverem configurados, a migração será iniciada automaticamente. Se você criar

Para migrar vários bancos de dados para o banco de dados de destino, inicie esse processo novamente com detalhes sobre o novo banco de dados do EC2.

Gerenciar migrações de dados

Depois de usar a ação Migrar dados do banco de dados do EC2 do console do RDS, o Aurora inicia a migração automaticamente.

Se você tiver usado o console do AWS DMS para criar os recursos de migração, poderá iniciar o processo de migração.

Início da migração de dados

Siga estas etapas para iniciar a migração de dados:

Início de uma migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.
2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. Na guia Migrações de dados, a opção Migrações de dados associadas lista as migrações de dados disponíveis.

   As migrações configuradas usando o console do Aurora são iniciadas automaticamente quando os recursos necessários são configurados.

   As migrações configuradas usando o console do DMS são definidas como Prontas.

   Para começar essas migrações, selecione o menu suspenso Ações e selecione Iniciar.
4. Isso inicia a migração de dados para seu banco de dados do EC2.

Interrupção da migração de dados

Para migrações de dados cujo tipo de replicação é de carga completa, interromper a migração faz com que o processo pare e não possa ser retomado. Depois de interrompida, a migração deve ser reiniciada.

Para migrações com o tipo de replicação definido para captura de dados de alteração (CDC) ou carga total e CDC, você pode interromper o processo de replicação contínua e retomar o processo posteriormente.

Interrupção de uma migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.
2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. Na guia Migrações de dados, a opção Migrações de dados associadas lista as migrações de dados em andamento.

   Para interromper uma migração, selecione uma migração de dados e selecione Parar no menu suspenso Ações.
4. Isso para a migração de dados para seu banco de dados do EC2.

Retomada da migração de dados

Para migrações de dados cujo tipo de replicação é carga total e captura de dados de alteração (CDC) ou migração de captura de dados de alteração (CDC), você pode retomar o processo CDC a partir do último ponto de interrupção.

Retomada de uma migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.
2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. Na guia Migrações de dados, a opção Migrações de dados associadas lista as migrações de dados interrompidas.

   Para retomar uma migração, selecione uma migração de dados e selecione Retomar processamento no menu suspenso Ações.
4. Isso retoma a migração de dados para seu banco de dados do EC2.

Exclusão da migração de dados

Para excluir uma migração de dados associada, use as instruções a seguir

Exclusão de uma migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.
2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. Para excluir uma migração, selecione uma migração de dados e selecione Excluir no menu suspenso Ações.
4. Isso exclui a migração de dados.

A exclusão de uma migração de dados que estava em andamento não afeta nenhum dado que já tenha sido carregado no banco de dados de destino.

Reinicialização da migração de dados

Para reiniciar uma migração de dados associada a partir de um ponto inicial do CDC, use as instruções a seguir

Reinicialização de uma migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.
2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. Para reiniciar uma migração, selecione uma migração de dados e escolha Reiniciar no menu suspenso Ações.
4. Isso reinicia a migração de dados a partir de um ponto inicial do CDC.

A reinicialização de uma migração de dados que estava em andamento não afeta nenhum dado que já tenha sido carregado no banco de dados de destino.

Monitoramento das migrações de dados

Depois de iniciar migrações de dados, é possível monitorar seu status e progresso. As migrações de dados de grandes conjuntos de dados demoram horas para serem concluídas. Para manter a confiabilidade, a disponibilidade e o alto desempenho da migração de dados, monitore o progresso regularmente.

Como verificar o status e o progresso da migração de dados

1. Escolha o banco de dados de destino na página Bancos de dados no console do RDS.

2. Na página de detalhes do banco de dados, escolha a guia Migrações de dados.
3. A seção Migrações de dados associadas lista suas migrações de dados. Verifique a coluna Status.
4. Para migrações de dados em execução, a coluna Processo da migração exibe a porcentagem de dados migrados.
5. Para monitorar o processo no CloudWatch, use o link na coluna CloudWatch.

Status das migrações

Para cada migração de dados executada, o console do Aurora exibe o Status. A lista a seguir inclui os status:

- `Ready`: a migração de dados está pronta para ser iniciada.
- `Starting`: o Aurora está criando o ambiente com tecnologia sem servidor para a migração de dados.
- `Load running`: o Aurora está executando a migração de carga máxima.
- `Load complete, replication ongoing`: o Aurora concluiu a carga máxima e agora replica as alterações em andamento. Esse status só se aplica a migrações de carga máxima e do tipo CDC.
- `Replication ongoing`: o Aurora está replicando as alterações em andamento. Esse status só se aplica a migrações do tipo CDC.
- `Stopping`: o Aurora está interrompendo as migrações de dados. Esse status se aplica quando você opta por interromper a migração de dados no menu Ações.
- `Stopped`: o Aurora interrompeu as migrações de dados.
- `Failed`: falha na migração de dados. Para obter mais informações, consulte os arquivos de log.
- `Restarting`: a migração de dados reiniciou uma replicação de dados em andamento a partir de um ponto inicial do CDC.

Tutorial: criar um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado

Neste tutorial, é necessário criar um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado. Para ter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

> **Important**
>
> Não há cobrança para criar uma conta da AWS. No entanto, ao concluir este tutorial, é possível gerar custos para os recursos da AWS que você usa. Se esses recursos não forem mais necessários após a conclusão do tutorial, você poderá excluí-los.

Para criar um cluster de banco de dados com configurações e definições personalizadas, é possível usar grupos de parâmetros personalizados. Os grupos de parâmetros personalizados são particularmente úteis se você trabalha com vários bancos de dados e quer definir as configurações de modo uniforme para eles.

Ao concluir essas etapas, você aprende a:

- Usar o Amazon Aurora para criar um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado.
- Como usar parâmetros específicos para clusters de banco de dados do MySQL.

Para concluir este tutorial, realize as seguintes tarefas:

1. Crie um grupo de parâmetros do cluster de banco de dados com o parâmetro `default_password_lifetime` do MySQL.
2. Crie um cluster de banco de dados do MySQL com o grupo de parâmetros personalizado do cluster de banco de dados que você criou.

Tópicos

- Pré-requisitos
- Criar um grupo de parâmetros do cluster de banco de dados do Amazon Aurora
- Modificar o valor do parâmetro em seu grupo de parâmetros personalizado

- Criar um cluster de banco de dados do MySQL com um grupo de parâmetrosdo cluster de banco de dados

Pré-requisitos

Este tutorial exige que você tenha uma Conta da AWS e um usuário com acesso administrativo. Se ainda não os configurou, conclua as etapas apresentadas nas seguintes seções:

- Inscrever-se para uma Conta da AWS
- Criar um usuário com acesso administrativo

Criar um grupo de parâmetros do cluster de banco de dados do Amazon Aurora

Neste tutorial, você aprende a criar um grupo de parâmetros personalizado com default_password_lifetime para um cluster de banco de dados do MySQL no console. O parâmetro `default_password_lifetime` controla o número de dias antes da expiração automática da senha do cliente. Para ter mais informações sobre outros parâmetros disponíveis para cluster de banco de dados do MySQL, consulte Parâmetros de configuração do Aurora MySQL.

Para criar um parameter group

1. Abra o console do Amazon RDS e selecione Grupos de parâmetros.
2. Em Grupos de parâmetros personalizados, selecione Criar grupo de parâmetros.
3. Defina os detalhes do grupo de parâmetros.
    1. Insira um nome para o grupo de parâmetros.
    2. Insira uma descrição do grupo de parâmetros.
    3. Em Tipo de mecanismo, escolha Aurora MySQL.
    4. Em Família de grupos de parâmetros, selecione aurora-mysql8.0.
    5. Em Tipo, escolha Grupo de parâmetros do cluster de banco de dados.
4. Escolha Criar.

O novo grupo de parâmetros do cluster de banco de dados aparece na página Grupos de parâmetros no console do Amazon RDS. As etapas a seguir mostram como modificar valores de parâmetro para personalizar um grupo de parâmetros.

Modificar o valor do parâmetro em seu grupo de parâmetros personalizado

Use as etapas a seguir para modificar o valor do parâmetro no grupo de parâmetros que você criou em Criar um grupo de parâmetros do cluster de banco de dados do Amazon Aurora.

Como modificar os valores dos parâmetros em seu grupo de parâmetros

1. Abra o console do Amazon RDS e selecione Grupos de parâmetros.
2. Em Grupos de parâmetros personalizados, escolha o nome do grupo de parâmetros do cluster de banco de dados que você criou.
3. Escolha Editar.
4. Na caixa de pesquisa Filtrar parâmetros, pesquise o parâmetro personalizado `default_password_lifetime`.
5. Marque a caixa de seleção ao lado do parâmetro e insira um valor para o número de dias a ser definido para esse parâmetro de ciclo de vida da senha.
6. Escolha Salvar alterações.

O grupo de parâmetros personalizado agora está disponível para ser associado ao Amazon Aurora para clusters de banco de dados do MySQL 8.0.

Criar um cluster de banco de dados do MySQL com um grupo de parâmetrosdo cluster de banco de dados

Por fim, crie um cluster de banco de dados do MySQL com o grupo de parâmetros personalizado que você criou nas etapas anteriores. As etapas a seguir mostram como criar o cluster de banco de dados do MySQL com um grupo de parâmetros personalizado.

Como criar um cluster de banco de dados com um grupo de parâmetros personalizado e um novo grupo de opções

1. Abra o console do Amazon RDS e selecione Bancos de dados.
2. Selecione Criar banco de dados.
3. Em Escolher um método de criação de banco de dados, escolha Criação padrão.

4. Em Opções do mecanismo, escolha Aurora (compatível com MySQL).
5. Selecione Configuração adicional.
   - Em Nome do banco de dados inicial, selecione um nome para o cluster de banco de dados.
   - No menu suspenso Grupo de parâmetros do cluster de banco de dados, selecione o nome do grupo de parâmetros do cluster de banco de dados que você criou anteriormente.
6. Neste tutorial, é possível deixar as configurações padrão para qualquer outra configuração de banco de dados ou modificá-las conforme necessário.
7. Selecione Criar banco de dados.

O RDS cria um cluster de banco de dados do MySQL com um grupo de parâmetros personalizado. Para ver mais informações sobre esse banco de dados, consulte a página Bancos de dados do console do Amazon RDS.

Neste tutorial, você vai configurar um cluster de banco de dados do MySQL com configurações personalizadas usando um grupo de parâmetros personalizado. Esse cluster de banco de dados recém-criado do MySQL gerencia o ciclo de vida da senha do usuário usando o parâmetro `default_password_lifetime`. Para otimizar o banco de dados, é possível aplicar configurações adicionais no grupo de parâmetros personalizado e adicionar opções.

Assim que criar seu cluster de banco de dados personalizado, você deve excluir seus recursos para evitar custos indesejados. Para excluir um cluster de banco de dados, siga as instruções em Excluir clusters de banco de dados e instâncias de banco de dados do Aurora.

Como gerenciar um cluster de banco de dados do Amazon Aurora

Esta seção mostra como fazer gerenciamento e manutenção do cluster de banco de dados Aurora. O Aurora envolve clusters de servidores de banco de dados conectados em uma topologia de replicação. Assim, gerenciar o Aurora normalmente envolve implantar alterações em vários servidores e garantir que todas as réplicas do Aurora estejam acompanhando o servidor de origem. Como o Aurora dimensiona o armazenamento subjacente de forma transparente à medida que os dados crescem, gerenciar o Aurora exige relativamente pouco gerenciamento de armazenamento em disco. Da mesma forma, como o Aurora realiza backups contínuos automaticamente, um cluster do Aurora não exige planejamento extensivo ou tempo de inatividade para realizar backups.

Tópicos

- Interromper e iniciar um cluster de banco de dados do Amazon Aurora
- Conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora
- Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora
- Modificar um cluster de bancos de dados Amazon Aurora
- Adicionar réplicas do Aurora a um cluster de banco de dados
- Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora
- Clonar um volume para um cluster de banco de dados do Amazon Aurora
- Integração do Aurora com outros produtos da AWS
- Manutenção de um cluster de banco de dados do Amazon Aurora
- Reinicializar um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora
- Fazer failover de um cluster de banco de dados do Amazon Aurora
- Excluir clusters de banco de dados e instâncias de banco de dados do Aurora
- Marcar recursos do Amazon Aurora e do Amazon RDS
- Nomes de recurso da Amazon (ARNs) no Amazon RDS
- Atualizações do Amazon Aurora

Interromper e iniciar um cluster de banco de dados do Amazon Aurora

Interromper e iniciar os clusters de banco de dados do Aurora ajuda a gerenciar os custos dos ambientes de teste e de desenvolvimento. Você pode interromper temporariamente todas as instâncias de banco de dados em seu cluster, em lugar de configurar e destruir todas as instâncias cada vez que você usa o cluster.

Tópicos

- Visão geral de como interromper e iniciar um cluster de banco de dados do Aurora
- Limitações para interromper e iniciar clusters de banco de dados do Aurora
- Interromper um cluster de banco de dados do Aurora
- Operações possíveis quando um cluster de banco de dados do Aurora foi interrompido
- Iniciar um cluster de banco de dados do Aurora

Visão geral de como interromper e iniciar um cluster de banco de dados do Aurora

Durante os períodos em que não é necessário um cluster de banco de dados do Aurora, é possível interromper todas as instâncias nesse cluster simultaneamente. Você pode iniciar o cluster novamente a qualquer momento, sempre que precisar usá-lo. Iniciar e interromper simplifica os processos de configuração e destruição dos clusters usados em desenvolvimento, teste ou atividades afins que não exijam disponibilidade contínua. Você pode realizar todos os procedimentos do Console de gerenciamento da AWS envolvidos somente com uma ação única, independentemente de quantas instâncias estão no cluster.

Durante a interrupção do cluster de banco de dados, serão cobrados somente o armazenamento do cluster, os snapshots manuais e o armazenamento do backup automático dentro da janela de retenção especificada. Não haverá cobrança por horas de instância de banco de dados.

> **Important**
>
> Você pode interromper um cluster de banco de dados por até sete dias. Se você não iniciar o cluster de banco de dados manualmente após sete dias, ele será iniciado automaticamente para que não perca nenhuma atualização de manutenção necessária.

Para minimizar as cobranças para um cluster do Aurora levemente carregado, você pode interromper o cluster em vez de excluir todas as réplicas do Aurora dele. Para clusters com mais de uma ou duas instâncias, a frequente exclusão e recriação de instâncias de banco de dados só é prática usando a AWS CLI ou a API do Amazon RDS. Uma sequência de operações como essa também pode ser difícil de realizar na ordem correta. Por exemplo, a exclusão de todas as réplicas do Aurora antes da exclusão da instância primária para evitar a ativação do mecanismo de failover.

Evite iniciar e interromper se você precisar manter seu cluster de banco de dados em execução mas ele tiver mais capacidade do que o necessário. Se o cluster for muito caro ou não estiver muito ocupado, exclua uma ou mais instâncias de banco de dados ou altere todas as suas instâncias de banco de dados para uma classe de instância pequena. Não é possível interromper uma instância de banco de dados individual do Aurora.

O tempo para interromper o cluster de banco de dados varia de acordo com determinados fatores, como classes de instância de banco de dados, estado da rede, tipo de mecanismo de banco de dados e estado do banco de dados. O processo pode demorar alguns minutos. O serviço do Amazon RDS realiza as seguintes ações:

- Encerra os processos do mecanismo de banco de dados.
- Encerra os processos da plataforma do RDS.
- Encerra as instâncias do Amazon EC2 subjacentes.

O tempo para reiniciar o cluster de banco de dados varia de acordo com determinados fatores, como tamanho do banco de dados, classes de instância de banco de dados, estado da rede, tipo de mecanismo de banco de dados e estado do banco de dados quando o cluster foi encerrado. O processo de inicialização pode levar de minutos a horas, mas geralmente leva alguns minutos. Ao criar o plano de disponibilidade, recomendamos pensar na variabilidade do tempo de inicialização.

Para iniciar o cluster de banco de dados, o serviço executa ações como as seguintes:

- Provisiona as instâncias do Amazon EC2 subjacentes.
- Inicia os processos da plataforma do RDS.
- Inicia os processos do mecanismo de banco de dados.
- Recupera as instâncias de banco de dados (a recuperação ocorre mesmo após um desligamento normal).

Limitações para interromper e iniciar clusters de banco de dados do Aurora

Alguns cluster do Aurora não podem ser interrompidos e iniciados.

- Só é possível interromper e iniciar um cluster que faça parte de um banco de dados global do Aurora se ele for o único cluster no banco de dados global.
- Não é possível interromper e iniciar um cluster que tenha uma réplica de leitura entre regiões.
- Não é possível interromper e iniciar um cluster que faça parte de uma implantação azul/verde.
- Não é possível interromper e iniciar um cluster Aurora Serverless v1. Com o Aurora Serverless v2, não é possível interromper e iniciar um cluster .

Interromper um cluster de banco de dados do Aurora

Para usar um cluster de banco de dados do Aurora ou administrá-lo, você sempre deve começar com um cluster de banco de dados do Aurora em execução, depois interromper o cluster, e depois iniciá-lo novamente. Durante a interrupção do cluster, serão cobrados o armazenamento do cluster, os snapshots manuais e o armazenamento do backup automático dentro da janela de retenção especificada. As horas de instância de banco de dados não serão cobradas.

A operação de interrupção interrompe as instâncias de réplica do Aurora primeiro e, em seguida, a instância primária, para evitar a ativação do mecanismo de failover.

Console

Para interromper um cluster do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha um cluster. Você pode executar a operação de interrupção nesta página ou navegar até a página de detalhes do cluster de banco de dados que você deseja interromper.
3. Em Actions (Ações), escolha Stop temporarily (Parar temporariamente).
4. Na janela Stop DB cluster temporarily (Interromper temporariamente o cluster de banco de dados), selecione a confirmação de que o cluster de banco de dados será reiniciado automaticamente após sete dias.
5. Escolha Stop temporarily (Parar temporariamente) para interromper o cluster de banco de dados, ou escolha Cancel (Cancelar) para cancelar a operação.

AWS CLI

Para interromper uma instância de banco de dados usando a AWS CLI, chame o comando stop-db-cluster com os seguintes parâmetros:

- `--db-cluster-identifier` – o nome do cluster do Aurora.

Example

```
aws rds stop-db-cluster --db-cluster-identifier mydbcluster
```

API do RDS

Para interromper uma instância de banco de dados usando a API do Amazon RDS, chame a operação StopDBCluster com o seguinte parâmetro:

- `DBClusterIdentifier` – o nome do cluster do Aurora.

Operações possíveis quando um cluster de banco de dados do Aurora foi interrompido

Com um cluster de banco de dados do Aurora interrompido, você pode executar uma restauração point-in-time para qualquer ponto dentro da janela de retenção especificada para o backup automático. Para obter detalhes sobre como realizar uma restauração point-in-time, consulte Restaurar dados.

Você não poderá modificar a configuração de um cluster de banco de dados do Aurora, ou de qualquer uma de suas instâncias de banco de dados, enquanto o cluster estiver interrompido. Também não é possível adicionar ou remover instâncias de banco de dados do cluster ou excluir o cluster, se ele ainda tiver alguma instância de banco de dados associada. Você deverá iniciar o cluster antes de realizar uma dessas ações administrativas.

Parar um cluster de banco de dados remove ações pendentes, exceto para o grupo de parâmetro de cluster de banco de dados ou para os grupos de parâmetros de banco de dados das instâncias de cluster de banco de dados.

O Aurora aplicará as manutenções programadas em seu cluster interrompido depois que ele for reiniciado. Lembre-se de que após sete dias o Aurora inicia automaticamente qualquer cluster interrompido para não atrasar demais seu status de manutenção.

O Aurora também não realizará nenhum backup automático, pois os dados subjacentes não poderão ser alterados enquanto o cluster estiver interrompido. O Aurora não estende o período de retenção de backup enquanto o cluster está interrompido.

Iniciar um cluster de banco de dados do Aurora

Você deve sempre iniciar um cluster de banco de dados do Aurora começando com um cluster do Aurora que já esteja em estado interrompido. Quando você inicia o cluster, todas as suas instâncias de banco de dados ficam disponíveis novamente. O cluster mantém suas configurações, como endpoints, parameter groups e grupos de segurança da VPC.

A reinicialização do cluster de banco de dados normalmente leva alguns minutos.

Console

Para iniciar um cluster do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha um cluster. É possível executar a operação de início nesta página ou navegar até a página de detalhes do cluster de banco de dados que você deseja iniciar.
3. Em Actions (Ações), escolha Start (Iniciar).

AWS CLI

Para iniciar um cluster de banco de dados usando a AWS CLI, chame o comando start-db-cluster com os seguintes parâmetros:

- `--db-cluster-identifier` – o nome do cluster do Aurora. Esse nome é um identificador específico do cluster que você escolheu quando criou o cluster ou o DB instance identifier que você escolheu, com `-cluster` anexado no final.

Example

```
aws rds start-db-cluster --db-cluster-identifier mydbcluster
```

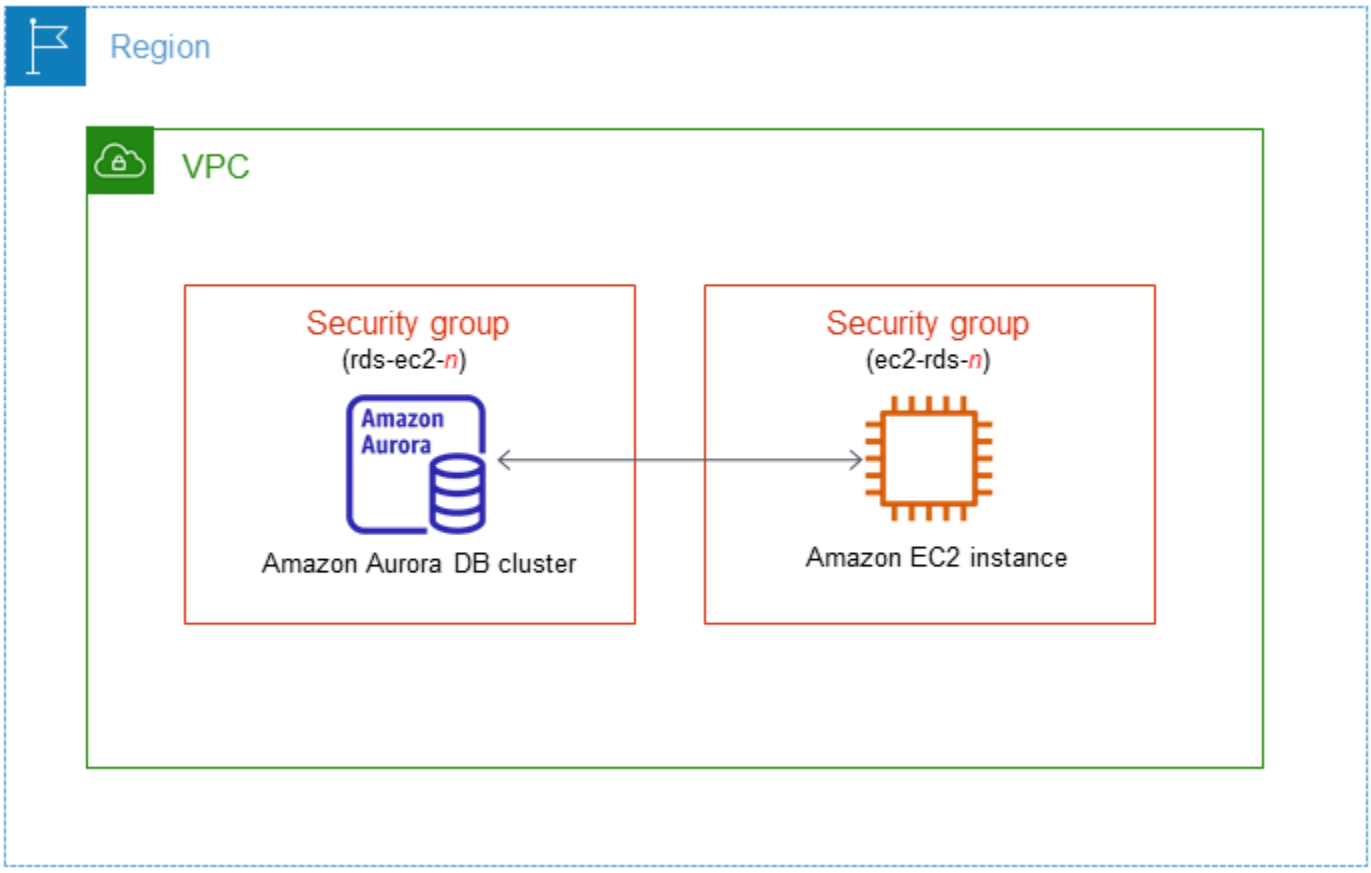
API do RDS

Para iniciar um cluster de banco de dados do Aurora usando a API do Amazon RDS, chame a operação StartDBCluster com o seguinte parâmetro:

- `DBCluster` – o nome do cluster do Aurora. Esse nome é um identificador específico do cluster que você escolheu quando criou o cluster ou o DB instance identifier que você escolheu, com `-cluster` anexado no final.

Conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora

Você pode usar o console do Amazon RDS para simplificar a configuração de uma conexão entre uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e um cluster de banco de dados do Aurora. Geralmente, seu cluster de banco de dados está em uma sub-rede privada, e sua instância do EC2 está em uma sub-rede pública em uma VPC. É possível utilizar um cliente SQL na instância do EC2 para se conectar ao cluster de banco de dados. A instância do EC2 também pode executar aplicações ou servidores que acessam seu cluster de banco de dados privado.



Se você quiser se conectar a uma instância do EC2 que não esteja na mesma VPC do cluster de banco de dados Aurora, veja os cenários em Cenários para acessar um cluster de banco de dados em uma VPC.

Tópicos

- Visão geral da conectividade automática com uma instância do EC2
- Conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora

- Visualizar recursos computacionais conectados
- Conectar a uma instância de banco de dados que está executando um mecanismo de banco de dados específico

Visão geral da conectividade automática com uma instância do EC2

Quando você configura uma conexão entre uma instância do EC2 e um cluster de banco de dados do Aurora, o RDS automaticamente configura o grupo de segurança da VPC para a instância do EC2 e para o cluster de banco de dados.

Confira a seguir os requisitos para conectar uma instância do EC2 ao cluster de banco de dados do Aurora:

- A instância do EC2 deve existir na mesma VPC do cluster de banco de dados.

  Se não houver nenhuma instância do EC2 na mesma VPC, o console fornecerá um link para que você crie uma.
- No momento, o cluster de banco de dados não pode ser um cluster de banco de dados do Aurora Serverless ou parte de um banco de dados global do Aurora.
- O usuário que configura a conectividade deve ter permissões para realizar as seguintes operações do Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Se a instância de banco de dados e a instância do EC2 estiverem em zonas de disponibilidade diferentes, sua conta poderá incorrer em custos entre as zonas.

Quando você configura uma conexão com uma instância do EC2, o Amazon RDS atua de acordo com a configuração atual dos grupos de segurança associados ao cluster de banco de dados e à instância do EC2, conforme descrito na tabela a seguir.

Configuração atual do grupo de segurança do RDS	Configuração atual do grupo de segurança do EC2	Ação do RDS
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n* (em que *n* é um número). Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-rds-`*n* (em que *n* é um número). Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem.	O RDS não realiza nenhuma ação. Uma conexão já foi configurada automaticamente entre a instância do EC2 e o cluster de banco de dados. Como já existe uma conexão entre a instância do EC2 e o banco de dados do RDS, os grupos de segurança não são modificados.
Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. • Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. No entanto,	Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado à instância do EC2 com um nome que corresponde ao padrão `ec2-rds-`*n*. • Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-rds-`*n*. No entanto, o Amazon RDS não pode usar	RDS action: create new security groups

Configuração atual do grupo de segurança do RDS	Configuração atual do grupo de segurança do EC2	Ação do RDS
o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com a instância do EC2. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da instância do EC2 como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado. São exemplos de modificação a adição de uma regra ou a alteração da porta de uma regra existente.	nenhum desses grupos de segurança para a conexão com o cluster de banco de dados. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	

Configuração atual do grupo de segurança do RDS	Configuração atual do grupo de segurança do EC2	Ação do RDS
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-rds-`*n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	RDS action: create new security groups
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Existe um grupo de segurança do EC2 válido para a conexão, mas ele não está associado à instância do EC2. Esse grupo de segurança tem um nome que corresponde ao padrão `ec2-rds-`*n*. Não foi modificado. Ele tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem.	RDS action: associate EC2 security group

Configuração atual do grupo de segurança do RDS	Configuração atual do grupo de segurança do EC2	Ação do RDS
Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. • Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome que corresponde ao padrão `rds-ec2-`*n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com a instância do EC2. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da instância do EC2 como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-rds-`*n*. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como origem.	RDS action: create new security groups

Ação do RDS: criar grupos de segurança

O Amazon RDS realiza as seguintes ações:

- Cria um grupo de segurança que corresponde ao padrão `rds-ec2-`*n*. Esse grupo de segurança tem uma regra de entrada com o grupo de segurança da VPC da instância do EC2 como origem. Esse grupo de segurança está associado ao cluster de banco de dados e permite que a instância do EC2 acesse o cluster de banco de dados.
- Cria um grupo de segurança que corresponde ao padrão `ec2-rds-`*n*. Esse grupo de segurança tem uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados como destino. Esse grupo de segurança está associado à instância do EC2 e permite que ela envie tráfego ao cluster de banco de dados.

Ação do RDS: associar o grupo de segurança do EC2

O Amazon RDS associa o grupo de segurança do EC2 existente e válido à instância do EC2. Esse grupo de segurança permite que a instância do EC2 envie tráfego ao cluster de banco de dados.

Conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora

Antes de configurar uma conexão entre uma instância do EC2 e um cluster de banco de dados do Aurora, atenda aos requisitos descritos em Visão geral da conectividade automática com uma instância do EC2.

Se você alterar esses grupos de segurança depois de configurar a conectividade, as alterações poderão afetar a conexão entre a instância do EC2 e o cluster de banco de dados do Aurora.

> **Note**
>
> Você só pode configurar automaticamente uma conexão entre uma instância do EC2 e um cluster de banco de dados do Aurora utilizando o Console de gerenciamento da AWS. Você não pode configurar uma conexão automaticamente com a AWS CLI nem a API do RDS.

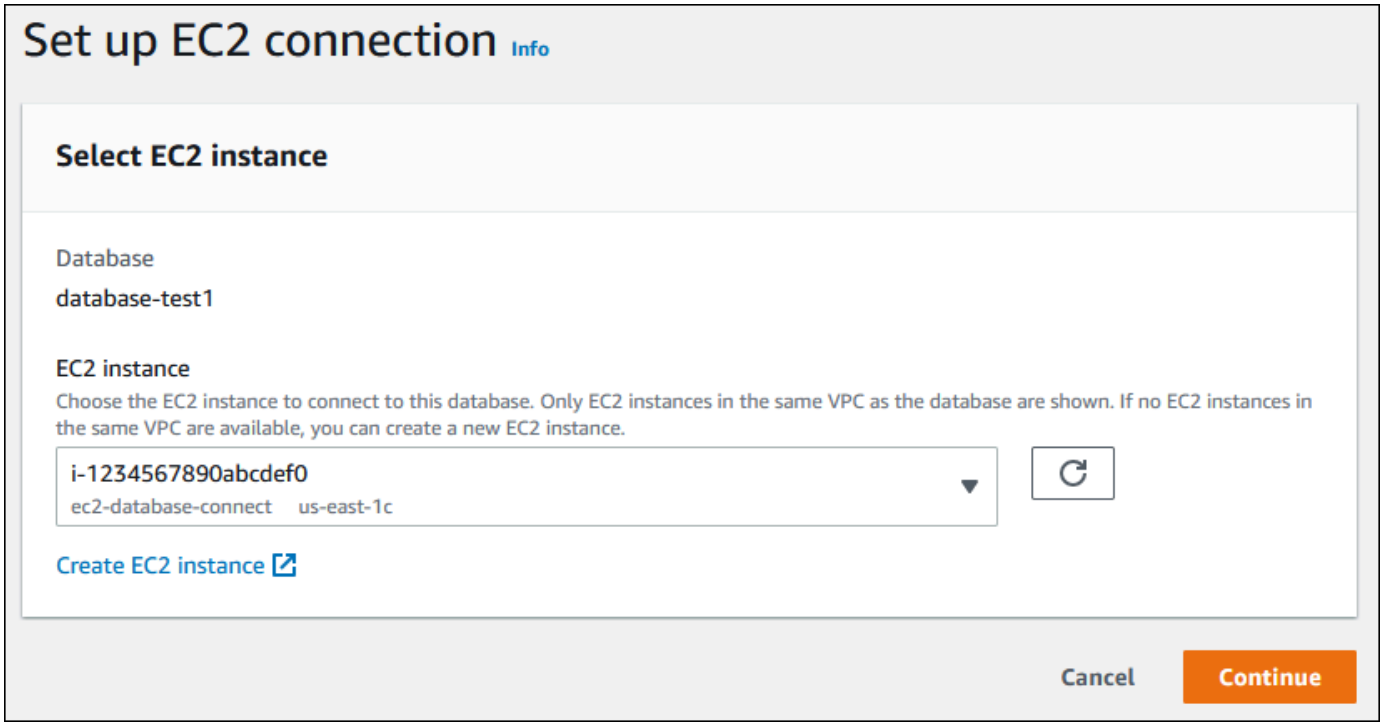
Como conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e depois selecione cluster de banco de dados.

3. Em Ações, selecione Configurar conexão do EC2.

   A página Set up EC2 connection (Configurar conexão do EC2) é exibida.

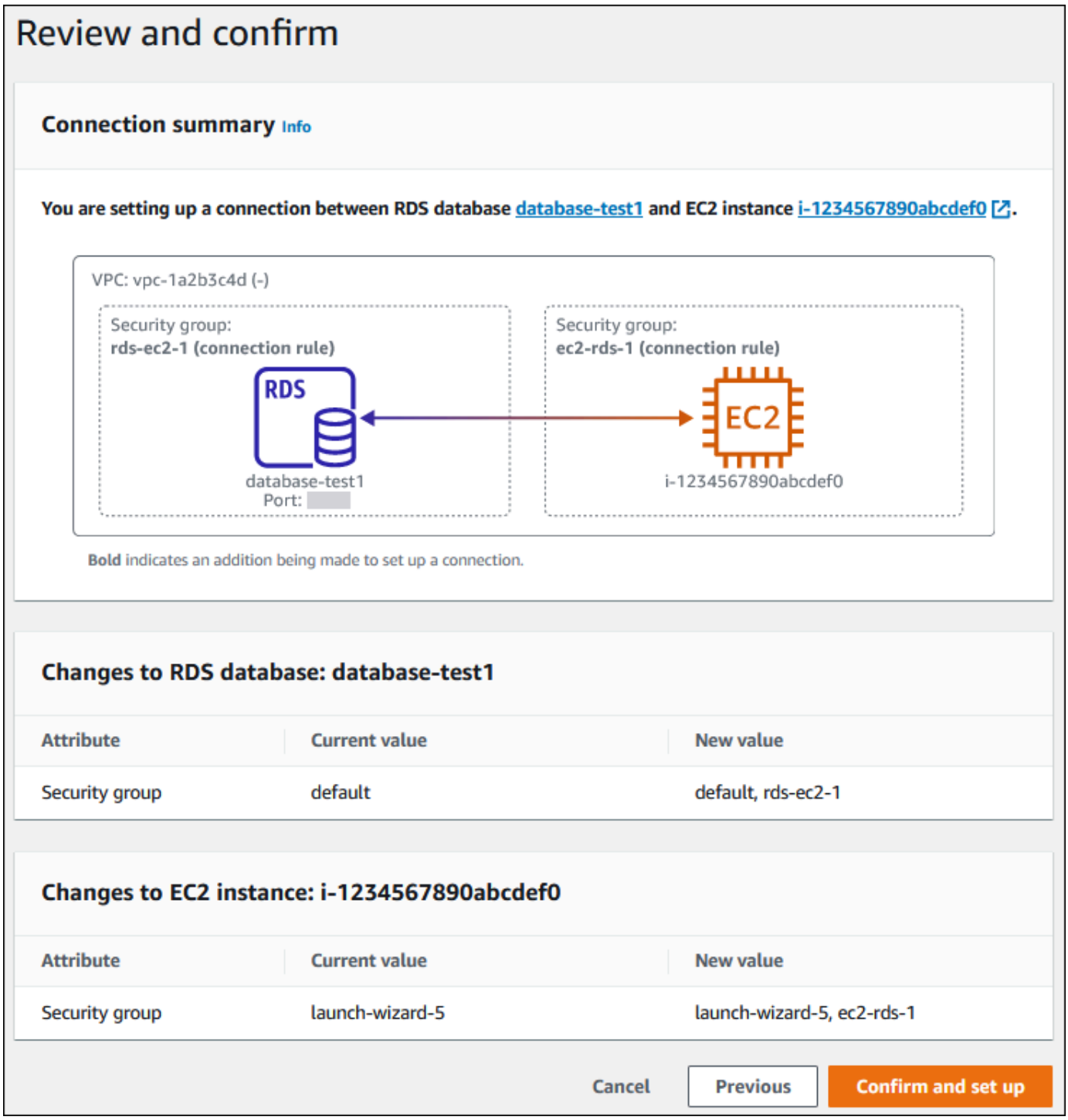
4. Na página Set up EC2 connection (Configurar conexão do EC2), selecione a instância do EC2.



   Se não houver nenhuma instância do EC2 na mesma VPC, selecione Create EC2 instance (Criar instância do EC2) para criar uma. Nesse caso, a nova instância do EC2 deve estar na mesma VPC do cluster de banco de dados.

5. Escolha Continuar.

   A página Review and confirm (Revisar e confirmar) é exibida.



6. Na página Review and confirm (Revisar e confirmar), analise as alterações que o RDS fará para configurar a conectividade com a instância do EC2.

   Se as alterações estiverem corretas, selecione Confirmar e configurar.

   Se as alterações não estiverem corretas, selecione Previous (Anterior) ou Cancel (Cancelar).

Visualizar recursos computacionais conectados

Você pode usar o Console de gerenciamento da AWS para visualizar os recursos computacionais conectados a um cluster de banco de dados Aurora. Os recursos mostrados incluem conexões de recursos computacionais que foram configuradas automaticamente. É possível configurar a conectividade com recursos computacionais automaticamente das seguintes maneiras:

- Você pode selecionar o recurso computacional ao criar o banco de dados.

  Para ter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.
- Você pode configurar a conectividade entre um banco de dados existente e um recurso computacional.

  Para ter mais informações, consulte Conectar automaticamente uma instância do EC2 e um cluster de banco de dados do Aurora.

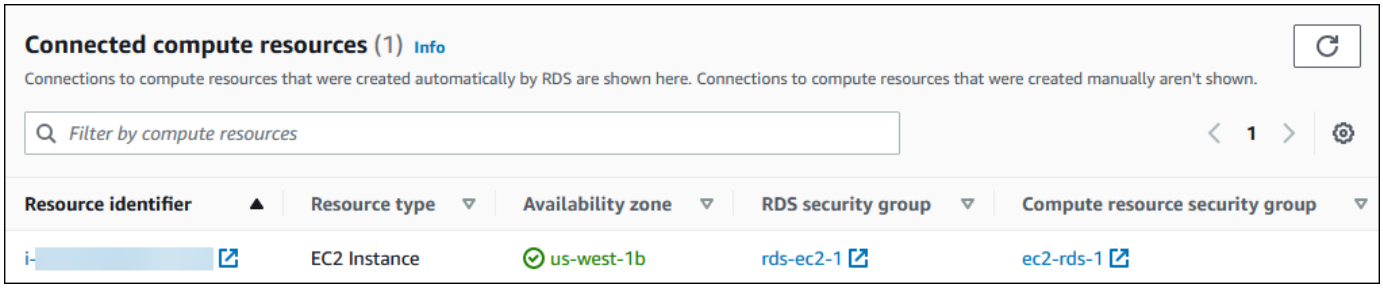
Os recursos computacionais listados não incluem aqueles que foram conectados manualmente ao banco de dados. Por exemplo, você pode permitir que um recurso computacional acesse um banco de dados manualmente adicionando uma regra ao grupo de segurança da VPC associado ao banco de dados.

Para que um recurso computacional seja listado, as seguintes condições devem ser atendidas:

- O nome do grupo de segurança associado ao recurso de computação corresponde ao padrão `ec2-rds-`*n* (em que *n* é um número).
- O grupo de segurança associado ao recurso de computação tem uma regra de saída com o intervalo de portas definido como a porta usada pelo cluster de banco de dados.
- O grupo de segurança associado ao recurso computacional tem uma regra de saída com o intervalo de portas definido como um grupo de segurança associado ao cluster de banco de dados.
- O nome do grupo de segurança associado ao cluster de banco de dados corresponde ao padrão `rds-ec2-`*n* (em que *n* é um número).
- O grupo de segurança associado ao cluster de banco de dados tem uma regra de entrada com o intervalo de portas definido como a porta usada pelo cluster de banco de dados.
- O grupo de segurança associado ao cluster de banco de dados tem uma regra de entrada com a fonte definida como um grupo de segurança associado ao recurso computacional.

Como visualizar os recursos de computação conectados a um cluster de banco de dados do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e depois selecione o nome do cluster de banco de dados.
3. Na guia Connectivity & security (Conectividade e segurança), veja os recursos computacionais em Connected compute resources (Recursos computacionais conectados).



Conectar a uma instância de banco de dados que está executando um mecanismo de banco de dados específico

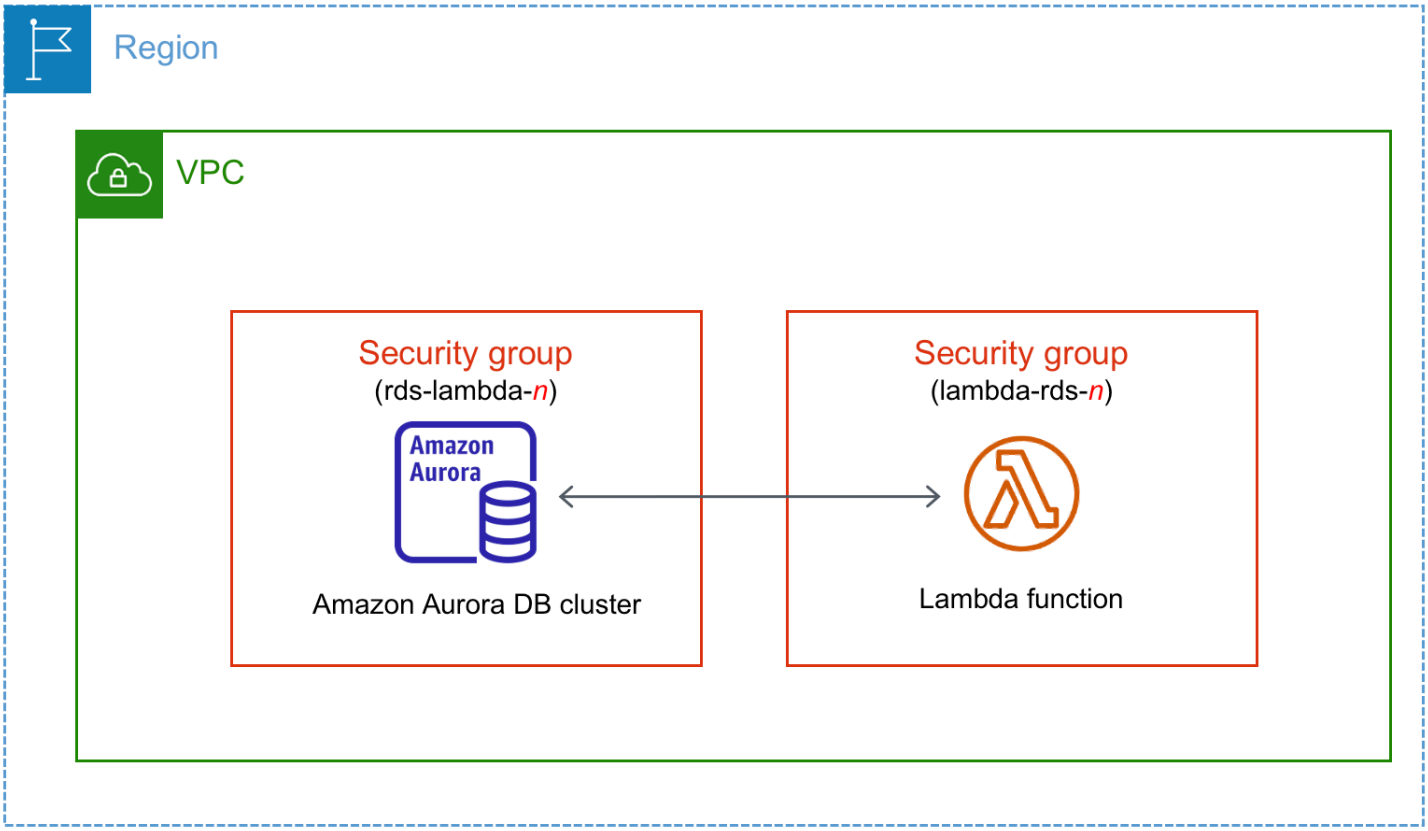
Para obter informações sobre a conexão a uma instância de banco de dados que esteja executando um mecanismo de banco de dados específico, siga as instruções do mecanismo de banco de dados:

- Como conectar-se a um cluster de bancos de dados Amazon Aurora MySQL
- Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL

Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora

Você pode usar o console do Amazon RDS para simplificar a configuração de uma conexão entre uma função do Lambda e um cluster de banco de dados do Aurora. Muitas vezes, o cluster de banco de dados está em uma sub-rede privada dentro de uma VPC. A função do Lambda pode ser usada por aplicações para acessar o cluster de banco de dados privado.

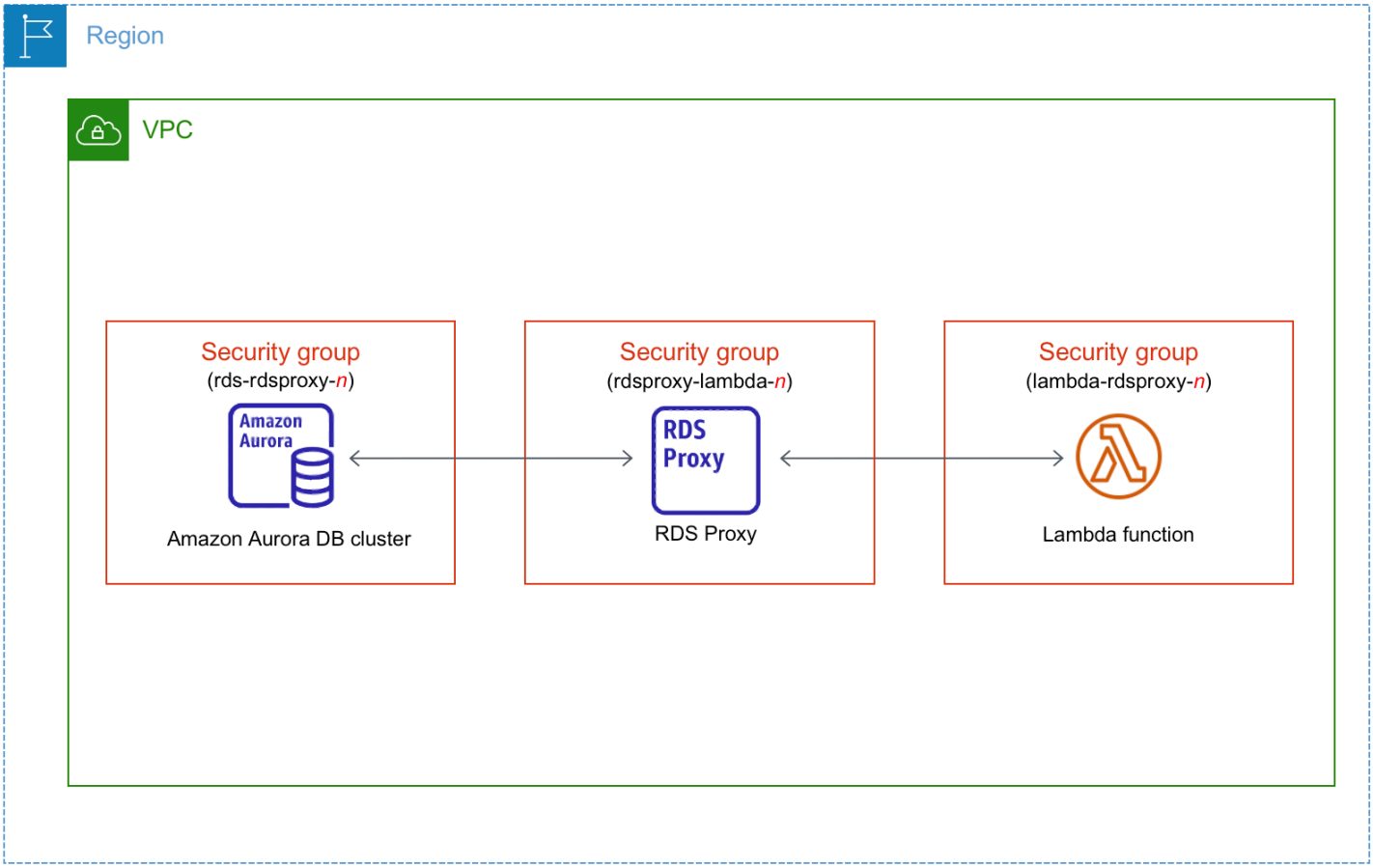
A imagem a seguir mostra uma conexão direta entre o cluster de banco de dados e a função do Lambda.



Você pode configurar a conexão entre a função do Lambda e o cluster de banco de dados por meio do RDS Proxy para melhorar a performance e a resiliência do banco de dados. Em geral, as funções do Lambda fazem conexões curtas frequentes com o banco de dados que se beneficiam do grupo de conexões oferecido pelo RDS Proxy. É possível aproveitar qualquer autenticação do AWS Identity and Access Management (IAM) que você já tenha para funções do Lambda, em vez de gerenciar credenciais de banco de dados no código de aplicação do Lambda. Para obter mais informações, consulte Amazon RDS Proxypara Aurora.

Quando você usa o console para se conectar a um proxy existente, o Amazon RDS atualiza o grupo de segurança do proxy para permitir conexões do cluster de banco de dados e a função do Lambda.

Você também pode criar um proxy na mesma página do console. Ao criar um proxy no console, para acessar o cluster de banco de dados, você deve inserir suas credenciais do banco de dados ou selecionar um segredo do AWS Secrets Manager.



Tip

Para conectar rapidamente uma função do Lambda a um cluster de banco de dados Aurora, você também pode usar o assistente guiado no console. Para abrir o assistente, faça o seguinte:

1. Abra a página Funções do console do Lambda.
2. Selecione a função à qual você deseja conectar um banco de dados.
3. Na guia Configuração, selecione bancos de dados do RDS.
4. Escolha Conectar ao banco de dados do RDS.

Depois de conectar a função a um banco de dados, você pode criar um proxy escolhendo Adicionar proxy.

Tópicos

- Visão geral da conectividade automática com uma função do Lambda
- Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora
- Visualizar recursos computacionais conectados

Visão geral da conectividade automática com uma função do Lambda

Confira abaixo os requisitos para conectar uma função do Lambda a um cluster de banco de dados do Aurora:

- A função do Lambda deve existir na mesma VPC que o cluster de banco de dados.
- No momento, o cluster de banco de dados não pode ser um cluster de banco de dados do Aurora Serverless ou parte de um banco de dados global do Aurora.
- O usuário que configura a conectividade deve ter permissões para realizar as seguintes operações do Amazon RDS, do Amazon EC2, do Lambda, do Secrets Manager e do IAM:
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBClusters`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBCluster`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`

  - `ec2:RevokeSecurityGroupEgress`
  - `ec2:RevokeSecurityGroupIngress`
- Lambda
  - `lambda:CreateFunctions`
  - `lambda:ListFunctions`
  - `lambda:UpdateFunctionConfiguration`
- Secrets Manager
  - `secretsmanager:CreateSecret`
  - `secretsmanager:DescribeSecret`
- IAM
  - `iam:AttachPolicy`
  - `iam:CreateRole`
  - `iam:CreatePolicy`
- AWS KMS
  - `kms:describeKey`

> **Note**
>
> Se o cluster de banco de dados e a função do Lambda estiverem em zonas de disponibilidade diferentes, sua conta poderá incorrer em custos entre as zonas.

Quando você configura uma conexão entre uma função do Lambda e um cluster de banco de dados do Aurora, o Amazon RDS configura o grupo de segurança da VPC para sua instância e para o cluster de banco de dados. Se você usa o RDS Proxy, o Amazon RDS também configura o grupo de segurança da VPC para o proxy. O Amazon RDS atua de acordo com a configuração atual dos grupos de segurança associados ao cluster de banco de dados, à função do Lambda e ao proxy, conforme descrito na tabela a seguir.

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se um proxy já estiver conectado ao cluster de banco de dados, o RDS conferirá se o `TargetHealth` de um proxy associado está `AVAILABLE`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de entrada com o grupo de segurança da VPC da função do Lambda ou do proxy como origem.	Há um ou mais grupos de segurança associados à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n* (em que *n* é um número). Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como destino.	Há um ou mais grupos de segurança associados ao proxy com um nome correspondente ao padrão `rdsproxy-lambda-` *n* (em que *n* é um número). Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem regras de entrada e saída com os grupos de segurança da VPC da função do Lambda e do cluster de banco de dados.	O Amazon RDS não realiza nenhuma ação. Uma conexão já foi configurada automaticamente entre a função do Lambda, o proxy (opcional) e o cluster de banco de dados. Como já existe uma conexão entre a função, o proxy e o banco de dados, os grupos de segurança não são modificados.
Qualquer uma das seguintes condições se aplica:	Qualquer uma das seguintes condições se aplica:	Qualquer uma das seguintes condições se aplica:	RDS action: create new security groups

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
• Não há nenhum grupo de segurança associado ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado está `AVAILABLE`. • Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado estiver `AVAILABLE`. No entanto, nenhum desses grupos de segurança pode ser usado para a	• Não há um grupo de segurança associado à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. • Há um ou mais grupos de segurança associados à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de	• Não há um grupo de segurança associado ao proxy com um nome correspondente ao padrão `rdsproxy-lambda-` *n*. • Há um ou mais grupos de segurança associados ao proxy com um nome correspondente ao `rdsproxy-lambda-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados ou a função do Lambda. O Amazon RDS não pode usar um grupo de segurança que não tenha regras de entrada e saída com o grupo de segurança	

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
conexão com a função do Lambda. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da função do Lambda ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado. São exemplos de modificação a adição de uma regra ou a alteração da porta de uma regra existente.	saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	da VPC do cluster de banco de dados ou da função do Lambda. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado estiver `AVAILABLE`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de entrada com o grupo de segurança da VPC da função do Lambda ou do proxy como origem.	Há um ou mais grupos de segurança associados à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	Há um ou mais grupos de segurança associados ao proxy com um nome correspondente ao padrão `rdsproxy-lambda-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados ou a função do Lambda. O Amazon RDS não pode usar um grupo de segurança que não tenha regras de entrada e saída com o grupo de segurança da VPC do cluster de banco de dados ou da função do Lambda. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	RDS action: create new security groups

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado estiver `AVAILABLE`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de entrada com o grupo de segurança da VPC da função do Lambda ou do proxy como origem.	Existe um grupo de segurança do Lambda válido para a conexão, mas ele não está associado à função do Lambda. Esse grupo de segurança tem um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Não foi modificado. Ele tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como destino.	Existe um grupo de segurança do proxy válido para a conexão, mas ele não está associado ao proxy. Esse grupo de segurança tem um nome que corresponde ao padrão `rdsproxy-lambda-` *n*. Não foi modificado. Ele tem regras de entrada e saída com os grupos de segurança da VPC do cluster de banco de dados e da função do Lambda.	RDS action: associate Lambda security group

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Qualquer uma das seguintes condições se aplica: • Não há nenhum grupo de segurança associado ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado está `AVAILABLE`. • Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado estiver `AVAILABLE`. No entanto, o	Há um ou mais grupos de segurança associados à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como destino.	Há um ou mais grupos de segurança associados ao proxy com um nome correspondente ao padrão `rdsproxy-lambda-` *n*. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem regras de entrada e saída com os grupos de segurança da VPC do cluster de banco de dados e da função do Lambda.	RDS action: create new security groups

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com a função do Lambda ou o proxy. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da função do Lambda ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.			

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Qualquer uma das seguintes condições se aplica: • Não há nenhum grupo de segurança associado ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado está `AVAILABLE`. • Há um ou mais grupos de segurança associados ao cluster de banco de dados com um nome correspondente ao padrão `rds-lambda-` *n* ou se o `TargetHealth` de um proxy associado estiver `AVAILABLE`. No entanto, o	Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. • Há um ou mais grupos de segurança associados à função do Lambda com um nome correspondente ao padrão `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados.	Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado ao proxy com um nome correspondente ao padrão `rdsproxy-lambda-` *n*. • Há um ou mais grupos de segurança associados ao proxy com um nome correspondente ao `rdsproxy-lambda-` *n*. No entanto, o Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com o cluster de banco de dados ou a função do Lambda. O Amazon RDS não pode usar um grupo	RDS action: create new security groups

Configuração atual do grupo de segurança do RDS	Configuração do grupo de segurança do Lambda atual	Configuração do grupo de segurança de proxy atual	Ação do RDS
Amazon RDS não pode usar nenhum desses grupos de segurança para a conexão com a função do Lambda ou o proxy. O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da função do Lambda ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	O Amazon RDS não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como origem. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	de segurança que não tenha regras de entrada e saída com o grupo de segurança da VPC do cluster de banco de dados ou da função do Lambda. O Amazon RDS também não pode usar um grupo de segurança que tenha sido modificado.	

Ação do RDS: criar grupos de segurança

O Amazon RDS realiza as seguintes ações:

- Cria um grupo de segurança que corresponde ao padrão `rds-lambda-`*n* ou `rds-rdsproxy-`*n* (se você optar por usar o RDS Proxy). Esse grupo de segurança tem uma regra de entrada com o grupo de segurança da VPC da função do Lambda ou do proxy como origem. Esse grupo de segurança está associado ao cluster de banco de dados e permite que a função ou o proxy acesse o cluster de banco de dados.

- Cria um grupo de segurança que corresponde ao padrão `lambda-rds-`*n* ou ao `lambda-rdsproxy-`*n*. Esse grupo de segurança tem uma regra de saída com o grupo de segurança da VPC do cluster de banco de dados ou do proxy como destino. Esse grupo de segurança está associado à função do Lambda e permite que a função envie tráfego para o cluster de banco de dados ou envie tráfego por meio de um proxy.
- Cria um grupo de segurança que corresponde ao padrão `rdsproxy-lambda-`*n*. Esse grupo de segurança tem regras de entrada e saída com os grupos de segurança da VPC do cluster de banco de dados e da função do Lambda.

Ação do RDS: associar o grupo de segurança do Lambda

O Amazon RDS associa o grupo de segurança do Lambda válido e existente à função do Lambda. Esse grupo de segurança permite que a função envie tráfego para o cluster de banco de dados ou envie tráfego por meio de um proxy.

Conectar automaticamente uma função do Lambda e um cluster de banco de dados do Aurora

Você pode usar o console do Amazon RDS para conectar automaticamente uma função do Lambda ao cluster de banco de dados. Isso simplifica o processo de configuração de uma conexão entre esses recursos.

Você também pode usar o RDS Proxy para incluir um proxy em sua conexão. As funções do Lambda fazem conexões curtas frequentes com o banco de dados que se beneficiam do grupo de conexões oferecido pelo RDS Proxy. Também é possível usar qualquer autenticação do IAM que você já tenha para funções do Lambda, em vez de gerenciar credenciais de banco de dados no código da aplicação do Lambda.

Você pode conectar um cluster de banco de dados existente a funções do Lambda novas e existentes usando a página Configurar conexão do Lambda. O processo de configuração define automaticamente os grupos de segurança necessários para você.

Antes de configurar uma conexão entre uma função do Lambda e um cluster de banco de dados, garanta que:

- Sua função do Lambda e o cluster de banco de dados estejam na mesma VPC.

- Você tenha as permissões corretas para sua conta do usuário. Para receber mais informações sobre os requisitos, consulte Visão geral da conectividade automática com uma função do Lambda.

Se você alterar os grupos de segurança depois de configurar a conectividade, as alterações poderão afetar a conexão entre a função do Lambda e o cluster de banco de dados.

> **Note**
>
> Você possa configurar automaticamente uma conexão entre um cluster de banco de dados e uma função do Lambda somente no Console de gerenciamento da AWS. Para conectar uma função do Lambda, todas as instâncias no cluster de banco de dados devem estar no estado Disponível.

Como conectar automaticamente uma função do Lambda e um cluster de banco de dados

<result>

Depois de confirmar a configuração, o Amazon RDS inicia o processo de conexão da função do Lambda, do RDS Proxy (se você usou um proxy) e do cluster de banco de dados. O console mostra a caixa de diálogo Detalhes da conexão, que lista as alterações do grupo de segurança que permitem conexões entre seus recursos.
</result>

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Bancos de dados e, depois, o cluster de banco de dados que você deseja conectar a uma função do Lambda.
3. Em Ações, selecione Configurar conexão do Lambda.
4. Na página Configurar conexão do Lambda, em Selecionar função do Lambda, faça o seguinte:
   - Se você tiver uma função do Lambda existente na mesma VPC que o cluster de banco de dados, selecione Selecionar função existente e, depois, escolha a função.
   - Se você não tiver uma função do Lambda na mesma VPC, selecione Criar função do Lambda e, depois, insira um Nome da função. O runtime padrão é definido como Nodejs.18. Você pode modificar as configurações de sua nova função do Lambda no console do Lambda depois de concluir a configuração da conexão.

5. (Opcional) Em RDS Proxy, selecione Conectar usando o RDS Proxy e, depois, faça o seguinte:
   - Se você tiver um proxy que deseja usar, escolha Selecionar um proxy existente e, depois, escolha o proxy.
   - Se você não tiver um proxy e quiser que o Amazon RDS crie um automaticamente para você, selecione Criar um proxy. Então, para Credenciais de banco de dados, faça o seguinte:

     a. Selecione Nome de usuário e senha do banco de dados e, depois, insira o Nome do usuário e a Senha para o cluster de banco de dados.

     b. Selecione Segredo do Secrets Manager. Então, em Selecionar segredo, escolha um segredo do AWS Secrets Manager. Se você não tiver um segredo do Secrets Manager, selecione Criar um segredo do Secrets Manager para criar um segredo. Depois de criar o segredo, em Selecionar segredo, escolha o novo segredo.

     Depois de criar o proxy, escolha Selecionar proxy existente e, depois, escolha o proxy. Observe que pode levar algum tempo até que o proxy esteja disponível para conexão.
6. (Opcional) Expanda Resumo da conexão e verifique as atualizações destacadas para seus recursos.
7. Escolha Configurar.

Visualizar recursos computacionais conectados

Você pode usar o Console de gerenciamento da AWS para visualizar as funções do Lambda que estão conectadas ao cluster de banco de dados. Os recursos mostrados incluem conexões de recursos de computação que o Amazon RDS configurou automaticamente.

Os recursos de computação listados não incluem aqueles que são conectados manualmente ao cluster de banco de dados. Por exemplo, você pode permitir que um recurso de computação acesse um cluster de banco de dados manualmente adicionando uma regra ao grupo de segurança da VPC associado ao banco de dados.

Para que o console liste uma função do Lambda, as seguintes condições devem ser aplicadas:

- O nome do grupo de segurança associado ao recurso de computação corresponde ao padrão `lambda-rds-`*n* ou `lambda-rdsproxy-`*n* (em que *n* é um número).

- O grupo de segurança associado ao recurso de computação tem uma regra de saída com o intervalo de portas definido como a porta do cluster de banco de dados ou de um proxy associado. O destino da regra de saída deve ser definido como um grupo de segurança associado ao cluster de banco de dados ou a um proxy associado.
- Se a configuração incluir um proxy, o nome do grupo de segurança anexado ao proxy associado ao banco de dados corresponderá ao padrão `rdsproxy-lambda-`*n* (em que *n* é um número).
- O grupo de segurança associado à função tem uma regra de saída com a porta definida como a porta que o cluster de banco de dados ou o proxy associado usa. O destino deve ser definido como um grupo de segurança associado ao cluster de banco de dados ou ao proxy associado.

Como visualizar os recursos de computação conectados automaticamente a um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Bancos de dados e escolha o cluster de banco de dados.
3. Na guia Conectividade e segurança, veja os recursos de computação em Recursos de computação conectados.

Modificar um cluster de bancos de dados Amazon Aurora

Você pode alterar as configurações de um cluster de banco de dados para realizar tarefas como alterar o período de retenção de backup ou a porta do banco de dados. Você também pode modificar instâncias de banco de dados em um cluster de banco de dados para realizar tarefas como alterar a classe da instância de banco de dados ou habilitar Insights de Performance para ela. Este tópico orienta você sobre como fazer modificações em um cluster de bancos de dados Aurora e nas instâncias de banco de dados, além de descrever as configurações de cada uma.

Recomendamos testar todas as alterações feitas em um cluster, ou instância, de banco de dados antes de modificá-los em seu ambiente de produção. Dessa maneira, você compreenderá completamente o impacto de cada alteração. Isso é especialmente importante ao atualizar as versões do banco de dados.

Tópicos

- Modificar o cluster de banco de dados usando o console, a CLI e a API
- Modificar uma instância de banco de dados em um cluster de banco de dados
- Alterar a senha do usuário mestre do banco de dados
- Configurações do Amazon Aurora
- Configurações não aplicáveis a clusters de banco de dados Amazon Aurora
- Configurações não aplicáveis a instâncias de banco de dados do Amazon Aurora

Modificar o cluster de banco de dados usando o console, a CLI e a API

Você pode modificar um cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Note**
>
> A maioria das modificações pode ser aplicada imediatamente ou durante a próxima janela de manutenção. Algumas modificações, como ativar a proteção contra exclusão, são aplicadas imediatamente, independentemente de quando você decide aplicá-las.
> A alteração da senha mestra no Console de gerenciamento da AWS é sempre aplicada imediatamente. No entanto, ao usar a AWS CLI ou a API do RDS, você pode escolher se deseja aplicar essa alteração imediatamente ou durante a próxima janela de manutenção programada.

Se você estiver usando endpoints SSL e alterar o identificador do cluster de banco de dados, pare e reinicie esse cluster para atualizar os endpoints SSL. Para obter mais informações, consulte Interromper e iniciar um cluster de banco de dados do Amazon Aurora.

Console

Para modificar um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha o cluster de banco de dados que você deseja modificar.
3. Escolha Modify (Modificar). A página Modify DB cluster (Modificar cluster de banco de dados) é exibida.
4. Altere qualquer uma das configurações desejadas. Para obter informações sobre cada configuração, consulte Configurações do Amazon Aurora.

   > **Note**
   >
   > No Console de gerenciamento da AWS, algumas alterações feitas no nível da instância só se aplicam à instância de banco de dados atual, enquanto outras se aplicam a todo o cluster de banco de dados. Para obter informações sobre se uma configuração se aplica à instância ou ao cluster de banco de dados, consulte o escopo da configuração em Configurações do Amazon Aurora. Para alterar uma configuração que modifica todo o cluster de banco de dados no nível de instância no Console de gerenciamento da AWS, siga as instruções em Modificar uma instância de banco de dados em um cluster de banco de dados.

5. Quando todas as alterações estiverem conforme o desejado, escolha Continue (Continuar) e verifique o resumo das modificações.
6. Para aplicar as alterações imediatamente, selecione Apply immediately (Aplicar imediatamente).
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify cluster (Modificar cluster) para salvar as alterações.

   Como alternativa, escolha Back (Voltar) para editar suas alterações ou escolha Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para modificar um cluster de banco de dados usando a AWS CLI, chame o comando modify-db-cluster. Especifique o identificador do cluster de banco de dados e os valores para as configurações que você deseja modificar. Para obter informações sobre cada configuração, consulte Configurações do Amazon Aurora.

> **Note**
>
> Algumas configurações são aplicáveis somente a instâncias de banco de dados. Para alterar essas configurações, siga as instruções em Modificar uma instância de banco de dados em um cluster de banco de dados.

Example

O comando a seguir modifica o `mydbcluster` configurando o período de retenção de backup como 1 semana (7 dias).

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 7
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 7
```

API do RDS

Para modificar um cluster de banco de dados usando a API do Amazon RDS, chame a operação ModifyDBCluster. Especifique o identificador do cluster de banco de dados e os valores para as configurações que você deseja modificar. Para obter informações sobre cada parâmetro, consulte Configurações do Amazon Aurora.

> **Note**
>
> Algumas configurações são aplicáveis somente a instâncias de banco de dados. Para alterar essas configurações, siga as instruções em Modificar uma instância de banco de dados em um cluster de banco de dados.

Modificar uma instância de banco de dados em um cluster de banco de dados

Você pode modificar uma instância em um cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Quando você modifica uma instância de banco de dados, pode aplicar as alterações imediatamente. Para aplicar as alterações imediatamente, selecione a opção Apply Immediately (Aplicar imediatamente) no Console de gerenciamento da AWS, use o parâmetro `--apply-immediately` ao chamar a AWS CLI ou configure o parâmetro `ApplyImmediately` como `true` ao usar a API do Amazon RDS.

Se você não optar por aplicar alterações imediatamente, as alterações serão adiadas até a próxima janela de manutenção. Durante a próxima janela de manutenção, qualquer uma dessas alterações adiadas é aplicada. Se você optar por aplicar alterações imediatamente, suas novas alterações e quaisquer alterações adiadas anteriormente serão aplicadas.

Para ver as modificações pendentes para a próxima janela de manutenção, use o comando describe-db-clusters da AWS CLI e selecione o campo `PendingModifiedValues`.

> **Important**
>
> Se alguma das modificações pendentes exigir tempo de inatividade, escolher Apply immediately (Aplicar imediatamente) pode causar um tempo de inatividade inesperado para a instância de banco de dados. Não há tempo de inatividade para as outras instâncias de banco de dados no cluster de banco de dados.
> As modificações adiadas não são listadas na saída do comando de CLI do `describe-pending-maintenance-actions`. As ações de manutenção incluem apenas atualizações do sistema que você programar para a próxima janela de manutenção.

Console

Para modificar uma instância de banco de dados em um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e selecione a instância de banco de dados que você deseja modificar.
3. Para Actions (Ações), selecione Modify (Modificar). A página Modify DB instance (Modificar instância de banco de dados) será exibida.
4. Altere qualquer uma das configurações desejadas. Para obter informações sobre cada configuração, consulte Configurações do Amazon Aurora.

   > **Note**
   >
   > Algumas configurações se aplicam a todo o cluster de banco de dados e devem ser alteradas no nível do cluster. Para alterar essas configurações, siga as instruções em Modificar o cluster de banco de dados usando o console, a CLI e a API.
   > No Console de gerenciamento da AWS, algumas alterações feitas no nível da instância só se aplicam à instância de banco de dados atual, enquanto outras se aplicam a todo o cluster de banco de dados. Para obter informações sobre se uma configuração se aplica à instância ou ao cluster de banco de dados, consulte o escopo da configuração em Configurações do Amazon Aurora.

5. Quando todas as alterações estiverem conforme o desejado, escolha Continue (Continuar) e verifique o resumo das modificações.
6. Para aplicar as alterações imediatamente, selecione Apply immediately (Aplicar imediatamente).
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify DB Instance (Modificar instância de banco de dados) para salvar suas alterações.

   Como alternativa, escolha Back (Voltar) para editar suas alterações ou escolha Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para modificar uma instância de banco de dados em um cluster de banco de dados usando a AWS CLI, chame o comando modify-db-instance. Especifique o DB instance identifier e os valores para as

configurações que você deseja modificar. Para obter informações sobre cada parâmetro, consulte Configurações do Amazon Aurora.

> **Note**
>
> Algumas configurações são aplicáveis somente ao cluster de banco de dados inteiro. Para alterar essas configurações, siga as instruções em Modificar o cluster de banco de dados usando o console, a CLI e a API.

Example

O código a seguir modifica `mydbinstance` configurando a classe da instância de banco de dados como `db.r4.xlarge`. As alterações serão aplicadas durante a janela de manutenção usando `--no-apply-immediately`. Use `--apply-immediately` para aplicar as alterações imediatamente.

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r4.xlarge \
    --no-apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.r4.xlarge ^
    --no-apply-immediately
```

API do RDS

Para modificar uma instância de banco de dados usando a API do Amazon RDS, chame a operação ModifyDBInstance. Especifique o DB instance identifier e os valores para as configurações que você deseja modificar. Para obter informações sobre cada parâmetro, consulte Configurações do Amazon Aurora.

> **Note**
>
> Algumas configurações são aplicáveis somente ao cluster de banco de dados inteiro. Para alterar essas configurações, siga as instruções em Modificar o cluster de banco de dados usando o console, a CLI e a API.

Alterar a senha do usuário mestre do banco de dados

É possível usar o Console de gerenciamento da AWS ou a AWS CLI para alterar a senha do usuário mestre.

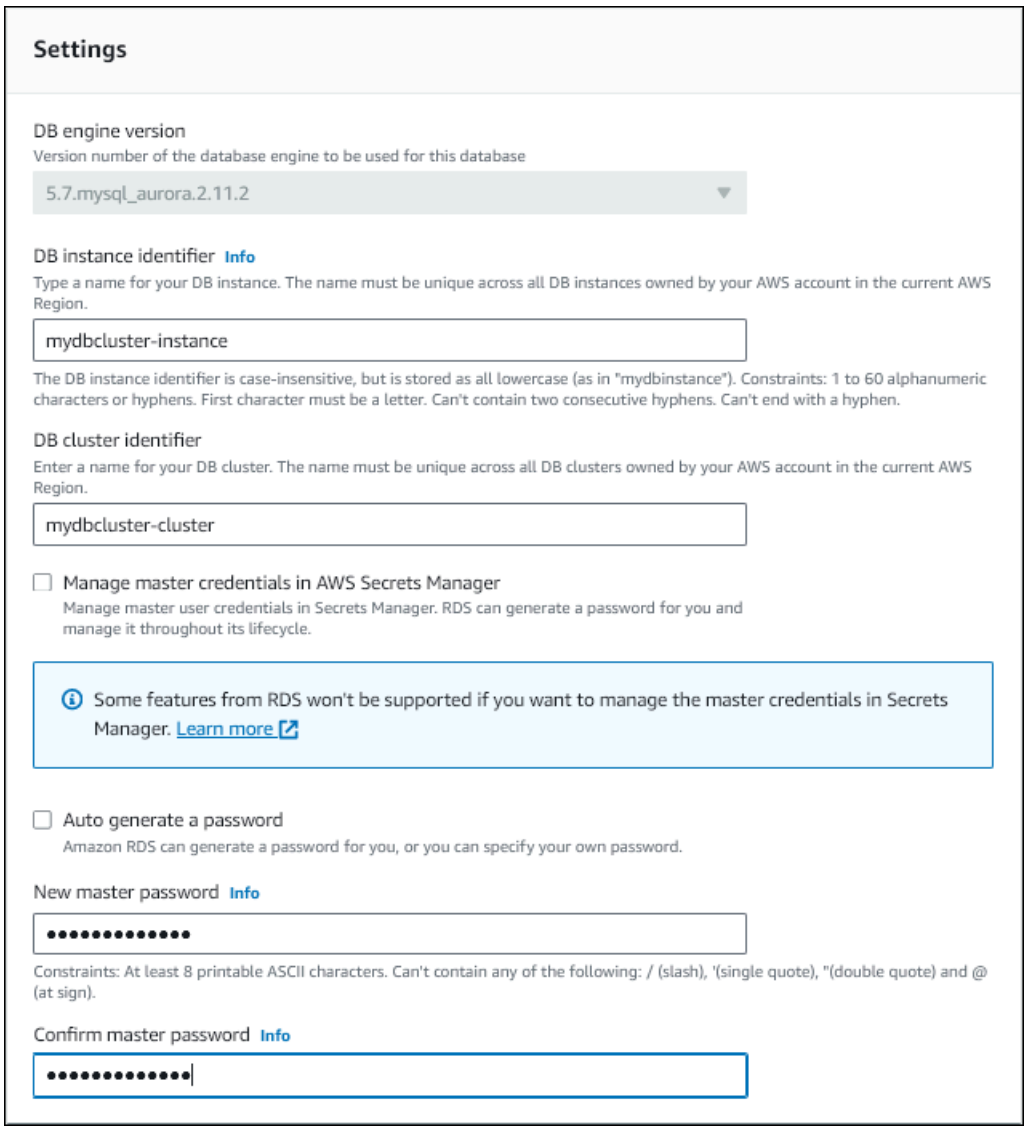
Console

Você vai modificar a instância de banco de dados de gravador para alterar a senha do usuário mestre usando o Console de gerenciamento da AWS.

Como alterar a senha do usuário mestre

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e selecione a instância de banco de dados que você deseja modificar.
3. Para Actions (Ações), selecione Modify (Modificar).

   A página Modify DB instance (Modificar instância de banco de dados) será exibida.
4. Insira uma Nova senha mestra.
5. Em Confirmar a senha principal, insira a mesma nova senha.



6. Escolha Continue (Continuar) e verifique o resumo de modificações.

> **Note**
>
> As alterações de senha sempre são aplicadas imediatamente.

7. Na página de confirmação, escolha Modify DB instance(Modificar a instância de banco de dados).

CLI

Você vai chamar o comando modify-db-cluster para alterar a senha do usuário mestre usando o AWS CLI. Especifique o identificador do cluster de banco de dados e a nova senha, conforme mostrado nos exemplos a seguir.

Você não precisa especificar `--apply-immediately|--no-apply-immediately`, pois as alterações de senha sempre são aplicadas imediatamente.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --master-user-password mynewpassword
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --master-user-password mynewpassword
```

Configurações do Amazon Aurora

A tabela a seguir contém detalhes sobre quais configurações você pode modificar, os métodos usados para modificá-las e o seu escopo. O escopo determina se uma configuração se aplica ao cluster de banco de dados inteiro ou se ela pode ser definida apenas para instâncias de banco de dados específicas.

> **Note**
>
> Configurações adicionais estarão disponíveis se você estiver modificando um cluster de banco de dados do Aurora Serverless v1 ou do Aurora Serverless v2. Para obter mais informações sobre essas configurações, consulte Modificar um cluster de banco de dados do Aurora Serverless v1 e Gerenciar clusters de banco de dados do Aurora Serverless v2. Algumas configurações não estão disponíveis para o Aurora Serverless v1 e o Aurora Serverless v2 devido às suas limitações. Para ter mais informações, consulte Limitações do Aurora Serverless v1 e Requisitos e limitações do Aurora Serverless v2.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Atualização da versão secundária automática Indica se você deseja que a instância de banco de dados receba atualizações secundárias de versão do mecanismo preferida automaticamente quando estiverem disponíveis. Upgrades são instalados apenas durante a janela de manutenção programada. Para obter mais informações sobre atualizações de mecanismos, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL e Atualizações do mecanismo de banco de dados Amazon Aurora MySQL. Para obter mais informações sobre a configuração	**Note** Esta definição é habilitada por padrão. Para cada novo cluster, escolha o valor apropriado para essa configuração com base em sua importância, tempo de vida esperado e a quantidade de testes de verificação que você faz após cada atualização. Ao alterar essa configuração, execute essa modificação para cada instância de banco de dados em seu cluster do Aurora. Se qualquer instância de banco de dados em seu cluster	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração. As interrupções ocorrem durante futuras janelas de manutenção quando o Aurora aplica atualizações automáticas.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Auto minor version upgrade (Atualização automática de versão secundária) para o Aurora MySQL, consulte Habilitar atualizações automáticas entre versões secundárias do Aurora MySQL.	tiver essa configuração desativada, o cluster não será atualizado automaticamente. Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute modify-db-instance e defina a opção `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` . Usando a API do RDS, chame ModifyDBInstance e defina o parâmetro `AutoMinorVersionUpgrade` .		

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Backup retention period (Período de retenção de backup) Por quantos dias os backups automáticos serão mantidos. O valor mínimo é 1. Para obter mais informações, consulte Backups.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute modify-db-cluster e defina a opção `--backup-retention-period`. Usando a API do RDS, chame ModifyDBCluster e defina o parâmetro `BackupRetentionPeriod` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Janela de backup (Hora de início) O intervalo de tempo durante o qual ocorrem backups automatizados dos seus bancos de dados. A janela de backup é uma hora de início no Tempo Coordenado Universal (UTC) e uma duração em horas. Aurora os backups são contínuos e incrementais, mas a janela de backup é usada para criar um backup diário do sistema que é preservado dentro do período de retenção de backup. Você pode copiá-lo para preservá-lo fora do período de retenção. A janela de manutenção e a janela de backup da instância de banco de	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute modify-db-cluster e defina a opção `--preferred-backup-window`. Usando a API do RDS, chame ModifyDBCluster e defina o parâmetro `PreferredBackupWindow` .	O cluster de banco de dados inteiro.	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
dados não podem se sobrepor. Para obter mais informações, consulte Janela de backup.			
Configurações de capacidade As propriedades de ajuste de escala de um cluster de banco de dados do Aurora Serverless v1. Só é possível modificar as propriedades de escalabilidade dos clusters de banco de dados no modo de mecanismo de banco de dados `serverless` . Para obter mais informações sobre o Aurora Serverless v1, consulte Usar o Amazon Aurora Serverless v1.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute modify-db-cluster e defina a opção `--scaling-configuration` . Usando a API do RDS, chame ModifyDBCluster e defina o parâmetro `ScalingConfiguration` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração. A alteração ocorre imediatamente. Essa configuração ignora a configuração para aplicar imediatamente.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Autoridade certificadora A autoridade de certificação (CA) para o certificado do servidor usado pela instância de banco de dados.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute modify-db-instance e defina a opção `--ca-certificate-identifier` . Usando a API do RDS, chame ModifyDBInstance e defina o parâmetro `CACertificateIdentifier` .	Somente a instância de banco de dados especificada	Uma interrupção ocorrerá somente se o mecanismo de banco de dados não for compatível com alternância sem reinicialização. Você pode usar o comando da AWS CLI describe-db-engine-versions para determinar se o mecanismo de banco de dados é compatível com alternância sem reinicialização.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Configuração de armazenamento do cluster O tipo de armazenamento do cluster de banco de dados: Aurora I/O-Optimized ou Aurora Standard. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute modify-db-cluster e defina a opção `--storage-type`. Usando a API do RDS, chame ModifyDBCluster e defina o parâmetro `StorageType` .	O cluster de banco de dados inteiro	A alteração do tipo de armazenamento de um cluster de banco de dados do Aurora PostgreSQL com classes de instância de Leituras otimizadas causa uma interrupção. Isso não ocorre ao alterar os tipos de armazenamento para clusters com outros tipos de classe de instância. Consulte mais informações sobre os tipos de classe de instância de banco de dados em Tipos de classe de instância de banco de dados.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Copiar tags para snapshots Selecione para especificar se as tags definidas para esse cluster de banco de dados são copiadas para snapshot de banco de dadoss criados a partir desse cluster de banco de dados. Para obter mais informações, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-cluster.html e defina a opção `--copy-tags-to-snapshot` ou `--no-copy-tags-to-snapshot` . Usando a API do RDS, chame https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_ModifyDBCluster.html e defina o parâmetro `CopyTagsToSnapshot` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Data API (API de dados) Acesse o Aurora Serverless v1 com aplicações baseadas em serviços da web, como o AWS Lambda e o AWS AppSync. Essa configuração apenas se aplica a um cluster de banco de dados do Aurora Serverless v1. Para obter mais informações, consulte Usar a API de dados do Amazon RDS.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--enable-http-endpoint` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `EnableHttpEndpoint` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Autenticação de banco de dados A autenticação de banco de dados que você deseja usar. Para MySQL: • Selecione Password authentication (Autenticação de senha) para autenticar usuários de banco de dados somente com senhas de banco de dados. • Selecione Password and IAM database authentication (Senha e autenticação de banco de dados do IAM) para autenticar usuários de banco de dados com senhas de banco de dados e credenciais de usuário por meio de funções e usuários do IAM. Para obter mais informações,	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Com a AWS CLI, execute modify-db-cluster e defina as seguintes opções: • Para autenticação do IAM, defina a opção `--enable-iam-database-authentication|--no-enable-iam-database-authentication` . • Para autenticação do Kerberos, defina as opções `--domain` e `--domain-iam-role-name` . Usando a API do RDS, chame o ModifyDBCluster e	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
consulte Autenticação do banco de dados do IAM. Para PostgreSQL: • Selecione IAM database authentication (Autenticação de banco de dados do IAM) para autenticar usuários de banco de dados com senhas de banco de dados e credenciais de usuário por meio de usuários e perfis. Para obter mais informações, consulte Autenticação do banco de dados do IAM. • Escolha Kerberos authentication (Autenticação do Kerberos) para autenticar senhas de banco de dados e credenciais de usuário usando a autenticação do Kerberos. Para obter mais	defina os seguintes parâmetros: • Para autenticação do IAM, defina o parâmetro `EnableIAMDatabaseAuthentication` . • Para autenticação do Kerberos, defina os parâmetros `Domain` e `DomainIAMRoleName` .		

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
informações, consulte Usar a autenticação Kerberos com o Aurora PostgreSQL.			
Porta de banco de dados A porta que você deseja usar para acessar o cluster de banco de dados.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--port`. Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `Port`.	O cluster de banco de dados inteiro	Ocorre uma interrupção durante esta alteração. Todas as instâncias de banco de dados em um cluster de banco de dados são reinicializadas imediatamente.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Identificador do cluster de banco de dados O identificador de cluster de banco de dados. Esse valor é armazenado como uma string em minúsculas. Quando você altera o identific ador de cluster de banco de dados, os endpoints de cluster de banco de dados são alterados . Os endpoints das instâncias de banco de dados no cluster de banco de dados também mudam.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--new-db-cluster-identifier` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `NewDBClusterIdentifier` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Grupo de parâmetros do cluster de banco de dados O parameter group do cluster de banco de dados que você deseja associar ao cluster de banco de dados. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--db-cluster-parameter-group-name` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `DBClusterParameterGroupName` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração. Quando você altera o grupo de parâmetros, as alterações feitas em alguns parâmetros são aplicadas às instâncias de banco de dados no cluster de banco de dados imediatamente sem uma reinicialização. As alterações feitas em outros parâmetros só serão aplicadas depois que as instâncias de banco de dados no cluster de banco de dados forem reinicializadas.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Classe de instância de banco de dados A classe da instância de banco de dados que você quer usar. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--db-instance-class` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `DBInstanceClass` .	Somente a instância de banco de dados especificada	Ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
DB instance identifier O DB instance identifier. Esse valor é armazenado como uma string em minúsculas.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--new-db-instance-identifier`. Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `NewDBInstanceIdentifier`.	Somente a instância de banco de dados especificada	Ocorre uma inatividade durante esta alteração. O RDS reinicia a instância de banco de dados para atualizar o seguinte: • Aurora MySQL: coluna `SERVER_ID` na tabela `information_schema.replica_host_status` • Aurora PostgreSQL: coluna `server_id` na função `aurora_replica_status()`

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Grupo de parâmetros de banco de dados O grupo de parâmetros de banco de dados que você deseja associar à instância de banco de dados. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--db-parameter-group-name` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `DBParameterGroupName` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração. Ao associar um novo grupo de parâmetros de banco de dados a uma instância de banco de dados, os parâmetros estáticos e dinâmicos modificados serão aplicados somente após a reinicialização da instância de banco de dados. No entanto, se você modificar parâmetros dinâmicos no grupo de parâmetros de banco de dados depois de associá-lo à instância de banco de dados, essas alterações serão aplicadas imediatamente sem uma reinicialização. Para ter mais informações, consulte Grupos de parâmetros para Amazon Aurora e Reinicializar

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
			um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora.
Deletion protection (Proteção contra exclusão) Selecione Enable deletion protection (Habilitar proteção contra exclusão) para impedir que seu cluster de banco de dados seja excluído. Para obter mais informações, consulte Proteção contra exclusão para clusters do Aurora..	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--deletion-protection|--no-deletion-protection` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `DeletionProtection` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Versão do mecanismo A versão do mecanismo de banco de dados que você deseja usar. Antes de atualizar seu cluster de banco de dados de produção, recomendamos que você teste o processo de atualização em um cluster de banco de dados de teste para verificar sua duração e validar seus aplicativos.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--engine-version` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `EngineVersion` .	O cluster de banco de dados inteiro	Ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Monitoramento avançado Selecione Enable enhanced monitoring (Habilitar o monitoramento avançado) para habilitar a coleta de métricas em tempo real do sistema operacional em que a instância de banco de dados é executada. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina as opções `--monitoring-role-arn` e `--monitoring-interval` . Usando a API do RDS, chame `ModifyDBInstance` e defina os parâmetros `MonitoringRoleArn` e `MonitoringInterval` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Exportações de log Selecione os tipos de log a serem publicados no Amazon CloudWatch Logs. Para obter mais informações, consulte Arquivos de log do banco de dados AuroraMySQL.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--cloudwatch-logs-export-configuration` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `CloudwatchLogsExportConfiguration` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Janela de manutenção O intervalo de tempo durante o qual a manutenção do sistema ocorre. A manutenção do sistema inclui upgrades, se aplicáveis. A janela de manutenção é uma hora de início no Tempo Coordenado Universal (UTC) e uma duração em horas. Se você definir a janela como a hora atual, deverá haver pelo menos 30 minutos entre a hora atual e o final da janela para garantir que as alterações pendentes sejam aplicadas. Você pode definir a janela de manutenção de maneira independente para o cluster de banco de dados e	Para alterar a janela de manutenção do cluster de banco de dados usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Para alterar a janela de manutenção de uma instância de banco de dados usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Para alterar a janela de manutenção do cluster de banco de dados usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--preferred-maintenance-window` .	O cluster de banco de dados inteiro ou uma única instância de banco de dados	Se houver uma ou mais ações pendentes que causam uma interrupção e a janela de manutenção for alterada para incluir a hora atual, essas ações pendentes serão aplicadas imediatamente, e ocorrerá uma interrupção.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
para cada instância de banco de dados no cluster. Quando o escopo de uma modificação for o cluster de banco de dados inteiro, a modificação será realizada durante a janela de manutenção do cluster. Quando o escopo de uma modificação for a instância de banco de dados, a modificação será realizada durante a janela de manutenção dessa instância. A janela de manutenção e a janela de backup do cluster de banco de dados não podem se sobrepor. Para obter mais informações, consulte Janela de manutenção do Amazon RDS.	Para alterar a janela de manutenção de uma instância de banco de dados usando a AWS CLI, execute `modify-db-instance` e defina a opção `--preferred-maintenance-window` . Para alterar a janela de manutenção do cluster de banco de dados usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `PreferredMaintenanceWindow` . Para alterar a janela de manutenção de uma instância de banco de dados usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `Preferred`		

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
	`MaintenanceWindow` .		

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Gerenciar credenciais principais no AWS Secrets Manager Selecione Gerenciar credenciais principais no AWS Secrets Manager para gerenciar a senha do usuário principal em um segredo no Secrets Manager. Opcionalmente, selecione uma chave do KMS a ser usada para proteger o segredo. Escolha entre uma das chaves do KMS da sua conta ou insira a chave de uma conta distinta. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Se o Aurora já estiver gerenciando a senha do usuário principal para o cluster de banco de dados,	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-cluster` e defina as opções `--manage-master-user-password | --no-manage-master-user-password` e `--master-user-secret-kms-key-id` . Para alternar a senha do usuário principal imediatamente, defina a opção `--rotate-master-user-password` . Usando a API do RDS, chame `ModifyDBCluster` e defina os parâmetros `ManageMasterUserPassword` e `MasterUse`	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
você poderá alternar a senha do usuário principal seleciona ndo Rotate secret immediately (Alternar segredo imediatam ente). Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.	`rSecretKm sKeyId` . Para alternar a senha do usuário principal imediatamente, defina o parâmetro `RotateMas terUserPa ssword` como `true`.		

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Tipo de rede Os protocolos de endereçamento IP compatíveis com o cluster de banco de dados. IPv4. para especificar que os recursos podem se comunicar com o cluster de banco de dados somente por meio do protocolo de endereçamento IPv4. Dual-stack mode (Modo de pilha dupla), para especificar que os recursos podem se comunicar com o cluster de banco de dados por IPv4, IPv6 ou ambos. Use o modo de pilha dupla se você tiver algum recurso que precise se comunicar com o cluster de banco de dados pelo protocolo de endereçamento IPv6. Para usar o modo de pilha dupla,	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--network-type` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `NetworkType` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
pelo menos duas sub-redes devem abranger duas zonas de disponibilidade compatíveis com os protocolos de rede IPv4 e IPv6. Além disso, associe um bloco CIDR IPv6 às sub-redes no grupo de sub-redes de banco de dados especificado. Para obter mais informações, consulte Endereçamento IP do Amazon Aurora.			

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Nova senha mestre A senha da conta de usuário mestre. • Para o Aurora MySQL, a senha deve conter 8 a 41 caracteres ASCII imprimíveis. • Para o Aurora PostgreSQL, ela deve conter de 8 a 99 caracteres ASCII imprimíveis. • Não pode conter `/`, `"`, `@` ou um espaço.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--master-user-password`. Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `MasterUserPassword`.	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Performance Insights Determina se vai habilitar o Performance Insights, uma ferramenta que monitora a carga da sua instância de banco de dados para que você possa analisar e solucionar problemas relacionados à performance do banco de dados. Para obter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--enable-performance-insights|--no-enable-performance-insights` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `EnablePerformanceInsights` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Performance Insights AWS KMS key O identificador da AWS KMS key para criptografia de dados do Performance Insights. O identificador da chave do KMS é o nome do recurso da Amazon (ARN), o identificador de chave ou o alias da chave da chave do KMS. Para obter mais informações, consulte Ativar e desativar o Insights de Performance para o Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--performance-insights-kms-key-id` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `PerformanceInsightsKMSKeyId` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Performance Insights retention period (Período de retenção do Insights de Performance) O período de tempo, em dias, de retenção dos dados do Performance Insights. A configuração de retenção no nível gratuito é Default (7 days) (Padrão (7 dias)). Para reter seus dados de performance por mais tempo, especifique entre 1 e 24 meses. Para obter mais informações sobre os períodos de retenção, consulte Preços e retenção de dados para o Performance Insights. Para obter mais informações, consulte Ativar e desativar o Insights de Performance para o Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--performance-insights-retention-period` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `PerformanceInsightsRetentionPeriod` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Promotion tier (Nível de promoção) Um valor que especifica a ordem em que uma réplica do Aurora é promovida para a instância primária em um cluster de banco de dados, depois de uma falha da instância primária existente. Para obter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--promotion-tier` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `PromotionTier` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Acesso público Publicly accessible (Acessível publicamente) para fornecer à instância de banco de dados um endereço IP público, o que significa que ela é acessível fora da VPC. Para ser acessível publicamente, a instância de banco de dados também deve estar em uma sub-rede pública na VPC. Not publicly accessible (Não acessível publicamente) para tornar a instância de banco de dados acessível somente de dentro da VPC. Para obter mais informações, consulte Ocultar um cluster de banco de dados em uma VPC da Internet. Para se conectar a uma instância de banco de dados de	Usando o Console de gerenciamento da AWS, Modificar uma instância de banco de dados em um cluster de banco de dados. Usando a AWS CLI, execute `modify-db-instance` e defina a opção `--publicly-accessible|--no-publicly-accessible` . Usando a API do RDS, chame `ModifyDBInstance` e defina o parâmetro `PubliclyAccessible` .	Somente a instância de banco de dados especificada	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
fora de sua Amazon VPC, a instância de banco de dados deve ser acessível ao público, o acesso deve ser concedido usando as regras de entrada do grupo de segurança da instância de banco de dados e outros requisitos devem ser atendidos. Para obter mais informaçõ es, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS. Se sua instância de banco de dados não estiver acessível publicamente, também será possível usar uma conexão AWS Site-to-Site VPN ou uma conexão do Direct Connect para acessá-la de uma rede privada. Para obter mais informaçõ es, consulte Privacida			

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
de do tráfego entre redes.			
Configurações de capacidade do Serverless v2 A capacidade do banco de dados de um cluster de banco de dados do Aurora Serverless v2 medida em unidades de capacidade do Aurora (ACUs). Para obter mais informações, consulte Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--serverless-v2-scaling-configuration` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `ServerlessV2ScalingConfiguration` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração. A alteração ocorre imediatamente. Essa configuração ignora a configuração para aplicar imediatamente.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Grupo de segurança O grupo de segurança a ser associado com o cluster de banco de dados. Para obter mais informações, consulte Controlar acesso com grupos de segurança.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--vpc-security-group-ids`. Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `VpcSecurityGroupIds` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configuração e descrição	Método	Escopo	Observações sobre tempo de inatividade
Target Backtrack window (Janela de retrocesso de destino) O tempo que você deseja retroceder em seu cluster de banco de dados, em segundos. Essa configuração só está disponível para o Aurora MySQL e se o cluster de banco de dados foi criado com o recurso Retrocesso habilitado.	Usando o Console de gerenciamento da AWS, Modificar o cluster de banco de dados usando o console, a CLI e a API. Usando a AWS CLI, execute `modify-db-cluster` e defina a opção `--backtrack-window` . Usando a API do RDS, chame `ModifyDBCluster` e defina o parâmetro `BacktrackWindow` .	O cluster de banco de dados inteiro	Não ocorre uma interrupção durante esta alteração.

Configurações não aplicáveis a clusters de banco de dados Amazon Aurora

As configurações a seguir no comando `modify-db-cluster` da AWS CLI e na operação da API do RDS `ModifyDBCluster` não são aplicáveis aos clusters de banco de dados Amazon Aurora.

> **Note**
>
> Não é possível usar o Console de gerenciamento da AWS para modificar essas configurações de clusters de banco de dados Aurora.

Configuração da AWS CLI	Configuração da API do RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`

Configurações não aplicáveis a instâncias de banco de dados do Amazon Aurora

As configurações a seguir no comando `modify-db-instance` da AWS CLI e na operação da API do RDS `ModifyDBInstance` não se aplicam a instâncias de banco de dados Amazon Aurora.

> **Note**
>
> Não é possível usar o Console de gerenciamento da AWS para modificar essas configurações de instâncias de banco de dados Aurora.

Configuração da AWS CLI	Configuração da API do RDS
`--allocated-storage`	`AllocatedStorage`
`--allow-major-version-upgrade|--no-allow-major-version-upgrade`	`AllowMajorVersionUpgrade`
`--copy-tags-to-snapshot|--no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--domain`	`Domain`
`--db-security-groups`	`DBSecurityGroups`
`--db-subnet-group-name`	`DBSubnetGroupName`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--multi-az|--no-multi-az`	`MultiAZ`
`--iops`	`Iops`
`--license-model`	`LicenseModel`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--processor-features`	`ProcessorFeatures`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--use-default-processor-features|--no-use-default-processor-features`	`UseDefaultProcessorFeatures`

Adicionar réplicas do Aurora a um cluster de banco de dados

Um cluster de banco de dados do Aurora com replicação tem uma instância de banco de dados primária e até 15 réplicas do Aurora. A instância do banco de dados primário oferece suporte a operações de leitura e gravação, além de realizar todas as modificações de dados no volume do cluster. As réplicas do Aurora se conectam ao mesmo volume de armazenamento da instância do banco de dados primário e só oferecem suporte a operações de leitura. Você usa réplicas do Aurora para descarregar workloads de leitura da instância do banco de dados primário. Para obter mais informações, consulte Réplicas do Aurora.

Amazon Aurora As réplicas têm as seguintes limitações:

- Não é possível criar uma réplica do Aurora para um cluster de banco de dados do Aurora Serverless v1. O Aurora Serverless v1 tem uma única instância de banco de dados que aumenta e diminui automaticamente para oferecer suporte a todas as operações de leitura e gravação de banco de dados.

  No entanto, é possível adicionar instâncias do leitor aos clusters de banco de dados do Aurora Serverless v2. Para obter mais informações, consulte Adicionar um leitor do Aurora Serverless v2.

Recomendamos distribuir a instância primária e as réplicas do Aurora no cluster de banco de dados do Aurora em várias zonas de disponibilidade para melhorar a disponibilidade do cluster de banco de dados. Para obter mais informações, consulte Disponibilidade de regiões.

Para remover uma réplica do Aurora de um cluster de banco de dados do Aurora, exclua a réplica do Aurora de acordo com as instruções em Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.

> **Note**
>
> O Amazon Aurora também oferece suporte a replicação com um banco de dados externo como uma instância do RDS. A instância de banco de dados do RDS deve estar na mesma região da AWS que o Amazon Aurora. Para obter mais informações, consulte Replicação com o Amazon Aurora.

Você pode adicionar réplicas do Aurora a um cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como adicionar uma réplica do Aurora a um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e selecione o cluster de banco de dados no qual você deseja adicionar a nova instância de banco de dados.
3. Verifique se o cluster e a instância primária estão no estado Disponível. Se o cluster de banco de dados ou a instância primária estiverem em um estado de transição, como Criando, não será possível adicionar uma réplica.

   Se o cluster não tiver uma instância primária, crie uma usando o comando da AWS CLI create-db-instance. Essa situação pode surgir se você usou a CLI para restaurar um snapshot de cluster de banco de dados e visualizar o cluster no Console de gerenciamento da AWS.
4. Para Actions (Ações), escolha Add reader (Adicionar leitor).

   A página Add reader (Adicionar leitor) será exibida.
5. Na página Add reader (Adicionar leitor), especifique as opções para a réplica do Aurora. A tabela a seguir mostra configurações para uma réplica do Aurora.

Para esta opção	Faça o seguinte
Availability zone	Determine se você deseja especificar uma Zona de disponibilidade específica. A lista inclui apenas as zonas de disponibilidade mapeadas para o grupo de sub-redes de banco de dados que você escolheu quando criou o cluster de banco de dados. Para obter mais informações sobre zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.
Publicly accessible	Selecione `Yes` para dar um endereço IP público à réplica do Aurora, do contrário, selecione `No`. Para obter mais informações sobre como ocultar réplicas do Aurora do acesso público, consulte Ocultar um cluster de banco de dados em uma VPC da Internet.

Para esta opção	Faça o seguinte
Criptografia	Selecione `Enable encryption` para habilitar a criptografia em repouso para essa réplica do Aurora. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
Classe da instância de banco de dados	Selecione uma classe de instância de banco de dados que defina os requisitos de processamento e memória para cada réplica do Aurora. Para obter mais informações sobre as opções de classe de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.
Aurora replica source (Origem da réplica do Aurora	Selecione o identificador da instância primária para criar uma réplica do Aurora.
Identificador de instância de banco de dados	Insira um nome para a instância que seja exclusivo para a sua conta na região da AWS selecionada. É possível optar por adicionar informações ao nome, como incluir a região da AWS e o mecanismo de banco de dados selecionados, por exemplo, **`aurora-read-instance1`** .
Prioridade	Escolha uma prioridade de failover para a instância. Se você não selecionar um valor, o padrão será tier-1. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Para obter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.
Database port	A porta de uma réplica do Aurora é a mesma a porta do cluster de banco de dados.

Para esta opção	Faça o seguinte
Grupo de parâmetros de banco de dados	Selecione um grupo de parâmetros. O Aurora conta com um grupo de parâmetros padrão que você pode usar, ou você pode criar seu próprio grupo de parâmetros. Para obter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.
Insights de Performance	A caixa de seleção Turn on Performance Insights (Ativar o Performance Insights) é selecionada por padrão. O valor não é herdado da instância do gravador. Para obter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.
Monitoramento avançado	Escolha Enable enhanced monitoring (Habilitar monitoramento avançado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para ter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Monitoring Role (Monitoramento de perfis	Disponível apenas quando Enhanced Monitoring (Monitoramento avançado) estiver definido como Enable enhanced monitoring (Habilitar monitoramento avançado). Escolha o perfil do IAM que você criou para permitir que o Amazon RDS se comunique com o Amazon CloudWatch Logs para você. Ou escolha Default (Padrão) para que o RDS crie um perfil para você chamado `rds-monitoring-role` . Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.

Para esta opção	Faça o seguinte
Granularity	Disponível apenas quando Enhanced Monitoring estiver definido como Enable enhanced monitoring. Defina o intervalo, em segundos, em que as métricas são coletadas para o seu cluster de banco de dados.
Atualização da versão secundária automática	Selecione Enable auto minor version upgrade (Habilitar atualização automática de versão secundária) se você quiser permitir que o cluster de banco de dados do Aurora receba automaticamente as atualizações de versão secundária do mecanismo de banco de dados quando disponíveis. A configuração Atualização automática de versão secundária se aplica aos clusters de banco de dados do Aurora PostgreSQL e do Aurora MySQL. Para clusters do Aurora MySQL 2.x, essa configuração faz upgrade dos clusters para uma versão máxima de 2.07.2. Para obter mais informações sobre atualizações de mecanismos para o Aurora PostgreSQL, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL. Para obter mais informações sobre atualizações de mecanismos para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

6. Escolha Add reader (Adicionar leitor) para criar a réplica do Aurora.

AWS CLI

Para criar uma réplica do Aurora em seu cluster de banco de dados, execute o comando da create-db-instance da AWS CLI. Inclua o nome de um cluster de banco de dados como a opção `--db-cluster-identifier`. Como alternativa, você pode especificar uma zona de disponibilidade para

a réplica do Aurora usando o parâmetro `--availability-zone`, conforme mostrado nos exemplos a seguir.

Por exemplo, o comando a seguir cria uma nova réplica do Aurora compatível com o MySQL 5.7 chamada `sample-instance-us-west-2a`.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

O comando a seguir cria uma réplica do Aurora compatível com o MySQL 5.7 chamada `sample-instance-us-west-2a`.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

O comando a seguir cria uma réplica do Aurora compatível com o PostgreSQL chamada `sample-instance-us-west-2a`.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large ^
    --availability-zone us-west-2a
```

API do RDS

Para criar uma réplica do Aurora em seu cluster de banco de dados, chame a operação CreateDBInstance. Inclua o nome de um cluster de banco de dados como o parâmetro `DBClusterIdentifier`. Como alternativa, você pode especificar uma zona de disponibilidade para a réplica do Aurora usando o parâmetro `AvailabilityZone`.

Para ter informações sobre ajuste de escala automático do Amazon Aurora com réplicas, consulte as seções a seguir.

Tópicos

- Amazon Aurora Auto Scaling com réplicas do Aurora
- Adicionar uma política de ajuste de escala automático a um cluster de banco de dados do Amazon Aurora
- Editar uma política de ajuste de escala automático para um cluster de banco de dados do Amazon Aurora
- Excluir uma política de ajuste de escala automático do cluster de banco de dados do Amazon Aurora

Amazon Aurora Auto Scaling com réplicas do Aurora

Para atender aos requisitos de conectividade e de workload, a autoescalabilidade do Aurora ajusta dinamicamente o número de réplicas do Aurora (instâncias de banco de dados do leitor) provisionadas para um cluster de banco de dados Aurora. O Auto Scaling do Aurora está disponível para ambos Aurora MySQL e Aurora PostgreSQL. O Auto Scaling do Aurora permite que o cluster de banco de dados Aurora processe aumentos repentinos de conectividade ou de workload. Quando a conectividade ou a workload diminui, o Auto Scaling do Aurora elimina as réplicas desnecessárias do Aurora para que você não pague por instâncias de banco de dados provisionadas não usadas.

Você define e aplica as políticas de escalabilidade do Aurora para um cluster de banco de dados. A política de escalabilidade define o número mínimo e máximo de réplicas do Aurora que o Auto Scaling do Aurora pode gerenciar. Com base nas políticas, o Auto Scaling do Aurora ajusta o número de réplicas do Aurora para mais ou para menos em resposta às workloads reais, usando as métricas e valores de destino do Amazon CloudWatch.

> **Note**
>
> O Aurora Auto Scaling não se aplica à workload na instância de banco de dados do gravador. O Aurora Auto Scaling é útil para a workload somente nas instâncias do leitor.

Você pode usar o Console de gerenciamento da AWS para uma política de escalabilidade com base em uma métrica predefinida. Se preferir, você pode usar a AWS CLI ou a API do Auto Scaling do Aurora para aplicar uma política de escalabilidade com base em uma métrica predefinida ou personalizada.

Tópicos

- Antes de começar
- Políticas de Auto Scaling do Aurora
- IDs de instância de banco de dados e marcação
- Aurora Auto Scaling e Performance Insights

Antes de começar

Para usar o Aurora Auto Scaling com um cluster de banco de dados do Aurora, você deve primeiro criar um cluster de banco de dados do Aurora com uma instância de banco de dados (gravação)

primária. Para ter mais informações sobre como criar um cluster de banco de dados Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.

O Aurora Auto Scaling só dimensionará um cluster de banco de dados se o cluster de banco de dados estiver no estado disponível.

Quando o Auto Scaling do Aurora adicionar uma nova réplica do Aurora, a nova réplica do Aurora será da mesma classe de instância de banco de dados que a usada pela instância principal. Para mais informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora. Além disso, o nível de promoção para novas réplicas do Aurora está definido como a última prioridade, que é 15 por padrão. Isso significa que, durante um failover, uma réplica com uma prioridade maior, como uma criada manualmente, será promovida primeiro. Para ter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

O Auto Scaling do Aurora remove somente as réplicas do Aurora que ele criou.

Para que se beneficiem do Auto Scaling do Aurora, seus aplicativos devem oferecer suporte a conexões com as novas réplicas do Aurora. Para fazer isso, recomendamos usar o endpoint do leitor do Aurora. É possível usar um driver, como o driver JDBC da AWS. Para obter mais informações, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

> **Note**
>
> No momento, os bancos de dados globais Aurora não oferecem suporte ao Aurora Auto Scaling para clusters de banco de dados secundários.

Políticas de Auto Scaling do Aurora

O Auto Scaling do Aurora usa uma política de escalabilidade para ajustar o número de réplicas do Aurora em um cluster de banco de dados Aurora. O Aurora Auto Scaling tem os seguintes componentes:

- Uma função vinculada a serviços
- Uma métrica de destino
- Capacidades mínima e máxima
- Um desaquecimento

Tópicos

- Função vinculada ao serviço
- Métrica de destino
- Capacidades mínima e máxima
- Desaquecimento
- Habilitar ou desabilitar atividades de redução
- Adicionar, editar ou excluir políticas de ajuste de escala automático

Função vinculada ao serviço

O Auto Scaling do Aurora usa a função vinculada ao serviço `AWSServiceRoleForApplicationAutoScaling_RDSCluster`. Para ter mais informações, consulte Funções vinculadas a serviço do Application Auto Scaling, no Guia do usuário do Application Auto Scaling.

Métrica de destino

Neste tipo de política, uma métrica predefinida ou personalizada e um valor de destino dessa métrica são especificados na configuração de uma política de escalabilidade de rastreamento de destino. O Auto Scaling do Aurora cria e gerencia os alarmes do CloudWatch que acionam a política de escalabilidade e calculam o ajuste de escalabilidade com base na métrica e no valor de destino. A política de escalabilidade adiciona ou remove réplicas do Aurora conforme necessário para manter a métrica no valor de destino especificado ou próxima a ele. Além de manter a métrica próxima ao valor de destino, uma política de escalabilidade de rastreamento de destino também se ajusta às flutuações na métrica, devido a workloads variáveis. Essas políticas também minimizam flutuações rápidas no número de réplicas do Aurora disponíveis para seu cluster de banco de dados.

Por exemplo, considere uma política de escalabilidade que use a métrica predefinida de utilização média da CPU. Essa política pode manter a utilização da CPU a uma porcentagem específica de utilização específica, por exemplo, 40 por cento ou próxima disso.

> **Note**
>
> Para cada cluster de banco de dados Aurora, você pode criar somente uma política de Auto Scaling para cada métrica de destino.

Quando você configura o Aurora Auto Scaling, o valor da métrica de destino é calculado como a média de todas as instâncias de leitor no cluster. Esse cálculo é realizado da seguinte maneira:

- Inclui todas as instâncias de leitor no cluster do Aurora, independentemente de serem gerenciadas pelo Auto Scaling ou adicionadas manualmente.
- Inclui instâncias de banco de dados associadas a endpoints personalizados. Os endpoints personalizados não influenciam o cálculo das métricas de destino.
- Não inclui a instância de gravador do cluster.

As métricas são derivadas do CloudWatch usando as seguintes dimensões:

- `DBClusterIdentifier`
- `Role=READER`

Por exemplo, considere um cluster do Aurora MySQL com a seguinte configuração:

- Instâncias manuais (não controladas pelo Auto Scaling):
  - Gravador com 50% de utilização da CPU
  - Leitor 1 (endpoint personalizado: `custom-reader-1`) com 90% de utilização da CPU
  - Leitor 2 (endpoint personalizado: `custom-reader-2`) com 90% de utilização da CPU
- Instância do Auto Scaling:
  - Leitor 3 (adicionado usando o Auto Scaling) com 10% de utilização da CPU

Nesse cenário, a métrica de destino para a política do Auto Scaling é calculada da seguinte forma:

```
Target metric = (CPU utilization of reader 1 + reader 2 + reader 3) / total number of
 readers

Target metric = (90 + 90 + 10) / 3 = 63.33%
```

A política do Auto Scaling usa esse valor para avaliar se deve reduzir ou aumentar a escala horizontalmente com base no limite definido.

Considere o seguinte:

- Embora os endpoints personalizados determinem como o tráfego é roteado para leitores específicos, eles não excluem os leitores do cálculo da métrica.
- As instâncias manuais são sempre incluídas nos cálculos da métrica de destino.
- Para evitar um comportamento inesperado de escalabilidade, certifique-se de que a configuração do Auto Scaling considere todas as instâncias de leitor no cluster.
- Se um cluster não tiver leitores, a métrica não será calculada e os alarmes da política do Auto Scaling permanecerão inativos. Para que a política do Auto Scaling funcione de forma eficaz, pelo menos um leitor deve estar presente o tempo todo.

Capacidades mínima e máxima

Você pode especificar o número máximo de réplicas do Aurora a serem gerenciadas pelo Application Auto Scaling. Este valor deve ser definido entre 0 – 15, e deve ser igual ou maior que o valor especificado para o número mínimo de réplicas do Aurora.

Você também pode especificar o número mínimo de réplicas do Aurora a serem gerenciadas pelo Application Auto Scaling. Este valor deve ser definido entre 0 – 15, e deve ser igual ou menor que o valor especificado para o número máximo de réplicas do Aurora.

Deve haver pelo menos um leitor na instância de banco de dados do leitor para que o Aurora Auto Scaling funcione. Se o cluster de banco de dados não tiver uma instância do leitor e você definir a capacidade mínima como 0, o Aurora Auto Scaling não funcionará.

> **Note**
>
> A capacidade mínima e máxima são definidas para um cluster de banco de dados Aurora. Os valores especificados se aplicam a todas as políticas associadas a um cluster de banco de dados Aurora.

Desaquecimento

Você pode ajustar a responsividade das políticas de escalabilidade de rastreamento de destino, adicionando desaquecimentos que afetam a escalabilidade de seu cluster de banco de dados Aurora pra mais ou para menos. Um desaquecimento bloqueia solicitações subsequentes de redução ou expansão até o período expirar. Esses blocos retardam as exclusões de réplicas do Aurora em seu cluster de banco de dados Aurora para solicitações de redução e a criação de réplicas do Aurora para solicitações de expansão.

Você pode especificar os seguintes desaquecimentos:

- A atividade de redução diminui o número de réplicas do Aurora em seu cluster de banco de dados Aurora. Um desaquecimento de redução especifica a quantidade de tempo, em segundos, após a conclusão de uma ação de redução antes que quaisquer outras atividades de redução possam iniciar.
- A atividade de expansão aumenta o número de réplicas do Aurora em seu cluster de banco de dados Aurora. Um desaquecimento de expansão especifica a quantidade de tempo, em segundos, após a conclusão de uma ação de expansão antes que quaisquer outras atividades de expansão possam iniciar.

> **Note**
>
> Um período de esfriamento de aumento da escala horizontalmente será ignorado se uma solicitação subsequente para aumentar a escala horizontalmente for para um número maior de réplicas do Aurora do que o da primeira solicitação.

Se você não especificar um período de esfriamento de aumento ou redução da escala horizontalmente, o padrão para cada um será de 300 segundos.

Habilitar ou desabilitar atividades de redução

Você pode habilitar ou desabilitar atividades de redução para as políticas. Habilitar atividades de redução permite que as políticas de escalabilidade excluam réplicas do Aurora. Quando as atividades de redução são habilitadas, o desaquecimento de redução na política de escalabilidade aplica-se a atividades de redução. Desabilitar atividades de redução evita que as políticas de escalabilidade excluam réplicas do Aurora.

> **Note**
>
> As atividades de expansão são habilitadas para que a política de escalabilidade possa criar réplicas do Aurora, conforme necessário.

Adicionar, editar ou excluir políticas de ajuste de escala automático

É possível adicionar, editar ou excluir políticas de ajuste de escala automático usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Application Auto Scaling. Para ter mais informações

sobre adição, edição ou exclusão de políticas de ajuste de escala automático, consulte as seções a seguir.

- Adicionar uma política de ajuste de escala automático a um cluster de banco de dados do Amazon Aurora
- Editar uma política de ajuste de escala automático para um cluster de banco de dados do Amazon Aurora
- Excluir uma política de ajuste de escala automático do cluster de banco de dados do Amazon Aurora

IDs de instância de banco de dados e marcação

Quando uma réplica é adicionada pelo Aurora Auto Scaling, seu ID de instância de banco de dados é prefixado por `application-autoscaling-`, por exemplo, `application-autoscaling-61aabbcc-4e2f-4c65-b620-ab7421abc123`.

A tag a seguir é adicionada automaticamente à instância de banco de dados. Você pode visualizá-lo na guia Tags da página de detalhes da instância de banco de dados.

Tag	Valor
application-autoscaling:resourceId	cluster:mynewcluster-cluster

Para ter mais informações sobre Amazon RDS as tags de recurso, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.

Aurora Auto Scaling e Performance Insights

Você pode usar o Performance Insights para monitorar réplicas que foram adicionadas pelo Aurora Auto Scaling da mesma forma que com qualquer instância de banco de dados do leitor do Aurora.

Para ter mais informações sobre como usar o Performance Insights para monitorar clusters de banco de dados do Aurora, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

Adicionar uma política de ajuste de escala automático a um cluster de banco de dados do Amazon Aurora

Você pode adicionar uma política de escalabilidade usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Application Auto Scaling.

> **Note**
>
> Para obter um exemplo que adiciona uma política de escalabilidade usando o CloudFormation, consulte Declaring a scaling policy for an Aurora DB cluster (Declarar uma política de escalabilidade para um cluster de banco de dados Aurora) no Guia do usuário do AWS CloudFormation.

Console

Você pode adicionar uma política de escalabilidade a um cluster de banco de dados Aurora usando o Console de gerenciamento da AWS.

Como adicionar uma política do Auto Scaling a um cluster de banco de dados Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados Aurora ao qual você deseja adicionar a política.
4. Escolha a guia Logs & events (Logs e eventos).
5. Na seção Auto scaling policies (Políticas do Auto Scaling), selecione Add (Adicionar).

   A caixa de diálogo Add Auto Scaling policy (Adicionar política de Auto Scaling) será exibida.
6. Em Policy Name (Nome da política), digite o nome da política.
7. Quanto à métrica de destino, escolha uma das seguintes opções:

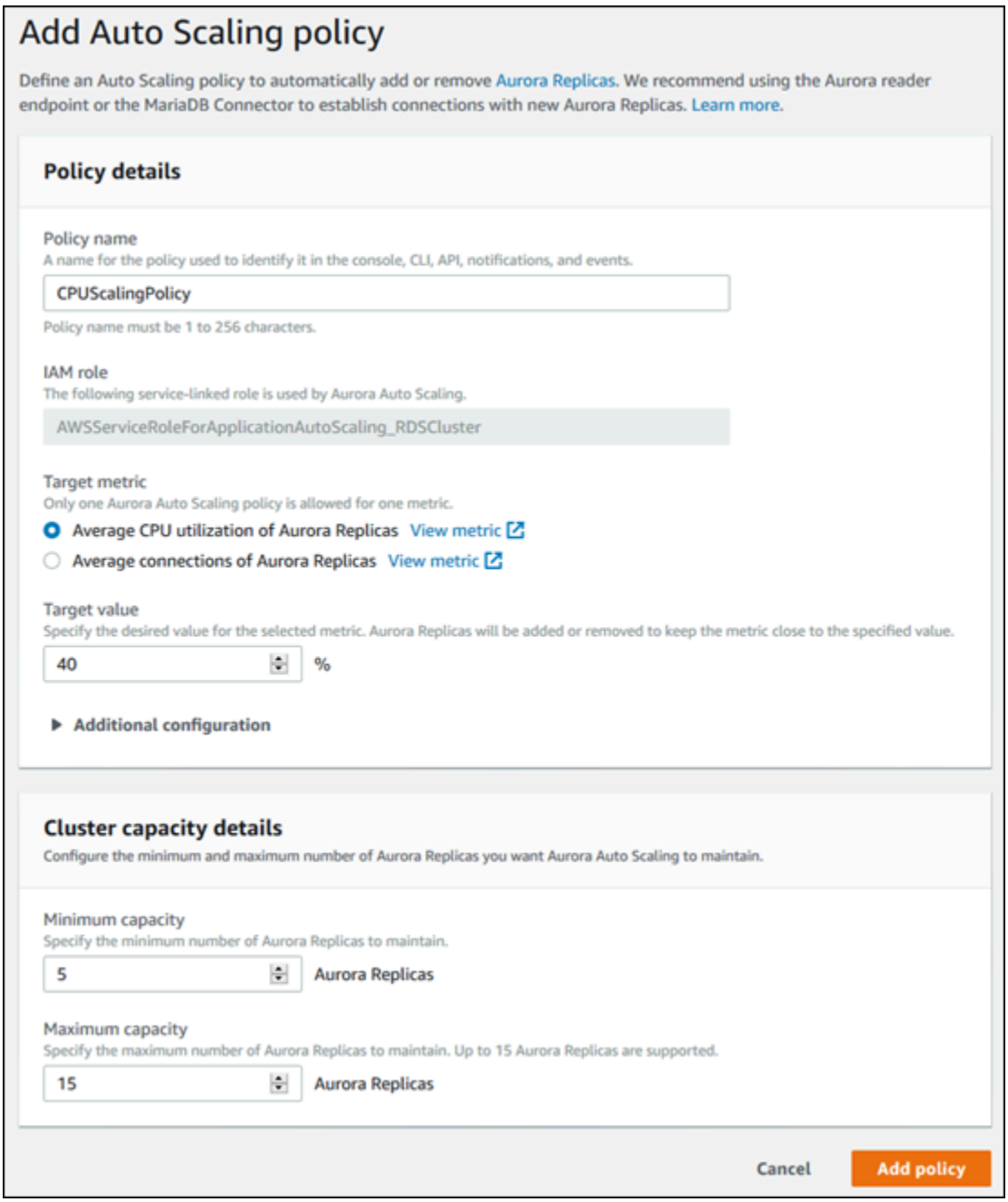
   - Average CPU utilization of Aurora Replicas (Utilização média da CPU de réplicas do Aurora) para criar uma política com base na utilização média da CPU.
   - Average connections of Aurora Replicas (Média de conexões de réplicas do Aurora) para criar uma política com base no número médio de conexões com réplicas do Aurora.
8. Quanto ao valor de destino, digite uma das seguintes opções:

- Se você escolheu Average CPU utilization of Aurora Replicas (Utilização média da CPU de réplicas do Aurora) na etapa anterior, digite a porcentagem de utilização da CPU que você deseja manter nas réplicas do Aurora.
- Se você escolheu Average connections of Aurora Replicas (Média de conexões de réplicas do Aurora) na etapa anterior, digite o número de conexões que você deseja manter.

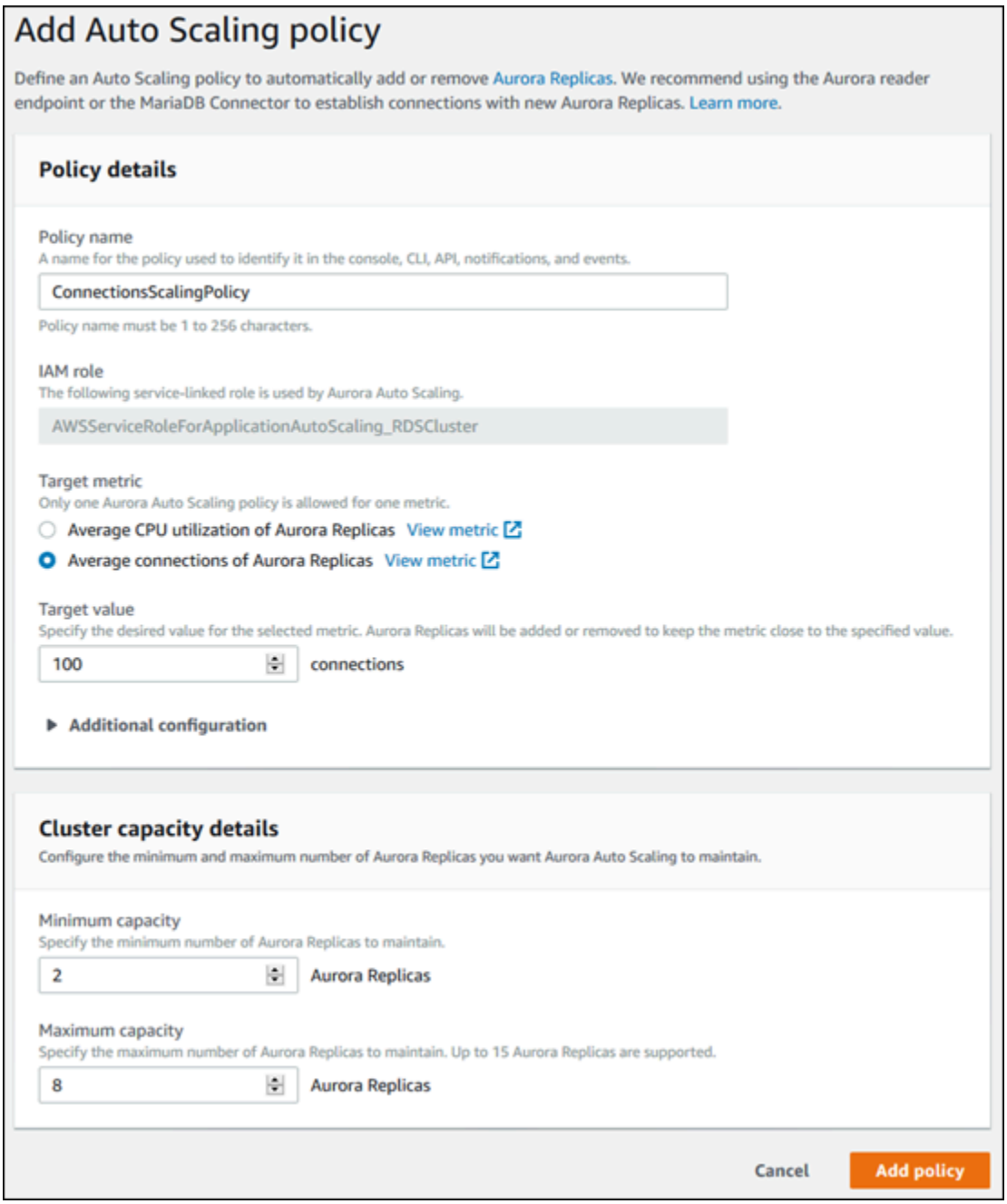
As réplicas do Aurora serão adicionadas ou removidas por manter a métrica próxima ao valor especificado.

9. (Opcional) Expanda Additional Configuration (Configuração adicional) para criar um período de esfriamento de redução ou aumento da escala horizontalmente.
10. Em Minimum capacity (Capacidade mínima), digite o número mínimo de réplicas do Aurora que a política de Auto Scaling do Aurora precisa manter.
11. Em Maximum capacity (Capacidade máxima), digite o número máximo de réplicas do Aurora que a política de Auto Scaling do Aurora precisa manter.
12. Escolha Add policy.

A caixa de diálogo a seguir cria uma política de Auto Scaling com base na utilização média da CPU em 40 por cento. A política especifica um mínimo de 5 réplicas do Aurora e o máximo 15 réplicas do Aurora.



A caixa de diálogo a seguir cria uma política do Auto Scaling com base no número médio de 100 conexões. A política especifica um mínimo de duas réplicas do Aurora e o máximo oito réplicas do Aurora.



AWS CLI ou API do Application Auto Scaling

Você pode aplicar uma política de escalabilidade com base em uma métrica predefinida ou personalizada. Para fazer isso, você pode usar a AWS CLI ou a API do Application Auto Scaling. A primeira etapa é registrar seu cluster de banco de dados Aurora no Application Auto Scaling.

Registro de um cluster de banco de dados Aurora

Antes de usar o Auto Scaling do Aurora com um cluster de banco de dados Aurora, você deve registrar seu cluster de banco de dados Aurora no Application Auto Scaling. Faça isso para definir a dimensão de escalabilidade e os limites a serem aplicados no cluster. O Application Auto Scaling escala dinamicamente o cluster de banco de dados Aurora ao longo da dimensão escalável `rds:cluster:ReadReplicaCount`, que representa o número de réplicas do Aurora.

Para registrar seu cluster de banco de dados Aurora, você pode usar a AWS CLI ou a API do Application Auto Scaling.

AWS CLI

Para registrar seu cluster de banco de dados Aurora, use o comando `register-scalable-target` da AWS CLI com os seguintes parâmetros:

- `--service-namespace` – Defina este valor como `rds`.
- `--resource-id` – o identificador do recurso para o cluster de banco de dados Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `--scalable-dimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.
- `--min-capacity` – o número mínimo de instâncias de banco de dados de leitura a serem gerenciadas pelo Application Auto Scaling. Para obter informações sobre a relação entre `--min-capacity`, `--max-capacity` e o número de instâncias de banco de dados no cluster, consulte Capacidades mínima e máxima.
- `--max-capacity` – o número máximo de instâncias de banco de dados de leitura a serem gerenciadas pelo Application Auto Scaling. Para obter informações sobre a relação entre `--min-capacity`, `--max-capacity` e o número de instâncias de banco de dados no cluster, consulte Capacidades mínima e máxima.

Example

No exemplo a seguir, registre um cluster de banco de dados Aurora chamado `myscalablecluster`. O registro indica que o cluster de banco de dados deve ser escalado dinamicamente para ter de uma a oito réplicas do Aurora.

Para Linux, macOS ou Unix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --min-capacity 1 \
    --max-capacity 8 \
```

Para Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

API do Application Auto Scaling

Para registrar seu cluster de banco de dados Aurora no Application Auto Scaling, use a operação `RegisterScalableTarget` da API do Application Auto Scaling com os seguintes parâmetros:

- `ServiceNamespace` – Defina este valor como `rds`.
- `ResourceID` – o identificador do recurso para o cluster de banco de dados Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `ScalableDimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.
- `MinCapacity` – o número mínimo de instâncias de banco de dados de leitura a serem gerenciadas pelo Application Auto Scaling. Para obter informações sobre a relação entre `MinCapacity`, `MaxCapacity` e o número de instâncias de banco de dados no cluster, consulte Capacidades mínima e máxima.
- `MaxCapacity` – o número máximo de instâncias de banco de dados de leitura a serem gerenciadas pelo Application Auto Scaling. Para obter informações sobre a relação entre `MinCapacity`, `MaxCapacity` e o número de instâncias de banco de dados no cluster, consulte Capacidades mínima e máxima.

Example

No exemplo a seguir, registre um cluster de banco de dados Aurora chamado `myscalablecluster` na API do Application Auto Scaling. Este registro indica que o cluster de banco de dados deve ser escalado dinamicamente para ter de uma a oito réplicas do Aurora.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Definir uma política de escalabilidade para um cluster de banco de dados Aurora

Uma configuração de política de escalabilidade de rastreamento de destino é representada por um bloco JSON no qual as métricas e valores de destino são definidos. Você pode salvar uma configuração de política de escalabilidade como um bloco JSON em um arquivo de texto. Você pode usar esse arquivo de texto ao invocar a AWS CLI ou a API do Application Auto Scaling. Para ter mais informações sobre a sintaxe de configuração de política, consulte `TargetTrackingScalingPolicyConfiguration` na Referência de API do Application Auto Scaling.

As seguintes opções estão disponíveis para definir uma configuração de política de escalabilidade de rastreamento de destino.

Tópicos

- Uso de uma métrica predefinida
- Uso de uma métrica personalizada

- Uso de períodos de desaquecimento
- Desabilitar a atividade de redução

Uso de uma métrica predefinida

Com o uso de métricas predefinidas, você pode definir rapidamente uma política de escalabilidade de rastreamento de destino para um cluster de banco de dados Aurora que funcione bem com o rastreamento de destino e com a escalabilidade dinâmica no Auto Scaling do Aurora.

Atualmente, o Aurora oferece suporte às seguintes métricas predefinidas no Auto Scaling do Aurora:

- RDSReaderAverageCPUUtilization – o valor médio da métrica `CPUUtilization` no CloudWatch em todas as réplicas do Aurora no cluster de banco de dados Aurora.
- RDSReaderAverageDatabaseConnections – o valor médio da métrica `DatabaseConnections` no CloudWatch em todas as réplicas do Aurora no cluster de banco de dados Aurora.

Para ter mais informações sobre as métricas `CPUUtilization` e `DatabaseConnections`, consulte Métricas do Amazon CloudWatch para o Amazon Aurora.

Para usar uma métrica predefinida em sua política de escalabilidade, crie uma configuração de rastreamento de destino para sua política de escalabilidade. Essa configuração deve incluir uma `PredefinedMetricSpecification` para a métrica predefinida e um `TargetValue` para o valor de destino dessa métrica.

Example

O exemplo a seguir descreve uma configuração de política típica para a escalabilidade de rastreamento de destino para um cluster de banco de dados Aurora. Nessa configuração, a métrica predefinida `RDSReaderAverageCPUUtilization` é usada para ajustar o cluster de banco de dados Aurora com base em uma utilização média da CPU de 40 por cento em todas as réplicas do Aurora.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    }
}
```

Uso de uma métrica personalizada

Com o uso de métricas personalizadas, você pode definir uma política de escalabilidade de rastreamento de destino que atenda a suas exigências personalizadas. Você pode definir uma métrica personalizada com base em qualquer métrica do Aurora que mude na proporção da escalabilidade.

Nem todas as métricas do Aurora funcionam para o rastreamento de destino. A métrica deve ser de utilização válida e descrever o quão ocupada uma instância está. O valor da métrica deve aumentar ou diminuir na proporção do número de réplicas do Aurora no cluster de banco de dados Aurora. Essa aumento ou redução proporcional é necessário para usar os dados da métrica para expandir ou reduzir proporcionalmente o número de réplicas do Aurora.

Example

O exemplo a seguir descreve uma configuração de rastreamento de destino para uma política de escalabilidade. Nessa configuração, uma métrica personalizada ajusta um cluster de banco de dados Aurora com base na utilização média de uma CPU em 50 por centro em todas as réplicas do Aurora, em um cluster de banco de dados Aurora chamado `my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Uso de períodos de desaquecimento

Você pode especificar um valor, em segundos, para `ScaleOutCooldown` a fim de adicionar um desaquecimento para expandir seu cluster de banco de dados Aurora. De forma similar, você pode adicionar um valor, em segundos, para `ScaleInCooldown` a fim de adicionar um desaquecimento para reduzir seu cluster de banco de dados Aurora.

Para ter mais informações sobre `ScaleInCooldown` e `ScaleOutCooldown`, consulte `TargetTrackingScalingPolicyConfiguration` na Referência de API do Application Auto Scaling.

Example

O exemplo a seguir descreve uma configuração de rastreamento de destino para uma política de escalabilidade. Nessa configuração, a métrica predefinida `RDSReaderAverageCPUUtilization` é usada para ajustar um cluster de banco de dados Aurora com base em uma utilização média da CPU de 40 por cento em todas as réplicas do Aurora nesse cluster de banco de dados Aurora. A configuração fornece um desaquecimento de redução de 10 minutos e em um desaquecimento de expansão de 5 minutos.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Desabilitar a atividade de redução

Você pode evitar que a configuração da política de escalabilidade de rastreamento de destino reduza seu cluster de banco de dados Aurora desabilitando a atividade de redução. Desabilitar a atividade de redução impede que a política de escalabilidade exclua réplicas do Aurora, enquanto permite ao mesmo tempo que a política de escalabilidade crie réplicas conforme o necessário.

Você pode especificar um valor booliano para `DisableScaleIn` a fim de permitir ou evitar a atividade de redução em seu cluster de banco de dados Aurora. Para ter mais informações sobre `DisableScaleIn`, consulte `TargetTrackingScalingPolicyConfiguration` na Referência de API do Application Auto Scaling.

Example

O exemplo a seguir descreve uma configuração de rastreamento de destino para uma política de escalabilidade. Nessa configuração, a métrica predefinida `RDSReaderAverageCPUUtilization` ajusta um cluster de banco de dados Aurora com base em uma utilização média da CPU de 40

por cento em todas as réplicas do Aurora nesse cluster de banco de dados Aurora. A configuração desativa a atividade de redução para a política de escalabilidade.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Aplicar uma política de escalabilidade a um cluster de banco de dados Aurora

Após registrar seu cluster de banco de dados Aurora no Application Auto Scaling e definir uma política de escalabilidade, aplique a política de escalabilidade ao cluster de banco de dados Aurora. Para aplicar uma política de escalabilidade a um cluster de banco de dados Aurora, você pode usar a AWS CLI ou a API do Application Auto Scaling.

AWS CLI

Para aplicar uma política de escalabilidade a seu cluster de banco de dados Aurora, use o comando `put-scaling-policy` da AWS CLI com os seguintes parâmetros:

- `--policy-name` – o nome da política de escalabilidade.
- `--policy-type` – Defina este valor como `TargetTrackingScaling`.
- `--resource-id` – o identificador do recurso para o cluster de banco de dados Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `--service-namespace` – Defina este valor como `rds`.
- `--scalable-dimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration` – a configuração da política de escalabilidade de rastreamento de destino a ser usada para o cluster de banco de dados Aurora.

Example

No exemplo a seguir, aplique a política de escalabilidade de rastreamento de destino chamada `myscalablepolicy` a um cluster de banco de dados Aurora chamado `myscalablecluster` com

o Application Auto Scaling. Para fazer isso, use uma configuração de política salva em um arquivo chamado `config.json`.

Para Linux, macOS ou Unix:

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --target-tracking-scaling-policy-configuration file://config.json
```

Para Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

API do Application Auto Scaling

Para aplicar a política de escalabilidade em seu cluster de banco de dados Aurora com a API do Application Auto Scaling, use a operação `PutScalingPolicy` da API do Application Auto Scaling com os seguintes parâmetros:

- `PolicyName` – o nome da política de escalabilidade.
- `ServiceNamespace` – Defina este valor como `rds`.
- `ResourceID` – o identificador do recurso para o cluster de banco de dados Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `ScalableDimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.
- `PolicyType` – Defina este valor como `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration` – a configuração da política de escalabilidade de rastreamento de destino a ser usada para o cluster de banco de dados Aurora.

Example

No exemplo a seguir, aplique a política de escalabilidade de rastreamento de destino chamada `myscalablepolicy` a um cluster de banco de dados do Aurora chamado `myscalablecluster` com o Application Auto Scaling. Use uma configuração de política com base na métrica predefinida `RDSReaderAverageCPUUtilization`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Editar uma política de ajuste de escala automático para um cluster de banco de dados do Amazon Aurora

Você pode editar uma política de escalabilidade usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Application Auto Scaling.

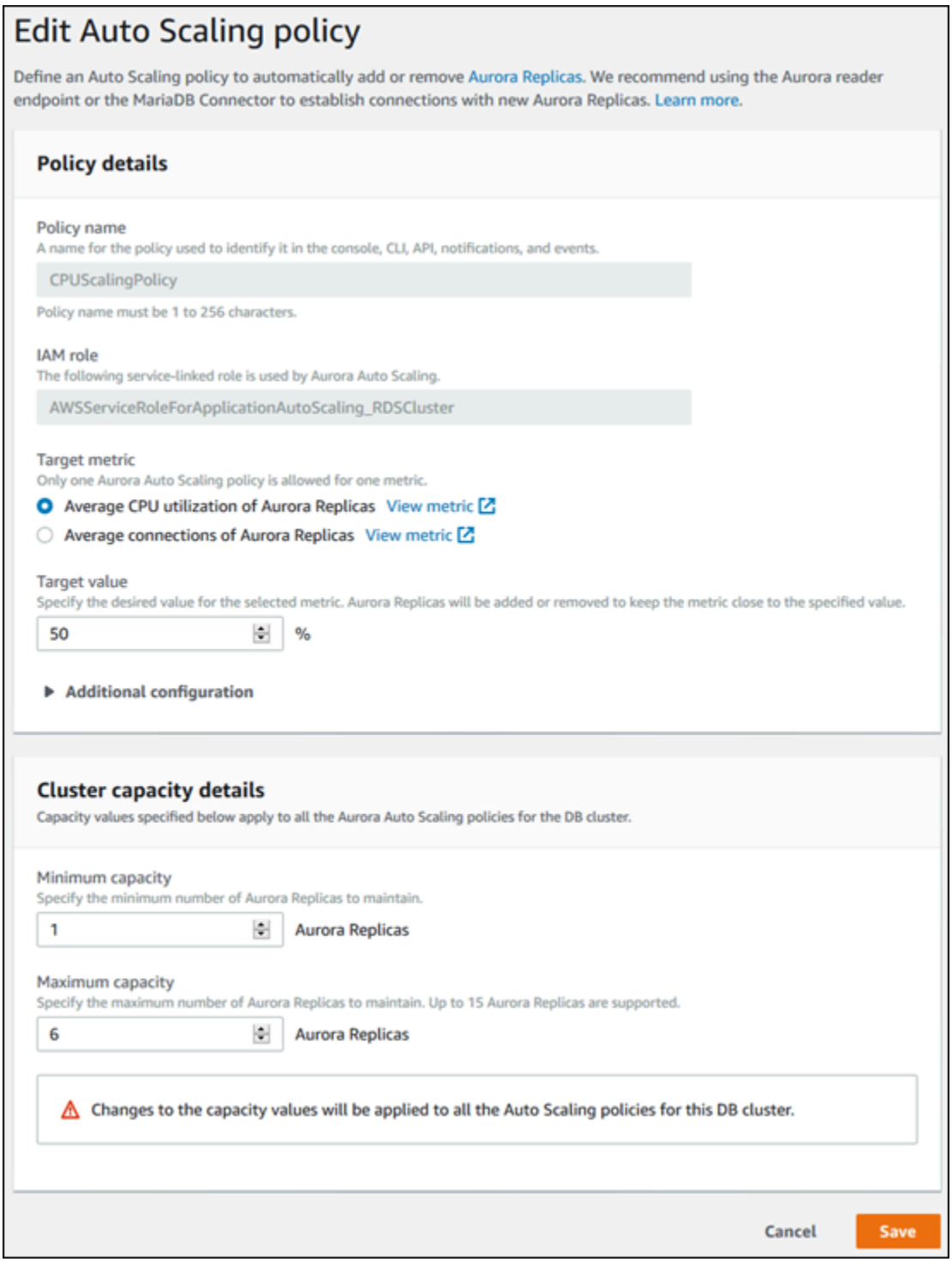
Console

Você pode editar uma política de escalabilidade usando o Console de gerenciamento da AWS.

Como editar uma política do Auto Scaling para um cluster de banco de dados Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados Aurora cuja política do Auto Scaling você deseja editar.
4. Escolha a guia Logs & events (Logs e eventos).
5. Na seção Auto Scaling Policies (Políticas do Auto Scaling), escolha a política do Auto Scaling e depois selecione Edit (Editar).
6. Faça as alterações na política.
7. Escolha Save (Salvar).

O exemplo a seguir mostra a caixa de diálogo Edit Auto Scaling policy.



AWS CLI ou API do Application Auto Scaling

Você pode usar a AWS CLI ou a API do Application Auto Scaling para editar uma política de escalabilidade da mesma forma que você aplica uma política de escalabilidade:

- Ao usar a AWS CLI, especifique o nome da política que você deseja editar no parâmetro `--policy-name`. Especifique novos valores para os parâmetros que você deseja alterar.
- Ao usar a API do Application Auto Scaling, especifique o nome da política que você deseja editar no parâmetro `PolicyName`. Especifique novos valores para os parâmetros que você deseja alterar.

Para ter mais informações, consulte Aplicar uma política de escalabilidade a um cluster de banco de dados Aurora.

Excluir uma política de ajuste de escala automático do cluster de banco de dados do Amazon Aurora

Você pode excluir uma política de escalabilidade usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Application Auto Scaling.

Console

Você pode excluir uma política de escalabilidade usando o Console de gerenciamento da AWS.

Como excluir uma política do Auto Scaling para um cluster de banco de dados Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados Aurora cuja política do Auto Scaling você deseja excluir.
4. Escolha a guia Logs & events (Logs e eventos).
5. Na seção Auto scaling policies (Políticas do Auto Scaling), escolha a política do Auto Scaling e depois selecione Delete (Excluir).

AWS CLI

Para excluir uma política de escalabilidade de seu cluster de banco de dados Aurora, use o comando `delete-scaling-policy` da AWS CLI com os seguintes parâmetros:

- `--policy-name` – o nome da política de escalabilidade.

- `--resource-id` – o identificador do recurso para o cluster de banco de dados do Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `--service-namespace` – Defina este valor como `rds`.
- `--scalable-dimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.

Example

No exemplo a seguir, a política de escalabilidade de rastreamento de destino chamada `myscalablepolicy` é excluída de um cluster de banco de dados Aurora chamado `myscalablecluster`.

Para Linux, macOS ou Unix:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

Para Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

API do Application Auto Scaling

Para excluir uma política de escalabilidade do seu cluster de banco de dados Aurora, use a operação `DeleteScalingPolicy` da API do Application Auto Scaling com os seguintes parâmetros:

- `PolicyName` – o nome da política de escalabilidade.
- `ServiceNamespace` – Defina este valor como `rds`.

- `ResourceID` – o identificador do recurso para o cluster de banco de dados do Aurora. Para este parâmetro, o tipo de recurso é `cluster` e o identificador exclusivo é nome do cluster de banco de dados Aurora, por exemplo `cluster:myscalablecluster`.
- `ScalableDimension` – Defina este valor como `rds:cluster:ReadReplicaCount`.

Example

No exemplo a seguir, a política de escalabilidade de rastreamento de destino chamada `myscalablepolicy` é excluída de um cluster de banco de dados Aurora chamado `myscalablecluster` com a API do Application Auto Scaling.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora

Você pode usar as seguintes opções para gerenciar a performance e a escalabilidade de clusters e instâncias de banco de dados do Aurora:

Tópicos

- Escalabilidade de armazenamento
- Escalabilidade de instâncias
- Escalabilidade de leitura
- Como gerenciar conexões
- Gerenciar planos de execução de consulta

Escalabilidade de armazenamento

O armazenamento do Aurora escala automaticamente com os dados no volume do cluster. À medida que os dados aumentam, o armazenamento do volume do cluster se expande com base na versão do mecanismo de banco de dados. Para ter informações sobre os tamanhos máximos de volume do cluster do Aurora para cada versão de mecanismo, consulte Limites de tamanho do Amazon Aurora. Para saber quais tipos de dados estão incluídos no volume do cluster, consulte Armazenamento do Amazon Aurora. Para obter detalhes sobre o tamanho máximo de uma versão específica, consulte Limites de tamanho do Amazon Aurora.

O tamanho do volume do cluster é avaliado de hora em hora para determinar os custos de armazenamento. Para obter informações sobre a definição de preço, consulte a página da definição de preço do Aurora.

Embora um volume de cluster do Aurora possa aumentar até muitos tebibytes, você só é cobrado pelo espaço que usa no volume. O mecanismo para determinar o espaço de armazenamento cobrado depende da versão do cluster do Aurora.

- Quando os dados do Aurora são removidos do volume do cluster, o espaço cobrado global diminui em um valor comparável. Esse comportamento de redimensionamento dinâmico ocorre quando espaços de tabela subjacentes são excluídos ou reorganizados para exigir menos espaço. Assim, é possível reduzir as cobranças de armazenamento excluindo tabelas e bancos de dados que não são mais necessários. O redimensionamento dinâmico se aplica a determinadas versões

do Aurora. Veja a seguir as versões do Aurora em que o volume do cluster é redimensionado dinamicamente à medida que você remove dados:

Mecanismo do banco de dados	Versões com redimensionamento dinâmico
Aurora MySQL	• Versão 3 (compatível com MySQL 8.0): todas as versões compatíveis • Versão 2 (compatível com MySQL 5.7): 2.11 e posterior
Aurora PostgreSQL	Todas as versões compatíveis
Aurora Serverless v2	Todas as versões compatíveis
Aurora Serverless v1	Indisponível

- Em versões do Aurora anteriores às dessa lista, o volume do cluster pode reutilizar o espaço que foi liberado quando você removeu dados, mas o volume em si nunca diminui.

O redimensionamento dinâmico se aplica a operações que removem ou redimensionam fisicamente espaços de tabela no volume do cluster. Assim, ele se aplica a instruções de SQL, como `DROP TABLE`, `DROP DATABASE`, `TRUNCATE TABLE` e `ALTER TABLE ... DROP PARTITION`. Ele não se aplica à exclusão de linhas usando a instrução `DELETE`. Se você excluir um grande número de linhas de uma tabela, será possível executar a instrução `OPTIMIZE TABLE` do Aurora MySQL ou a extensão `pg_repack` do Aurora PostgreSQL posteriormente para reorganizar a tabela e redimensionar dinamicamente o volume do cluster.

Para o Aurora MySQL, as seguintes considerações se aplicam:

- Depois de fazer upgrade do cluster de banco de dados para uma versão de mecanismo de banco de dados compatível com o redimensionamento dinâmico, e quando o recurso estiver habilitado nessa Região da AWS específica, qualquer espaço que for liberado posteriormente por determinadas instruções SQL, como `DROP TABLE`, será recuperável.

  Se o recurso estiver explicitamente desabilitado em uma Região da AWS específica, só será possível reutilizar (e não recuperar) o espaço, mesmo em versões compatíveis com o redimensionamento dinâmico.

O recurso foi habilitado para versões específicas do mecanismo de banco de dados (de 1.23.0 a 1.23.4, de 2.09.0 a 2.09.3 e 2.10.0) entre novembro de 2020 e março de 2022 e está habilitado por padrão em todas as versões subsequentes.

- Uma tabela é armazenada internamente em um ou mais fragmentos contíguos de tamanhos variados. Durante a execução das operações `TRUNCATE TABLE`, o espaço correspondente ao primeiro fragmento é reutilizável e não recuperável. Outros fragmentos são recuperáveis. Durante as operações `DROP TABLE`, o espaço correspondente a todo o espaço de tabela é recuperável.
- O parâmetro `innodb_file_per_table` afeta como o armazenamento da tabela é organizado. Quando as tabelas fazem parte do espaço de tabela do sistema, descartar a tabela não reduz o tamanho do espaço de tabela do sistema. Portanto, defina `innodb_file_per_table` como 1 para clusters de banco de dados do Aurora MySQL a fim de aproveitar ao máximo o redimensionamento dinâmico.
- Na versão 2.11 e posterior, o espaço de tabela temporário do InnoDB é excluído e recriado na reinicialização. Isso libera o espaço ocupado pelo espaço de tabela temporário para o sistema. Depois, o volume do cluster é redimensionado. Para aproveitar ao máximo o recurso de redimensionamento dinâmico, recomendamos que você atualize o cluster de banco de dados para a versão 2.11 ou posterior.

> **Note**
>
> O recurso de redimensionamento dinâmico não recupera espaço imediatamente quando as tabelas nos espaços de tabela são eliminadas, mas gradualmente a uma taxa de aproximadamente 10 TB por dia. O espaço no espaço de tabela do sistema não é recuperado porque ele nunca é removido. O espaço livre não recuperado em um espaço de tabela é reutilizado quando uma operação precisa de espaço nesse espaço de tabela. O atributo de redimensionamento dinâmico pode recuperar espaço de armazenamento somente quando o cluster está no estado disponível.

É possível verificar quanto espaço de armazenamento um cluster está usando monitorando a métrica `VolumeBytesUsed` no CloudWatch. Para receber mais informações sobre faturamento do armazenamento, consulte Como o armazenamento de dados do Aurora é faturado.

- No Console de gerenciamento da AWS, é possível ver essa figura em um gráfico exibindo a guia `Monitoring` na página de detalhes do cluster.

- Com a AWS CLI, é possível executar um comando semelhante ao seguinte exemplo do Linux. Substitua seus próprios valores pelas horas de início e término e o nome do cluster.

```
aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '6 hours ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" \
  --statistics Average Maximum Minimum \
  --dimensions Name=DBClusterIdentifier,Value=my_cluster_identifier
```

  Este comando gerará uma saída semelhante à seguinte:

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T21:25:00+00:00",
            "Average": 182871982080.0,
            "Minimum": 182871982080.0,
            "Maximum": 182871982080.0,
            "Unit": "Bytes"
        }
    ]
}
```

Os exemplos a seguir mostram como é possível rastrear o uso do armazenamento de um cluster do Aurora ao longo do tempo usando comandos da AWS CLI em um sistema Linux. Os parâmetros `--start-time` e `--end-time` definem o intervalo de tempo geral como um dia. O parâmetro `--period` solicita as medições em intervalos de uma hora. Não faz sentido escolher um valor pequeno de `--period`, porque as métricas são coletadas em intervalos, não continuamente. Além disso, as operações de armazenamento do Aurora às vezes continuam por um tempo em segundo plano após a conclusão da instrução SQL relevante.

O primeiro exemplo retorna a saída no formato JSON padrão. Os pontos de dados são retornados em ordem arbitrária, não classificados por carimbo de data/hora. É possível importar esses dados JSON em uma ferramenta de gráficos para fazer classificação e visualização.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id
```

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T19:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T00:40:00+00:00",
            "Maximum": 198573719552.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T05:40:00+00:00",
            "Maximum": 206827454464.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-04T17:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
... output omitted ...
```

Este exemplo retorna os mesmos dados que o anterior. O parâmetro `--output` representa os dados em formato de texto simples compacto. O comando `aws cloudwatch` canaliza sua saída para o comando `sort`. O parâmetro `-k` do comando `sort` classifica a saída pelo terceiro campo, que é o carimbo de data/hora no formato no Tempo Universal Coordenado (UTC).

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id \
  --output text | sort -k 3
VolumeBytesUsed
DATAPOINTS  182872522752.0  2020-08-04T17:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T18:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T19:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T20:41:00+00:00 Bytes
DATAPOINTS  187667791872.0  2020-08-04T21:41:00+00:00 Bytes
DATAPOINTS  190981029888.0  2020-08-04T22:41:00+00:00 Bytes
DATAPOINTS  195587244032.0  2020-08-04T23:41:00+00:00 Bytes
```

```
DATAPOINTS  201048915968.0  2020-08-05T00:41:00+00:00 Bytes
DATAPOINTS  205368492032.0  2020-08-05T01:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T02:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T03:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T04:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T05:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T06:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T07:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T08:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T09:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T10:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T11:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T12:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T13:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T14:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T15:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T16:41:00+00:00 Bytes
```

A saída classificada mostra quanto armazenamento foi usado no início e no término do período de monitoramento. Também é possível encontrar os pontos durante esse período quando o Aurora alocou mais armazenamento para o cluster. O exemplo a seguir usa comandos do Linux para reformatar os valores de início e término de `VolumeBytesUsed` como gigabytes (GB) e como gibibytes (GiB). Gigabytes representam unidades medidas em potências de 10 e são comumente usadas em discussões de armazenamento para discos rígidos rotacionais. Os gibibytes representam unidades medidas em potências de 2. As medidas e os limites de armazenamento do Aurora normalmente são indicados nas unidades de potência de 2, como gibibytes e tebibytes.

```
$ GiB=$((1024*1024*1024))
$ GB=$((1000*1000*1000))
$ echo "Start: $((182872522752/$GiB)) GiB, End: $((206833664000/$GiB)) GiB"
Start: 170 GiB, End: 192 GiB
$ echo "Start: $((182872522752/$GB)) GB, End: $((206833664000/$GB)) GB"
Start: 182 GB, End: 206 GB
```

A métrica de `VolumeBytesUsed` informa quanto armazenamento no cluster está incorrendo em cobranças. Assim, é melhor minimizar esse número quando for prático. No entanto, essa métrica não inclui um armazenamento que o Aurora usa internamente no cluster e não cobra por isso. Se o cluster estiver se aproximando do limite de armazenamento e puder ficar sem espaço, será mais útil monitorar a métrica `AuroraVolumeBytesLeftTotal` e tentar maximizar esse número. O exemplo

a seguir executa um cálculo semelhante ao anterior, mas para `AuroraVolumeBytesLeftTotal`, em vez de `VolumeBytesUsed`.

```
$ aws cloudwatch get-metric-statistics --metric-name "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_old_cluster_id \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS      140530528288768.0       2023-02-23T19:25:00+00:00       Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((69797067915264 / $TB)) TB remaining for this cluster"
69 TB remaining for this cluster
$ echo "$((69797067915264 / $TiB)) TiB remaining for this cluster"
63 TiB remaining for this cluster
```

Para um cluster executando o Aurora MySQL versão 2.09 ou posterior, ou Aurora PostgreSQL, o tamanho livre relatado por `VolumeBytesUsed` aumenta quando os dados são adicionados e diminui quando os dados são removidos. O exemplo a seguir mostra como. Este relatório mostra o tamanho máximo e mínimo de armazenamento de um cluster em intervalos de 15 minutos à medida que tabelas com dados temporários são criadas e descartadas. O relatório lista o valor máximo antes do valor mínimo. Assim, para entender como o uso do armazenamento mudou no intervalo de 15 minutos, interprete os números da direita para a esquerda.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Maximum Minimum --dimensions
 Name=DBClusterIdentifier,Value=my_new_cluster_id
  --output text | sort -k 4
VolumeBytesUsed
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T20:49:00+00:00 Bytes
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T21:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 14545305600.0 2020-08-05T21:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 15614263296.0 2020-08-05T23:19:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-05T23:49:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-06T00:19:00+00:00 Bytes
```

O exemplo a seguir mostra que, para clusters que estão executando versões compatíveis do Aurora MySQL ou Aurora PostgreSQL, o tamanho livre relatado por `AuroraVolumeBytesLeftTotal` reflete o limite de tamanho de 256 TiB. Para ter mais informações sobre versões compatíveis, consulte Limites de tamanho do Amazon Aurora.

```
$ aws cloudwatch get-metric-statistics --region us-east-1 --metric-name
 "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBClusterIdentifier,Value=pq-57 \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS 140515818864640.0 2020-08-05T20:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:26:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:56:00+00:00 Count
DATAPOINTS 140514866757632.0 2020-08-05T22:26:00+00:00 Count
DATAPOINTS 140511020580864.0 2020-08-05T22:56:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:26:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-06T00:26:00+00:00 Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((140515818864640 / $TB)) TB remaining for this cluster"
140 TB remaining for this cluster
$ echo "$((140515818864640 / $TiB)) TiB remaining for this cluster"
256 TiB remaining for this cluster
```

Escalabilidade de instâncias

Você pode escalar o cluster de banco de dados do Aurora, conforme a necessidade, modificando a classe da instância de banco de dados para cada instância de banco de dados do cluster. O Aurora é compatível com várias classes de instância de banco de dados otimizada para Aurora, dependendo da compatibilidade do mecanismo de banco de dados.

Mecanismo do banco de dados	Escalabilidade de instâncias
Amazon Aurora MySQL	Consulte Dimensionar instâncias de bancos de dados Aurora MySQL

Mecanismo do banco de dados	Escalabilidade de instâncias
Amazon Aurora PostgreSQL	Consulte Dimensionar instâncias de bancos de dados Aurora PostgreSQL

Escalabilidade de leitura

Você pode obter uma escalabilidade de leitura para o cluster de banco de dados Aurora criando até 15 réplicas do Aurora em um cluster de banco de dados. Cada réplica do Aurora retorna os mesmos dados do volume de cluster com atraso de réplica mínimo, geralmente considerado inferior a 100 milissegundos após a instância primária ter escrito uma atualização. Conforme seu tráfego de leitura aumenta, você pode criar mais réplicas do Aurora e conectar-se diretamente a elas para distribuir a carga de leitura para o seu cluster de banco de dados. As réplicas do Aurora não precisam ser da mesma classe da instância de banco de dados que a instância primária.

Para obter informações sobre como adicionar réplicas do Aurora a um cluster de banco de dados, consulte Adicionar réplicas do Aurora a um cluster de banco de dados.

Como gerenciar conexões

O número máximo de conexões permitido para uma instância de banco de dados do Aurora é determinado pelo parâmetro `max_connections` no parameter group do nível da instância para a instância de banco de dados. O valor padrão do parâmetro varia de acordo com a classe de instância de banco de dados usada para compatibilidade com a instância e o mecanismo de banco de dados.

Mecanismo do banco de dados	Valor padrão de max_connections
Amazon Aurora MySQL	Consulte Número máximo de conexões com uma instância de bancos de dados Aurora MySQL
Amazon Aurora PostgreSQL	Consulte Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL

> **Tip**
>
> Se suas aplicações abrem e fecham conexões com frequência ou mantêm um grande número de conexões de longa duração abertas, recomendamos usar o Amazon RDS Proxy. O RDS Proxy é um proxy de banco de dados totalmente gerenciado e altamente disponível que usa grupos de conexões para compartilhar conexões de banco de dados de forma segura e eficiente. Para saber mais sobre o RDS Proxy, consulte Amazon RDS Proxypara Aurora.

Gerenciar planos de execução de consulta

Se você usar o gerenciamento do plano de consultas do Aurora PostgreSQL, obterá controle sobre quais planos o otimizador executará. Para obter mais informações, consulte Gerenciar planos de execução de consultas do Aurora PostgreSQL.

Clonar um volume para um cluster de banco de dados do Amazon Aurora

Com a clonagem do Aurora, você pode criar um cluster que, inicialmente, compartilhe as mesmas páginas de dados do original mas seja um volume separado e independente. O processo foi projetado para ser rápido e econômico. O novo cluster e seu volume de dados associado é chamado de clone.. Criar um clone é mais rápido e eficiente em termos de espaço do que copiar fisicamente os dados usando outras técnicas, como restauração ou captura de tela.

Tópicos

- Visão geral da clonagem do Aurora
- Limitações da clonagem do Aurora
- Como a clonagem do Aurora funciona
- Criar um clone do Amazon Aurora
- Clonagem entre VPCs com o Amazon Aurora
- Clonar entre contas com o AWS RAM e Amazon Aurora

Visão geral da clonagem do Aurora

O Aurora usa um protocolo copy-on-write para criar um clone. Esse mecanismo usa um espaço adicional mínimo para criar um clone inicial. Quando o clone é criado pela primeira vez, o Aurora mantém uma única cópia dos dados usados pelo cluster de banco de dados do Aurora de origem e pelo novo cluster de banco de dados do Aurora (clonado). O armazenamento adicional é alocado somente quando as alterações são feitas nos dados (no volume de armazenamento do Aurora) pelo cluster de banco de dados do Aurora original ou pelo clone do cluster de banco de dados do Aurora. Para saber mais sobre o protocolo copy-on-write, consulte Como a clonagem do Aurora funciona.

A clonagem do Aurora serve principalmente para configurar rapidamente ambientes de teste usando seus dados de produção, sem arriscar corromper dados. É possível utilizar clones para vários tipos de aplicações, como:

- Experimente possíveis alterações (de esquema e de grupos de parâmetros, por exemplo) para avaliar todos os impactos.
- Execute operações com workloads intensivas, como exportar dados ou executar consultas analíticas no clone.

- Crie uma cópia do cluster de banco de dados de produção para desenvolvimento, teste ou outras finalidades.

É possível criar mais de um clone do mesmo cluster de banco de dados do Aurora. Também é possível criar vários clones a partir de outro clone.

Após criar um clone do Aurora, você pode configurar as instâncias de banco de dados do Aurora de forma diferente do cluster de banco de dados do Aurora de origem. Por exemplo, talvez você não precise de um clone para fins de desenvolvimento para atender aos mesmos requisitos de alta disponibilidade que o cluster de banco de dados de produção do Aurora de origem. Nesse caso, é possível configurar o clone com uma única instância de banco de dados Aurora em vez das várias instâncias de banco de dados usadas pelo cluster de banco de dados do Aurora.

Quando você cria um clone usando uma configuração de implantação diferente da origem, o clone é criado com a versão secundária mais recente do mecanismo de banco de dados do Aurora da origem.

Quando você cria clones a partir de seus clusters de banco de dados do Aurora, os clones são criados em sua conta da AWS, a mesma conta que contém o cluster do banco de dados do Aurora de origem. No entanto, também é possível compartilhar clusters e clones de banco de dados do Aurora Serverless v2 e provisionados do Aurora com outras contas da AWS. Para ter mais informações, consulte Clonar entre contas com o AWS RAM e Amazon Aurora.

Ao concluir o uso do clone para testes, desenvolvimento ou outras finalidades, você poderá excluí-lo.

Limitações da clonagem do Aurora

Atualmente, a clonagem do Aurora tem as seguintes limitações:

- Você pode criar quantos clones quiser, até o número máximo de clusters de banco de dados permitido na Região da AWS.
- Você pode criar até 15 clones com o protocolo copy-on-write. No entanto, depois de criar quinze clones, o próximo é uma cópia completa. O protocolo full-copy funciona como uma recuperação para um ponto no tempo.
- Não é possível criar um clone em uma região da AWS diferente da região de origem do cluster de banco de dados do Aurora.
- Não é possível criar o clone de um cluster de banco de dados do Aurora sem o recurso de consulta paralela para um cluster que usa consulta paralela. Para levar dados a um cluster que

usa a consulta paralela, crie um snapshot do cluster original e restaure-o para um cluster que está usando o recurso de consulta paralela.

- Não é possível criar um clone a partir de um cluster de banco de dados do Aurora que não tenha instâncias de banco de dados. Só é possível clonar clusters de banco de dados do Aurora com pelo menos uma instância de banco de dados.
- É possível criar um clone em uma virtual private cloud (VPC) diferente da do cluster de banco de dados do Aurora. Nesse caso, as sub-redes dessas VPCs devem ser mapeadas mas mesmas zonas de disponibilidade.
- É possível criar um clone provisionado do Aurora a partir de um cluster de banco de dados provisionado do Aurora.
- Os clusters com instâncias do Aurora Serverless v2 seguem as mesmas regras que os clusters provisionados.
- Para Aurora Serverless v1:
  - É possível criar um clone provisionado a partir de um cluster de banco de dados do Aurora Serverless v1.
  - É possível criar um clone do Aurora Serverless v1 de um cluster de banco de dados do provisionado ou do Aurora Serverless v1.
  - Não é possível criar um clone do Aurora Serverless v1 de um cluster de banco de dados provisionado do Aurora não criptografado.
  - No momento, a clonagem entre contas não é compatível com a clonagem de clusters de banco de dados do Aurora Serverless v1. Para ter mais informações, consulte Limitações da clonagem entre contas.
  - O cluster de banco de dados do Aurora Serverless v1 clonado tem o mesmo comportamento e limitações que qualquer cluster de banco de dados do Aurora Serverless v1. Para ter mais informações, consulte Usar o Amazon Aurora Serverless v1.
  - Os clusters de banco de dados do Aurora Serverless v1 são sempre criptografados Quando você clonar um banco de dados do Aurora Serverless v1 em um cluster de banco de dados provisionado do Aurora, o cluster de banco de dados do Aurora provisionado será criptografado. É possível escolher a chave de criptografia, mas não é possível desabilitar a criptografia. Para clonar de um cluster de banco de dados provisionado do Aurora para o Aurora Serverless v1, é necessário começar com um cluster de banco de dados provisionado do Aurora provisionado e criptografado.

Como a clonagem do Aurora funciona

A clonagem do Aurora funciona na camada de armazenamento de um cluster de banco de dados do Aurora. Utiliza um protocolo copy-on-write que é rápido e eficiente em termos de mídia durável subjacente compatível com o volume de armazenamento do Aurora. Saiba mais sobre os volumes de cluster do Aurora em Visão geral do armazenamento do Amazon Aurora.

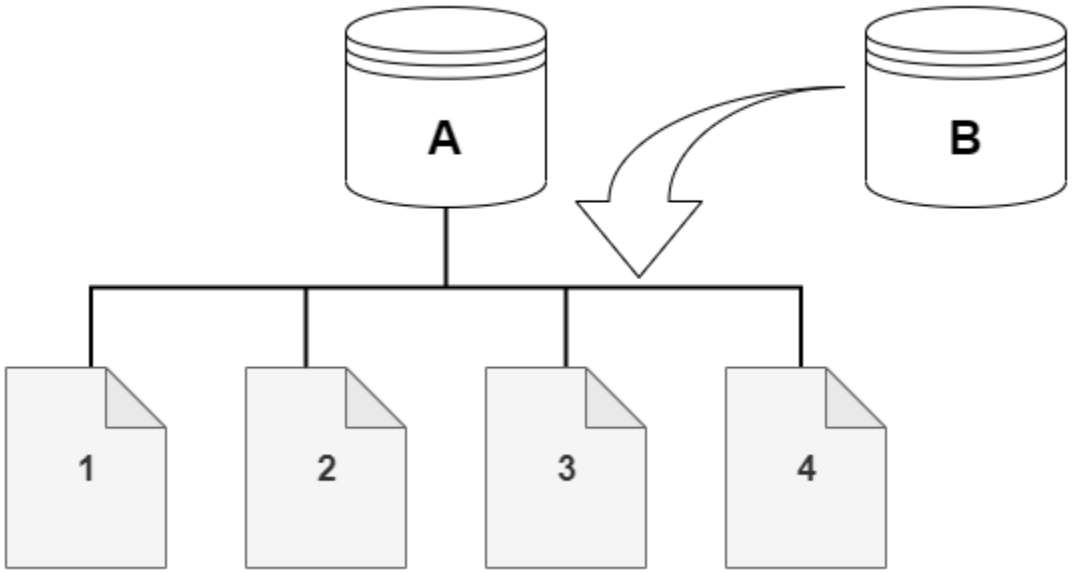
Tópicos

- Noções sobre o protocolo copy-on-write
- Excluir um volume de cluster de origem

Noções sobre o protocolo copy-on-write

Um cluster de banco de dados do Aurora armazena dados em páginas no volume de armazenamento subjacente do Aurora.

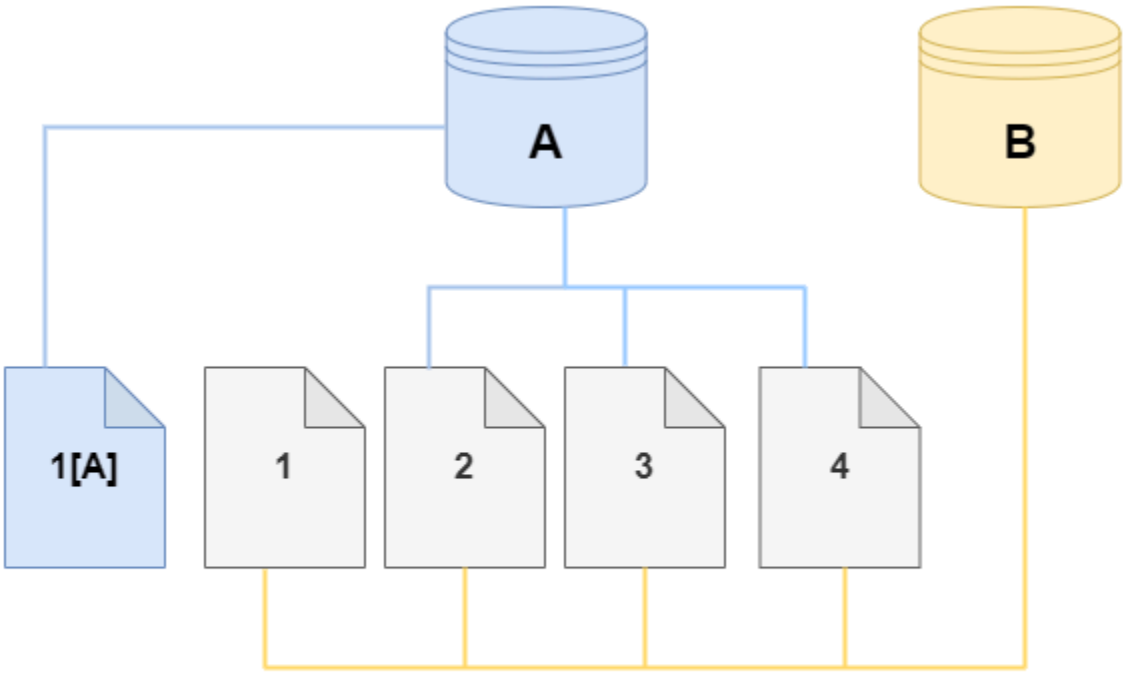
Por exemplo, no diagrama a seguir, você encontra um cluster de banco de dados do Aurora (A) que tem quatro páginas de dados: 1, 2, 3 e 4. Imagine que um clone, B é criado a partir do cluster de banco de dados do Aurora. Quando o clone é criado, nenhum dado é copiado. Em vez disso, o clone aponta para o mesmo conjunto de páginas que o cluster de banco de dados do Aurora de origem.



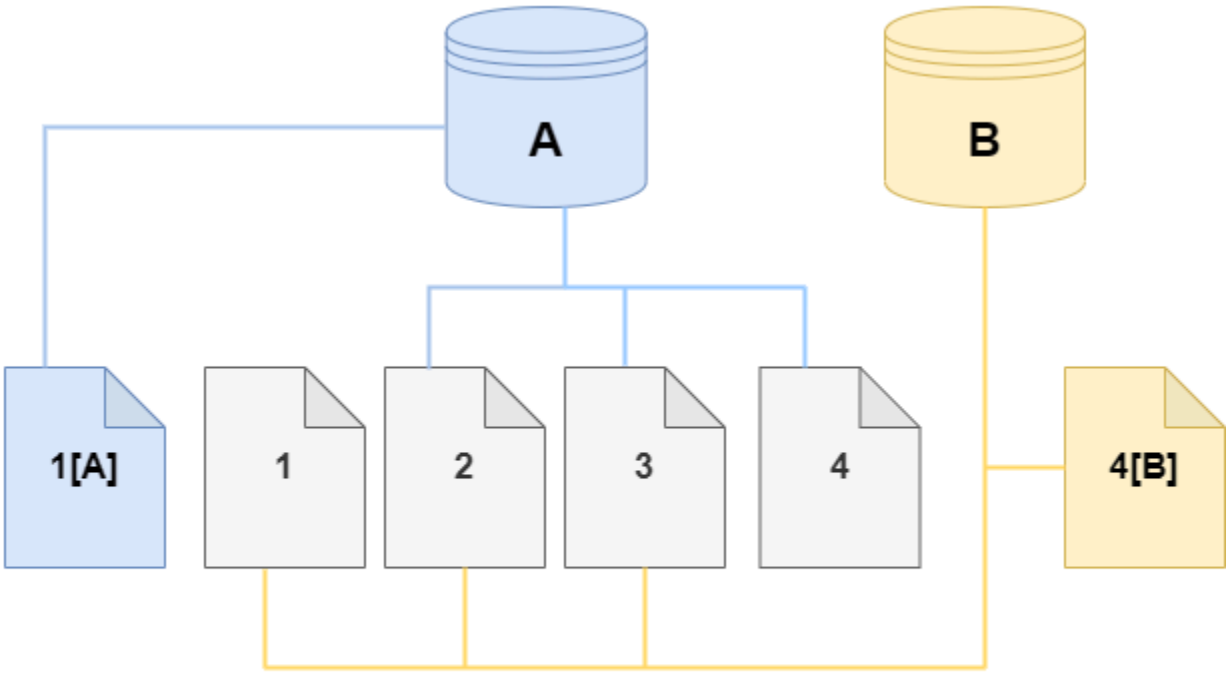
Quando o clone é criado, geralmente não é necessário armazenamento adicional. O protocolo copiar ao escrever usa o mesmo segmento na mídia de armazenamento física que no segmento de origem.

O armazenamento adicional é necessário somente se a capacidade do segmento de origem não for suficiente para todo o segmento do clone. Se for esse o caso, o segmento de origem será copiado para outro dispositivo físico.

Nos diagramas a seguir, você encontra um exemplo do protocolo copy-on-write em ação usando o mesmo cluster A e seu clone (B), conforme mostrado anteriormente. Digamos que você faça uma alteração no cluster de banco de dados do Aurora (A) que resulte em uma alteração nos dados mantidos na página 1. Em vez de gravar na página original 1, o Aurora cria uma nova página 1[A]. O volume do cluster de banco de dados do Aurora para o cluster (A) agora aponta para a página 1[A], 2, 3 e 4, enquanto o clone (B) ainda faz referência às páginas originais.



No clone, uma alteração é feita na página 4 no volume de armazenamento. Em vez de gravar na página original 4, o Aurora cria uma nova página 4[B]. O clone agora aponta para as páginas 1, 2, 3 e para a página 4[B], enquanto o cluster (A) continua apontando para 1[A], 2, 3 e 4.



À medida que ocorrerem mais alterações ao longo do tempo no volume do cluster do Aurora original e no clone, será necessário mais armazenamento incremental para capturar e armazenar as alterações.

Excluir um volume de cluster de origem

Inicialmente, o volume do clone compartilha as mesmas páginas de dados do volume original com base no qual o clone foi criado. Enquanto o volume original existir, o volume do clone só será considerado o proprietário das páginas que o clone criou ou modificou. Assim, a métrica `VolumeBytesUsed` do volume do clone começa pequena e só cresce à medida que os dados divergem entre o cluster original e o clone. Em relação às páginas idênticas entre o volume de origem e o clone, as cobranças de armazenamento se aplicam somente ao cluster original. Para obter mais informações sobre métricas do `VolumeBytesUsed`, consulte Métricas no nível do cluster do Amazon Aurora.

Quando você exclui um volume do cluster de origem com um ou mais clones associados a ele, os dados nos volumes do cluster dos clones não são alterados. O Aurora preserva as páginas

que pertenciam anteriormente ao volume do cluster de origem. O Aurora redistribui a cobrança de armazenamento das páginas que pertenciam ao cluster excluído. Por exemplo, suponha que um cluster original tenha dois clones e, depois, o cluster original seja excluído. Metade das páginas de dados pertencentes ao cluster original agora pertence a um clone. A outra metade das páginas pertence ao outro clone.

Se você excluir o cluster original, ao criar ou excluir mais clones, o Aurora continuará a redistribuir a propriedade das páginas de dados entre todos os clones que compartilham as mesmas páginas. Assim, talvez você descubra que o valor da métrica `VolumeBytesUsed` muda para o volume do cluster de um clone. O valor da métrica pode diminuir à medida que mais clones são criados e a propriedade das páginas é distribuída por mais clusters. O valor da métrica também pode aumentar à medida que os clones são excluídos e a propriedade das páginas é atribuída a um número menor de clusters. Para ter informações sobre como as operações de gravação afetam as páginas de dados em volumes de clones, consulte Noções sobre o protocolo copy-on-write.

Quando o cluster original e os clones pertencem à mesma conta da AWS, todas as cobranças de armazenamento desses clusters se aplicam à mesma conta da AWS. Se alguns dos clusters forem clones entre contas, a exclusão do cluster original poderá gerar cobranças adicionais de armazenamento para as contas da AWS que possuem os clones entre contas.

Por exemplo, suponha que um volume de cluster tenha mil páginas de dados usadas antes de você criar qualquer clone. Quando você clona esse cluster, inicialmente o volume do clone tem zero páginas utilizadas. Se o clone fizer modificações em cem páginas de dados, somente essas cem páginas serão armazenadas no volume do clone e marcadas como utilizadas. As outras novecentas páginas inalteradas do volume principal serão compartilhadas pelos dois clusters. Nesse caso, o cluster principal terá cobranças de armazenamento para mil páginas e o volume do clone para cem páginas.

Se você excluir o volume de origem, as cobranças de armazenamento do clone incluirão as cem páginas alteradas, além das novecentas páginas compartilhadas do volume original, totalizando mil páginas.

Criar um clone do Amazon Aurora

Você pode criar um clone na mesma conta da AWS que o cluster de banco de dados do Aurora de origem. Para fazer isso, você pode usar o Console de gerenciamento da AWS ou a AWS CLI e os procedimentos a seguir.

Para permitir que outra conta da AWS crie um clone ou compartilhar um clone com outra conta da AWS, use os procedimentos contidos em Clonar entre contas com o AWS RAM e Amazon Aurora.

Console

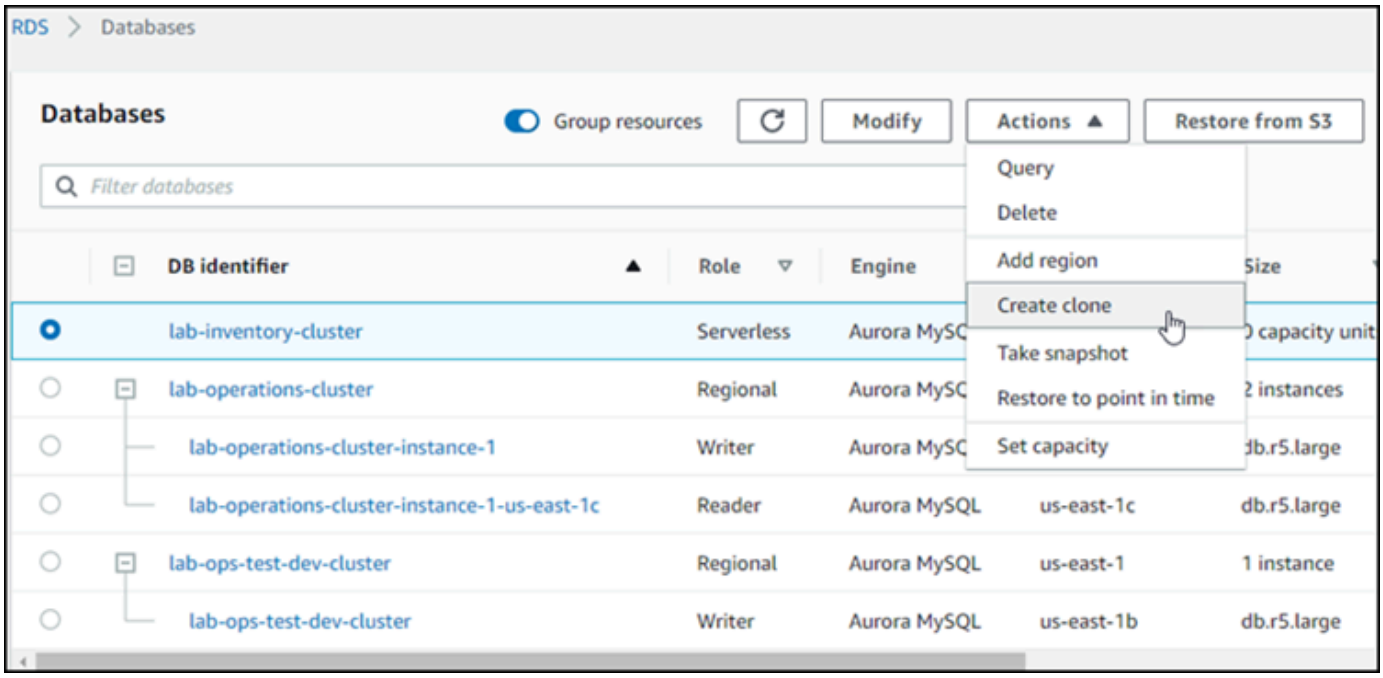
O procedimento a seguir descreve como clonar um cluster de banco de dados do Aurora usando o Console de gerenciamento da AWS.

Criar um clone usando os resultados do Console de gerenciamento da AWS em um cluster de banco de dados do Aurora com uma instância de banco de dados do Aurora.

Essas instruções se aplicam a clusters de banco de dados que são da mesma conta da AWS que está criando o clone. Se o cluster de banco de dados for de uma conta da AWS diferente, consulte Clonar entre contas com o AWS RAM e Amazon Aurora.

Para criar um clone de um cluster de banco de dados da sua conta da AWS usando o Console de gerenciamento da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha seu cluster de banco de dados do Aurora na lista e em Actions (Ações), escolha Create clone (Criar clone).



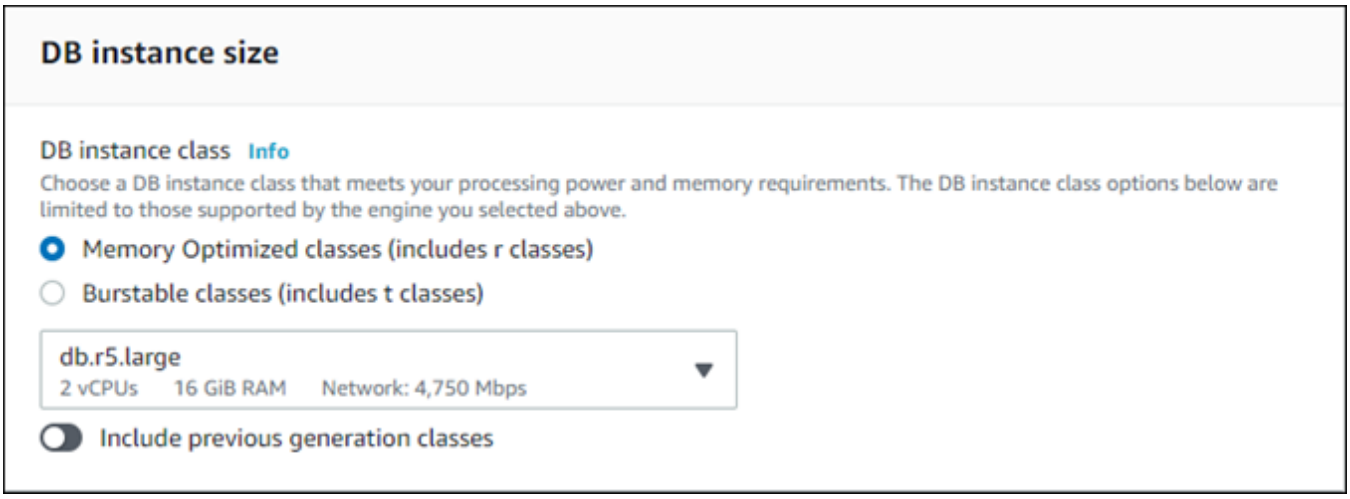
Abre-se a página Criar clone, onde é possível definir Configurações, Conectividade e outras opções para o clone de cluster de banco de dados do Aurora.

4. Em Identificador da instância de banco de dados, insira o nome que você deseja dar ao cluster de banco de dados clonado do Aurora.
5. Para clusters de banco de dados do Aurora Serverless v1, selecione Provisionado ou Sem servidor para Tipo de capacidade.

   Só será possível escolher Serverless (Sem servidor) se o cluster de banco de dados do Aurora de origem for um cluster de banco de dados do Aurora Serverless v1 ou um cluster de banco de dados do Aurora provisionado criptografado.
6. Para clusters de banco de dados provisionados ou do Aurora Serverless v2, selecione Aurora I/O-Optimized ou Aurora Standard para Configuração do armazenamento em cluster.

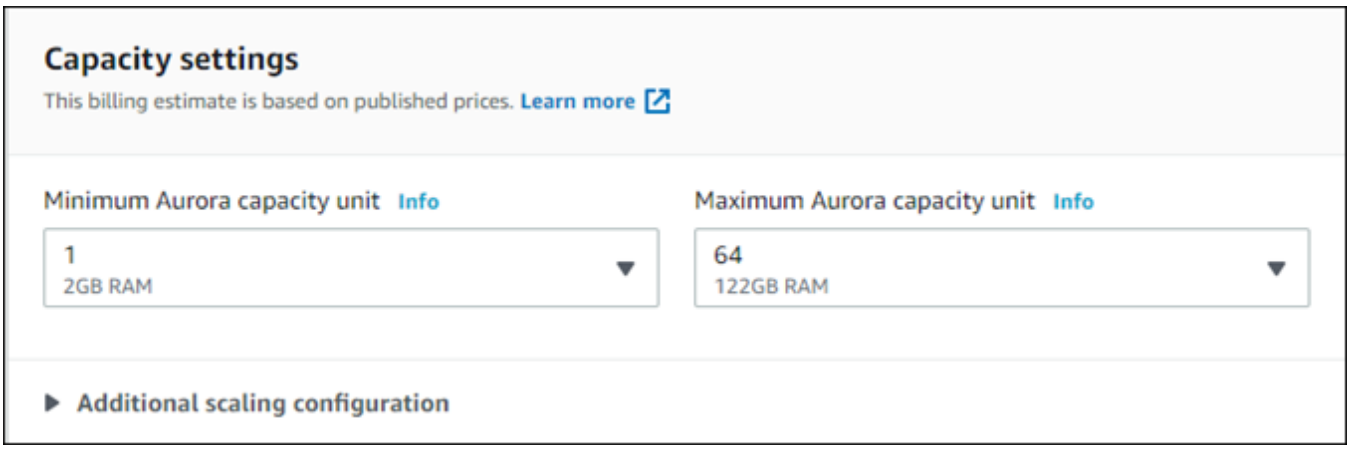
   Para ter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.
7. Escolha o tamanho da instância de banco de dados ou a capacidade do cluster de banco de dados:

   - Para um clone provisionado, selecione uma Classe da instância de banco de dados.



     Você pode aceitar a configuração fornecida ou usar uma classe de instância de banco de dados diferente para o clone.

   - Para um clone do Aurora Serverless v1 ou do Aurora Serverless v2, selecione as Configurações de capacidade.



É possível aceitar as configurações fornecidas ou alterá-las para o clone.

8. Selecione outras configurações, conforme necessário, para o clone. Para saber mais sobre as configurações de cluster e instância de banco de dados do Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.
9. Selecione Criar clone.

Ao ser criado, o clone é listado com seus outros clusters de banco de dados Aurora na seção Databases (Bancos de dados) do console e exibe seu estado atual. O clone está pronto para uso quando o estado é Available (Disponível).

AWS CLI

Usar a AWS CLI para clonar o cluster de banco de dados do Aurora envolve etapas separadas para criar o cluster clonado e adicionar uma ou mais instâncias de banco de dados a ele.

O comando `restore-db-cluster-to-point-in-time` da AWS CLI utilizado gera um cluster de banco de dados do Aurora com os mesmos dados de armazenamento que o cluster original, mas não instâncias de banco de dados do Aurora. Será necessário criar as instâncias de banco de dados separadamente após o clone ficar disponível. É possível selecionar o número de instâncias de banco de dados e suas classes de instância para dar ao clone mais ou menos capacidade computacional do que o cluster original. As etapas do processo são as seguintes:

1. Crie o clone usando o comando restore-db-cluster-to-pontual da CLI.
2. Crie a instância de banco de dados de gravador para o clone usando o comado create-db-instance da CLI.
3. (Opcional) Execute comandos adicionais create-db-instance da CLI para adicionar uma ou mais instâncias de leitor ao cluster clonado. O uso de instâncias de leitor ajuda a melhorar os aspectos

de alta disponibilidade e escalabilidade de leitura do clone. É possível ignorar essa etapa caso você pretenda usar o clone apenas para desenvolvimento e teste.

Tópicos

- Criar o clone
- Verificar o status e obter detalhes do clone
- Criar a instância de banco de dados do Aurora para seu clone
- Parâmetros a serem usados para clonagem

Criar o clone

Use o comando `restore-db-cluster-to-point-in-time` da CLI para criar o cluster clonado inicial.

Como criar um clone do cluster de banco de dados do Aurora original

- Use o comando de CLI de `restore-db-cluster-to-point-in-time`. Especifique os valores para os parâmetros a seguir. Nesse caso típico, o clone usa o mesmo modo de mecanismo do cluster original, provisionado ou o Aurora Serverless v1.
  - `--db-cluster-identifier` — escolha um nome significativo para o clone. Nomeie o clone ao usar o comando restore-db-cluster-to-point-in-time da CLI. Em seguida, passe o nome do clone no comando create-db-instance da CLI.
  - `--restore-type`: use `copy-on-write` para criar um clone do cluster de banco de dados de origem Sem esse parâmetro, `restore-db-cluster-to-point-in-time` restaura o cluster de banco de dados do Aurora em vez de criar um clone.
  - `--source-db-cluster-identifier`: use o nome do cluster de banco de dados do Aurora de origem que você deseja clonar.
  - `--use-latest-restorable-time`: esse valor aponta para dados de volume restauráveis mais recentes para o cluster de banco de dados de origem. Use-o para criar clones.

O exemplo a seguir mostra a criação de um clone chamado `my-clone` a partir de um cluster chamado `my-source-cluster`.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

O comando retorna o objeto JSON que contém detalhes do clone. Verifique se o cluster de banco de dados clonado está disponível antes de tentar criar a instância de banco de dados para o clone. Para ter mais informações, consulte Verificar o status e obter detalhes do clone.

Por exemplo, suponha que você tenha um cluster chamado `tpch100g` que deseja clonar. O exemplo do Linux a seguir cria um cluster clonado chamado `tpch100g-clone`, uma instância de gravador do Aurora Serverless v2 denominada `tpch100g-clone-instance` e uma instância de leitor provisionada denominada `tpch100g-clone-instance-2` para o novo cluster.

Não é necessário fornecer alguns parâmetros, como `--master-username` e `--master-user-password`. O Aurora determina automaticamente os parâmetros do cluster original. Você precisa especificar o mecanismo de banco de dados a ser usado. Assim, o exemplo testa o novo cluster para determinar o valor certo a ser usado para o parâmetro `--engine`.

Esse exemplo também inclui a opção `--serverless-v2-scaling-configuration` ao criar o cluster clonado. Dessa forma, é possível adicionar instâncias do Aurora Serverless v2 ao clone mesmo que o cluster original não tenha usado o Aurora Serverless v2.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier tpch100g \
  --db-cluster-identifier tpch100g-clone \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
  --restore-type copy-on-write \
  --use-latest-restorable-time

$ aws rds describe-db-clusters \
  --db-cluster-identifier tpch100g-clone \
```

```
    --query '*[].[Engine]' \
    --output text
aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.serverless \
  --engine aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance-2 \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.r6g.2xlarge \
  --engine aurora-mysql
```

Como criar um clone com um modo de mecanismo diferente do cluster de banco de dados do Aurora original

- Este procedimento só se aplica a versões mais antigas do mecanismo que comportam o Aurora Serverless v1. Suponha que você tenha um cluster do Aurora Serverless v1 e queira criar um clone que seja um cluster provisionado. Nesse caso, use o comando `restore-db-cluster-to-point-in-time` da CLI e especifique valores de parâmetros semelhantes aos do exemplo anterior, além destes parâmetros adicionais:

  - `--engine-mode`: use esse parâmetro somente para criar clones de um modo de mecanismo diferente do cluster de banco de dados do Aurora de origem. Esse parâmetro só se aplica a versões mais antigas do mecanismo que comportam o Aurora Serverless v1. Escolha o valor para passar com `--engine-mode`, desta maneira:

    - Use `--engine-mode provisioned` para criar um clone de cluster de banco de dados do Aurora provisionado a partir de um cluster de banco de dados do Aurora Serverless.

      > **Note**
      >
      > Se pretende usar o Aurora Serverless v2 com um cluster clonado do Aurora Serverless v1, especifique o modo de mecanismo para o clone como `provisioned`. Depois, execute etapas adicionais de atualização e migração.

  - Use `--engine-mode serverless` para criar um clone do Aurora Serverless v1 de um cluster de banco de dados do Aurora provisionado. Ao especificar o modo de mecanismo `serverless`, você também pode escolher a `--scaling-configuration`.
- `--scaling-configuration`: (opcional) use com `--engine-mode serverless` para configurar a capacidade mínima e máxima de um clone do Aurora Serverless v1. Se você não usar esse parâmetro, o Aurora criará um clone do Aurora Serverless v1 usando os valores de capacidade do Aurora Serverless v1 padrão para o mecanismo de banco de dados.

O exemplo a seguir cria um clone provisionado denominado `my-clone`, de um cluster de banco de dados do Aurora Serverless v1 chamado `my-source-cluster`.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --engine-mode provisioned \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --engine-mode provisioned ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Esses comandos retornam o objeto JSON que contém detalhes do clone necessário para criar a instância de banco de dados. Não é possível fazer isso até que o status do clone (o cluster de banco de dados do Aurora vazio) tenha o status Available (Disponível).

> **Note**
>
> O comando restore-db-cluster-to-point-in-time da AWS CLI restaura apenas o cluster de banco de dados, e não as instâncias de banco de dados para esse cluster de banco de dados. Execute o comando create-db-instance para criar instâncias de banco de dados para o cluster de banco de dados restaurado. Com esse comando, especifique o identificador do

cluster de banco de dados restaurado como o parâmetro `--db-cluster-identifier`. Você pode criar instâncias de banco de dados somente após o comando `restore-db-cluster-to-point-in-time` tiver sido concluído e o cluster de banco de dados estiver disponível.

Suponha que você comece com um cluster do Aurora Serverless v1 e pretenda migrá-lo para um cluster do Aurora Serverless v2. Crie um clone provisionado do cluster do Aurora Serverless v1 como a etapa inicial da migração. Para ver o procedimento completo, incluindo todas as atualizações de versão necessárias, consulte Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.

Verificar o status e obter detalhes do clone

É possível usar o comando a seguir para conferir o status do cluster clonado recém-criado.

```
$ aws rds describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]'
 --output text
```

Ou você pode obter o status e os outros valores necessários para criar a instância de banco de dados para seu clone usando a seguinte consulta da AWS CLI.

Para Linux, macOS ou Unix:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}'
```

Para Windows:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}"
```

Essa consulta retorna uma saída semelhante à saída abaixo.

```
[
  {
        "Status": "available",
        "Engine": "aurora-mysql",
```

```
        "EngineVersion": "8.0.mysql_aurora.3.04.1",
        "EngineMode": "provisioned"
    }
]
```

Criar a instância de banco de dados do Aurora para seu clone

Use o comando create-db-instance da CLI para criar a instância de banco de dados para o clone do Aurora Serverless v2 ou provisionado. Não crie uma instância de banco de dados para um clone do Aurora Serverless v1.

A instância de banco de dados herda as propriedades `--master-username` e `--master-user-password` do cluster de banco de dados de origem.

O exemplo a seguir cria uma instância de banco de dados para o clone provisionado.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.r6g.2xlarge \
    --engine aurora-mysql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.r6g.2xlarge ^
    --engine aurora-mysql
```

O exemplo a seguir cria uma instância de banco de dados do Aurora Serverless v2 para um clone que usa uma versão de mecanismo compatível com o Aurora Serverless v2.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
```

```
    --db-instance-class db.serverless \
  --engine aurora-postgresql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.serverless ^
    --engine aurora-mysql
```

Parâmetros a serem usados para clonagem

A tabela a seguir resume os vários parâmetros usados com `restore-db-cluster-to-point-in-time`para clonar clusters de banco de dados do Aurora.

Parâmetro	Descrição
`--source-db-cluster-identifier`	Use o nome do cluster de banco de dados do Aurora original que você deseja clonar.
`--db-cluster-identifier`	Selecione um nome significativo para o clone ao criá-lo com o comando `restore-db-cluster-to-point-in-time` . Em seguida, passe esse nome para o comando `create-db-instance` .
`--restore-type`	Especifique `copy-on-write` como `--restore-type` para criar um clone do cluster de banco de dados original em vez de restaurar o cluster de banco de dados do Aurora original.
`--use-latest-restorable-time`	Esse valor aponta para os dados de volume restauráveis mais recentes para o cluster de banco de dados de origem. Use-o para criar clones.
`--serverless-v2-scaling-configuration`	(Versões mais recentes que comportam o Aurora Serverless v2) Use esse parâmetro para configurar a capacidade mínima e máxima de um clone do Aurora Serverless v2. Se você não especificar esse parâmetro , não poderá criar nenhuma instância do Aurora Serverless v2 no cluster do clone até modificar o cluster para adicionar esse atributo.

Parâmetro	Descrição
`--engine-mode`	(Somente versões mais antigas que comportam o Aurora Serverless v1) Use esse parâmetro para criar clones de um tipo diferente do cluster de banco de dados do Aurora de origem, com um dos seguintes valores: • Use `provisioned` para criar um clone provisionado de um cluster de banco de dados do Aurora Serverless v1. • Use `serverless` para criar um clone do Aurora Serverless v1 de um cluster de banco de dados do Aurora Serverless v2 ou provision ado. Ao especificar o modo de mecanismo `serverless`, você também pode escolher a `--scaling-configuration`.
`--scaling-configuration`	(Somente versões mais antigas que comportam o Aurora Serverless v1) Use esse parâmetro para configurar a capacidade mínima e máxima de um clone do Aurora Serverless v1. Se você não especificar esse parâmetro, o Aurora criará o clone usando os valores de capacidade padrão para o mecanismo de banco de dados.

Para ter informações sobre clonagem entre VPCs e entre contas, consulte as seções a seguir.

Tópicos

- Clonagem entre VPCs com o Amazon Aurora
- Clonar entre contas com o AWS RAM e Amazon Aurora

Clonagem entre VPCs com o Amazon Aurora

Suponha que você queira impor diferentes controles de acesso à rede no cluster original e no clone. Por exemplo, é possível usar a clonagem para fazer uma cópia de um cluster do Aurora de produção em uma VPC diferente para desenvolvimento e testes. Ou você pode criar um clone como parte de uma migração de sub-redes públicas para privadas a fim de melhorar a segurança do banco de dados.

As seções a seguir demonstram como definir a configuração de rede do clone para que o cluster original e o clone possam acessar os mesmos nós de armazenamento do Aurora, mesmo de sub-

redes ou VPCs diferentes. A verificação prévia de recursos da rede pode evitar erros durante a clonagem que talvez sejam difíceis de diagnosticar.

Se você não souber como o Aurora interage com VPCs, sub-redes e grupos de sub-redes de banco de dados, consulte Amazon VPC e Amazon Aurora primeiro. Você pode seguir os tutoriais dessa seção para criar esses tipos de recurso no console da AWS e entender como eles se encaixam.

Como as etapas exigem a alternância entre os serviços RDS e EC2, os exemplos usam comandos da AWS CLI para ajudar a entender como automatizar essas operações e salvar o resultado.

Tópicos

- Antes de começar
- Reunir informações sobre o ambiente de rede
- Criar recursos de rede para o clone
- Criar um clone do Aurora com novas configurações de rede
- Mover um cluster de sub-redes públicas para privadas
- Exemplo completo da criação de um clone entre VPCs

Antes de começar

Antes de começar a configurar um clone entre VPCs, verifique se você tem os seguintes recursos:

- Um cluster de banco de dados do Aurora para usar como a origem da clonagem. Se você estiver criando um cluster de banco de dados do Aurora pela primeira vez, consulte os tutoriais em Conceitos básicos do Amazon Aurora para configurar um cluster usando o mecanismo de banco de dados do MySQL ou PostgreSQL.
- Uma segunda VPC, se você pretende criar um clone entre VPCs. Se você não tiver uma VPC para usar com o clone, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4) ou Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla).

Reunir informações sobre o ambiente de rede

Com a clonagem entre VPCs, o ambiente de rede pode diferir substancialmente entre o cluster original e o clone. Antes de criar o clone, colete e registre informações sobre a VPC, as sub-redes, o grupo de sub-redes de banco de dados e as AZs usadas no cluster original. Dessa forma, é possível minimizar a possibilidade de problemas. Se ocorrer um problema de rede, você não precisará

interromper nenhuma atividade de solução de problemas para pesquisar informações de diagnóstico. As seções a seguir mostram exemplos da CLI para coletar esses tipos de informação. É possível salvar os detalhes em qualquer formato conveniente para consulta ao criar o clone e solucionar qualquer problema.

- Etapa 1: verificar as zonas de disponibilidade do cluster original
- Etapa 2: verificar o grupo de sub-redes de banco de dados do cluster original
- Etapa 3: verificar as sub-redes do cluster original
- Etapa 4: verificar as zonas de disponibilidade das instâncias de banco de dados no cluster original
- Etapa 5: verificar as VPCs que você pode usar para o clone

Etapa 1: verificar as zonas de disponibilidade do cluster original

Antes de criar o clone, verifique quais AZs o cluster original usa para armazenamento. Conforme explicado em Armazenamento do Amazon Aurora, o armazenamento de cada cluster do Aurora está associado a exatamente três AZs. Como o Clusters de banco de dados do Amazon Aurora aproveita a separação entre computação e armazenamento, essa regra se aplica independentemente de quantas instâncias estão no cluster.

Por exemplo, execute um comando da CLI como o indicado a seguir, substituindo o nome do seu cluster por *my_cluster*. O exemplo a seguir produz uma lista classificada alfabeticamente pelo nome da AZ.

```
aws rds describe-db-clusters \
  --db-cluster-identifier my_cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' \
  --output text
```

O exemplo a seguir mostra a saída de amostra do comando `describe-db-clusters` anterior. Isso demonstra que o armazenamento do cluster do Aurora sempre usa três AZs.

```
us-east-1c
us-east-1d
us-east-1e
```

Para criar um clone em um ambiente de rede que não tenha todos os recursos disponíveis para estabelecer conexão com essas AZs, é necessário criar sub-redes associadas a pelo menos duas dessas AZs e, depois, criar um grupo de sub-redes de banco de dados contendo essas duas ou três sub-redes. Os exemplos a seguir mostram o procedimento.

Etapa 2: verificar o grupo de sub-redes de banco de dados do cluster original

Se você quiser usar o mesmo número de sub-redes do cluster original para o clone, identifique o número de sub-redes do grupo de sub-redes de banco de dados do cluster original. Um grupo de sub-redes de banco de dados do Aurora contém pelo menos duas sub-redes, cada uma associada a uma AZ diferente. Anote a quais AZs as sub-redes estão associadas.

O exemplo a seguir mostra como encontrar o grupo de sub-redes de banco de dados do cluster original e, depois, voltar para as AZs correspondentes. Substitua o nome do cluster por *my_cluster* no primeiro comando. Substitua o nome do grupo de sub-redes de banco de dados por *my_subnet* no segundo comando.

```
aws rds describe-db-clusters --db-cluster-identifier my_cluster \
  --query '*[].DBSubnetGroup' --output text

aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetAvailabilityZone.Name]' --output text
```

O exemplo de saída para um cluster com um grupo de sub-redes de banco de dados contendo duas sub-redes pode ser semelhante ao indicado a seguir. Nesse caso, `two-subnets` é um nome que foi especificado ao criar o grupo de sub-redes de banco de dados.

```
two-subnets

us-east-1d
us-east-1c
```

Com relação a um cluster em que o grupo de sub-redes de banco de dados contém três sub-redes, a saída pode ser semelhante à indicada a seguir.

```
three-subnets

us-east-1f
us-east-1d
us-east-1c
```

Etapa 3: verificar as sub-redes do cluster original

Se você precisar de mais detalhes sobre as sub-redes no cluster original, execute comandos da AWS CLI semelhantes aos indicados a seguir. É possível examinar os atributos da sub-rede, como

intervalos de endereços IP, proprietário e assim por diante. Dessa forma, você pode determinar se deve usar sub-redes diferentes na mesma VPC ou criar sub-redes com características semelhantes em uma VPC diferente.

Encontre os IDs de todas as sub-redes que estão disponíveis na VPC.

```
aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc \
  --query '*[].[SubnetId]' --output text
```

Encontre as sub-redes exatas usadas no grupo de sub-redes de banco de dados.

```
aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetIdentifier]' --output text
```

Depois, especifique as sub-redes que você deseja investigar em uma lista, como no comando a seguir. Substitua os nomes das sub-redes por *my_subnet_1* e assim por diante.

```
aws ec2 describe-subnets \
  --subnet-ids '["my_subnet_1","my_subnet2","my_subnet3"]'
```

O exemplo a seguir mostra a saída parcial desse comando `describe-subnets`. A saída mostra alguns dos atributos importantes que você pode ver em cada sub-rede, como a AZ associada e a VPC da qual ela faz parte.

```
{
    'Subnets': [
        {
            'AvailabilityZone': 'us-east-1d',
            'AvailableIpAddressCount': 54,
            'CidrBlock': '10.0.0.64/26',
            'State': 'available',
            'SubnetId': 'subnet-000a0bca00e0b0000',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
        },
        {
            'AvailabilityZone': 'us-east-1c',
            'AvailableIpAddressCount': 55,
            'CidrBlock': '10.0.0.0/26',
            'State': 'available',
            'SubnetId': 'subnet-4b4dbfe4d4a4fd4c4',
```

```
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
```

Etapa 4: verificar as zonas de disponibilidade das instâncias de banco de dados no cluster original

É possível usar esse procedimento para entender as AZs usadas com as instâncias de banco de dados no cluster original. Dessa forma, você pode configurar exatamente as mesmas AZs para as instâncias de banco de dados no clone. Também é possível usar mais ou menos instâncias de banco de dados no clone, dependendo se ele é usado para produção, desenvolvimento e testes etc.

Para cada instância no cluster original, execute um comando como o indicado a seguir. Primeiro verifique se a criação da instância já foi concluída e ela está no estado `Available`. Substitua o identificador da instância por *`my_instance`*.

```
aws rds describe-db-instances --db-instance-identifier my_instance \
  --query '*[].AvailabilityZone' --output text
```

O exemplo a seguir mostra a saída da execução do comando `describe-db-instances` anterior. O cluster do Aurora tem quatro instâncias de banco de dados. Portanto, executamos o comando quatro vezes, substituindo um identificador de instância de banco de dados diferente a cada vez. O resultado mostra como essas instâncias de banco de dados estão distribuídas em no máximo três AZs.

```
us-east-1a
us-east-1c
us-east-1d
us-east-1a
```

Depois que o clone é criado e você adiciona instâncias de banco de dados a ele, é possível especificar esses mesmos nomes de AZ nos comandos `create-db-instance`. Isso pode ser feito para configurar instâncias de banco de dados no novo cluster configurado exatamente para as mesmas AZs do cluster original.

Etapa 5: verificar as VPCs que você pode usar para o clone

Se pretende criar o clone em uma VPC diferente da original, é possível obter uma lista de IDs de VPC disponíveis para a conta. Você também pode executar essa etapa se precisar criar outras sub-redes na mesma VPC do cluster original. Ao executar o comando para criar uma sub-rede, você especifica o ID da VPC como parâmetro.

Para listar todas as VPCs da sua conta, execute este comando da CLI:

```
aws ec2 describe-vpcs --query '*[].[VpcId]' --output text
```

O exemplo a seguir mostra a saída de amostra do comando `describe-vpcs` anterior. O resultado demonstra que há quatro VPCs na conta da AWS atual que podem ser usadas como origem ou destino para clonagem entre VPCs.

```
vpc-fd111111
vpc-2222e2cd2a222f22e
vpc-33333333a33333d33
vpc-4ae4d4de4a4444dad
```

É possível usar a mesma VPC como destino para o clone ou uma VPC diferente. Se o cluster original e o clone estiverem na mesma VPC, você poderá reutilizar o mesmo grupo de sub-redes de banco de dados para o clone. Também é possível criar um grupo de sub-redes de banco de dados. Por exemplo, o novo grupo de sub-redes de banco de dados pode usar sub-redes privadas, enquanto o grupo de sub-redes de banco de dados do cluster original pode usar sub-redes públicas. Se você criar o clone em uma VPC diferente, verifique se há sub-redes suficientes na nova VPC e se elas estão associadas às AZs corretas do cluster original.

Criar recursos de rede para o clone

Se, ao coletar as informações da rede, você descobrir que são necessários recursos de rede adicionais para o clone, crie-os antes de tentar configurar o clone. Por exemplo, talvez seja necessário criar mais sub-redes, sub-redes associadas a AZs específicas ou um grupo de sub-redes de banco de dados.

- Etapa 1: criar sub-redes para o clone
- Etapa 2: criar o grupo de sub-redes de banco de dados para o clone

Etapa 1: criar sub-redes para o clone

Se você precisar criar sub-redes para o clone, execute um comando semelhante ao apresentado a seguir. Talvez seja necessário fazer isso ao criar o clone em uma VPC diferente ou ao fazer alguma outra alteração na rede, como usar sub-redes privadas em vez de públicas.

A AWS gera automaticamente o ID da sub-rede. Substitua o nome da VPC do clone por *`my_vpc`*. Escolha o intervalo de endereços para a opção `--cidr-block` a fim de permitir pelo menos 16

endereços IP no intervalo. É possível incluir qualquer outra propriedade que você queira especificar. Execute o comando `aws ec2 create-subnet help` para ver todas as opções.

```
aws ec2 create-subnet --vpc-id my_vpc \
  --availability-zone AZ_name --cidr-block IP_range
```

O exemplo a seguir mostra alguns atributos importantes de uma sub-rede recém-criada.

```
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1b',
        'AvailableIpAddressCount': 59,
        'CidrBlock': '10.0.0.64/26',
        'State': 'available',
        'SubnetId': 'subnet-44b4a44f4e44db444',
        'VpcId': 'vpc-555fc5df555e555dc',
        ...
        }
}
```

Etapa 2: criar o grupo de sub-redes de banco de dados para o clone

Se você estiver criando o clone em uma VPC diferente ou em um conjunto distinto de sub-redes na mesma VPC, crie um grupo de sub-redes de banco de dados e o especifique ao criar o clone.

Verifique se você conhece todos os detalhes a seguir. É possível encontrar tudo isso na saída dos exemplos anteriores.

1. VPC do cluster original. Para obter instruções, consulte Etapa 3: verificar as sub-redes do cluster original.
2. VPC do clone, se ele estiver sendo criado em uma VPC diferente. Para obter instruções, consulte Etapa 5: verificar as VPCs que você pode usar para o clone.
3. Três AZs associadas ao armazenamento do Aurora para o cluster original. Para obter instruções, consulte Etapa 1: verificar as zonas de disponibilidade do cluster original.
4. Duas ou três AZs associadas ao grupo de sub-redes de banco de dados do cluster original. Para obter instruções, consulte Etapa 2: verificar o grupo de sub-redes de banco de dados do cluster original.
5. Os IDs de sub-rede e as AZs associadas de todas as sub-redes na VPC que você pretende usar para o clone. Use o mesmo comando `describe-subnets` que em Etapa 3: verificar as sub-redes do cluster original, substituindo o ID da VPC de destino.

Verifique quantas AZs estão associadas ao armazenamento do cluster original e às sub-redes na VPC de destino. Para conseguir criar o clone, deve haver duas ou três AZs em comum. Se você tiver menos de duas AZs em comum, volte para Etapa 1: criar sub-redes para o clone. Crie uma, duas ou três sub-redes associadas às AZs correspondentes ao armazenamento do cluster original.

Escolha sub-redes na VPC de destino que estejam associadas às mesmas AZs do armazenamento do Aurora no cluster original. O ideal é escolher três AZs. Essa é opção que oferece o mais alto nível de flexibilidade para distribuir as instâncias de banco de dados do clone em várias AZs e ter alta disponibilidade de recursos de computação.

Execute um comando semelhante ao indicado a seguir para criar o grupo de sub-redes de banco de dados. Substitua os IDs das sub-redes na lista. Se você especificar os IDs de sub-rede usando variáveis de ambiente, tenha o cuidado de citar a lista de parâmetros `--subnet-ids` de uma forma que mantenha as aspas duplas ao redor dos IDs.

```
aws rds create-db-subnet-group --db-subnet-group-name my_subnet_group \
  --subnet-ids '["my_subnet_1","my_subnet_2","my_subnet3"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets for clone'
```

O exemplo a seguir mostra a saída parcial do comando `create-db-subnet-group`.

```
{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'my_subnet_group',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets for clone',
        'VpcId': 'vpc-555fc5df555e555dc',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'my_subnet_1',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1c'
                },
                'SubnetStatus': 'Active'
            },
            {
                'SubnetIdentifier': 'my_subnet_2',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1d'
                },
                'SubnetStatus': 'Active'
```

```
            }
            ...
        ],
        'SupportedNetworkTypes': [
            'IPV4'
        ]
    }
}
```

Nesse momento, o clone ainda não foi criado. Você criou todos os recursos relevantes de VPC e sub-rede para poder especificar os parâmetros apropriados para os comandos `restore-db-cluster-to-point-in-time` e `create-db-instance` ao criar o clone.

Criar um clone do Aurora com novas configurações de rede

Depois de verificar se a configuração correta de VPCs, sub-redes, AZs e grupos de sub-redes está disponível para o novo cluster usar, você pode realizar a operação de clonagem. Os exemplos de CLI a seguir destacam as opções que você especifica nos comandos, como `--db-subnet-group-name`, `--availability-zone` e `--vpc-security-group-ids`, para configurar o clone e as respectivas instâncias de banco de dados.

- Etapa 1: especificar o grupo de sub-redes de banco de dados para o clone
- Etapa 2: especificar as configurações de rede para instâncias no clone
- Etapa 3: estabelecer a conectividade de um sistema cliente com um clone

Etapa 1: especificar o grupo de sub-redes de banco de dados para o clone

Ao criar o clone, é possível definir todas as configurações corretas de VPC, sub-rede e AZ especificando um grupo de sub-redes de banco de dados. Use os comandos nos exemplos anteriores para verificar todas as relações e associações que entram no grupo de sub-redes do banco de dados.

Por exemplo, os comandos a seguir demonstram a clonagem de um cluster original em um clone. No primeiro exemplo, o cluster de origem está associado a duas sub-redes e o clone a três. O segundo exemplo mostra o caso oposto, a clonagem de um cluster com três sub-redes para um cluster com duas.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-3-subnets \
```

```
  --db-cluster-identifier cluster-cloned-to-2-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name two-subnets
```

Se você pretende usar instâncias do Aurora Sem Servidor v2 no clone, inclua uma opção `--serverless-v2-scaling-configuration` ao criar o clone, conforme mostrado. Isso permite que você use a classe `db.serverless` ao criar instâncias de banco de dados no clone. Você também pode modificar o clone posteriormente para adicionar esse atributo de configuração de escalabilidade. Os números de capacidade nesse exemplo permitem que cada instância do Aurora Sem Servidor v2 no cluster escale entre 2 e 32 unidades de capacidade do Aurora (ACUs). Consulte informações sobre o recurso Aurora Sem Servidor v2 e como escolher o intervalo de capacidade em Usar o Aurora Serverless v2.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-2-subnets \
  --db-cluster-identifier cluster-cloned-to-3-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name three-subnets \
  --serverless-v2-scaling-configuration 'MinCapacity=2,MaxCapacity=32'
```

Independentemente do número de sub-redes usadas pelas instâncias de banco de dados, o armazenamento do Aurora para o cluster de origem e o clone está associado a três AZs. O exemplo a seguir lista as AZs associadas ao cluster original e ao clone, para ambos os comandos `restore-db-cluster-to-point-in-time` nos exemplos anteriores.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-with-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-with-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text
```

```
us-east-1a
us-east-1c
us-east-1d

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d
```

Como pelo menos duas das AZs se sobrepõem entre cada par de clusters originais e clones, os dois clusters podem acessar o mesmo armazenamento subjacente do Aurora.

Etapa 2: especificar as configurações de rede para instâncias no clone

Quando você cria instâncias de banco de dados no clone, elas herdam o grupo de sub-redes de banco de dados do próprio cluster por padrão. Dessa forma, o Aurora atribui automaticamente cada instância a uma sub-rede específica e a cria na AZ associada à sub-rede. Essa opção é conveniente, especialmente para sistemas de desenvolvimento e teste, porque você não precisa monitorar os IDs de sub-rede ou as AZs ao adicionar novas instâncias ao clone.

Também é possível especificar a AZ ao criar uma instância de banco de dados Aurora para o clone. A AZ que você especifica deve ser do conjunto de AZs associadas ao clone. Se o grupo de sub-redes de banco de dados que você usa para o clone tiver apenas duas sub-redes, só será possível escolher entre as AZs associadas a essas duas sub-redes. Essa opção oferece flexibilidade e resiliência para sistemas altamente disponíveis, porque é possível garantir que a instância de gravador e a instância de leitor em espera estejam em AZs diferentes. Ou, se você adicionar mais leitores ao cluster, poderá garantir que eles estejam distribuídos em três AZs. Dessa forma, mesmo no caso raro de uma falha de AZ, você ainda tem uma instância de gravador e outra instância de leitor em duas outras AZs.

O exemplo a seguir adiciona uma instância de banco de dados provisionada a um cluster clonado do Aurora PostgreSQL que usa um grupo de sub-redes de banco de dados personalizado.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_postgresql_clone \
  --db-instance-identifier my_postgres_instance \
  --db-subnet-group-name my_new_subnet \
  --engine aurora-postgresql \
```

```
  --db-instance-class db.t4g.medium
```

O exemplo a seguir mostra a saída parcial desse comando.

```
{
  'DBInstanceIdentifier': 'my_postgres_instance',
  'DBClusterIdentifier': 'my_aurora_postgresql_clone',
  'DBInstanceClass': 'db.t4g.medium',
  'DBInstanceStatus': 'creating'
  ...
}
```

O exemplo a seguir adiciona uma instância de banco de dados do Aurora Sem Servidor v2 a um clone do Aurora MySQL que usa um grupo de sub-redes de banco de dados personalizado. Para poder usar instâncias do Aurora Sem Servidor v2, especifique a opção `--serverless-v2-scaling-configuration` para o comando `restore-db-cluster-to-point-in-time`, conforme mostrado nos exemplos anteriores.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_mysql_clone \
  --db-instance-identifier my_mysql_instance \
  --db-subnet-group-name my_other_new_subnet \
  --engine aurora-mysql \
  --db-instance-class db.serverless
```

O exemplo a seguir mostra a saída parcial desse comando.

```
{
  'DBInstanceIdentifier': 'my_mysql_instance',
  'DBClusterIdentifier': 'my_aurora_mysql_clone',
  'DBInstanceClass': 'db.serverless',
  'DBInstanceStatus': 'creating'
  ...
}
```

Etapa 3: estabelecer a conectividade de um sistema cliente com um clone

Se você já estiver se conectando a um cluster do Aurora por um sistema cliente, talvez queira permitir o mesmo tipo de conectividade a um novo clone. Por exemplo, você pode se conectar ao cluster original por uma instância do Amazon Cloud9 ou do EC2. Para permitir conexões dos mesmos sistemas cliente ou de novos sistemas que você cria na VPC de destino, configure grupos

de sub-rede de banco de dados e grupos de segurança de VPC equivalentes aos da VPC. Depois, especifique o grupo de sub-redes e grupos de segurança ao criar o clone.

Os exemplos a seguir configuram um clone do Aurora Sem Servidor v2. Essa configuração é baseada na combinação de `--engine-mode provisioned` e `--serverless-v2-scaling-configuration` ao criar o cluster de banco de dados e de `--db-instance-class db.serverless` ao criar cada instância de banco de dados no cluster. O modo `provisioned` do mecanismo é o padrão, então você poderá omitir essa opção se assim preferir.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier serverless-sql-postgres\
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-subnet-group-name 'default-vpc-1234' \
  --vpc-security-group-ids 'sg-4567' \
  --serverless-v2-scaling-configuration 'MinCapacity=0.5,MaxCapacity=16' \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Depois, ao criar as instâncias de banco de dados no clone, especifique a mesma opção `--db-subnet-group-name`. Se preferir, você poderá incluir a opção `--availability-zone` e especificar uma das AZs associadas às sub-redes nesse grupo de sub-redes. Essa AZ também deve ser uma das AZs associadas ao cluster original.

```
aws rds create-db-instance \
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-instance-identifier serverless-sql-postgres-clone-instance \
  --db-instance-class db.serverless \
  --db-subnet-group-name 'default-vpc-987zyx654' \
  --availability-zone 'us-east-1c' \
  --engine aurora-postgresql
```

Mover um cluster de sub-redes públicas para privadas

É possível usar a clonagem para migrar um cluster entre sub-redes públicas e privadas. Você pode fazer isso ao adicionar outras camadas de segurança à aplicação antes de implantá-la na produção. Nesse exemplo, antes de iniciar o processo de clonagem com o Aurora, você já deve ter as sub-redes privadas e o gateway NAT configurados.

Com relação às etapas que envolvem o Aurora, siga as mesmas etapas gerais dos exemplos anteriores para Reunir informações sobre o ambiente de rede e Criar um clone do Aurora com novas

configurações de rede. A principal diferença é que, mesmo que você tenha sub-redes públicas associadas a todas as AZs do cluster original, agora é necessário verificar se há sub-redes privadas suficientes para um cluster do Aurora e se elas estão associadas às mesmas AZs usadas para o armazenamento do Aurora no cluster original. De modo semelhante a outros casos de uso de clonagem, é possível criar o grupo de sub-redes de banco de dados para o clone com três ou duas sub-redes privadas associadas às AZs necessárias. No entanto, se você usar duas sub-redes privadas no grupo de sub-redes de banco de dados, deverá ter uma terceira sub-rede privada associada à terceira AZ usada para armazenamento do Aurora no cluster original.

É possível consultar nessa lista de verificação se todos os requisitos estão em vigor para realizar esse tipo de operação de clonagem.

- Registre as três AZs associadas ao cluster original. Para obter instruções, consulte Etapa 1: verificar as zonas de disponibilidade do cluster original.
- Registre as três ou duas AZs associadas às sub-redes públicas no grupo de sub-redes de banco de dados do cluster original. Para obter instruções, consulte Etapa 3: verificar as sub-redes do cluster original.
- Crie sub-redes privadas que correspondam às três AZs associadas ao cluster original. Também faça qualquer outra configuração de rede, como criar um gateway NAT, para estabelecer comunicação com as sub-redes privadas. Consulte instruções em Criar uma sub-rede no Guia do usuário da Amazon Virtual Private Cloud.
- Crie um grupo de sub-redes de banco de dados contendo três ou duas das sub-redes privadas associadas às AZs desde o primeiro ponto. Para obter instruções, consulte Etapa 2: criar o grupo de sub-redes de banco de dados para o clone.

Quando todos os pré-requisitos forem cumpridos, será possível pausar a atividade do banco de dados no cluster original enquanto você cria o clone e troca a aplicação para usá-lo. Depois que o clone for criado e você confirmar que consegue se conectar a ele, executar o código da aplicação e assim por diante, será possível interromper o uso do cluster original.

Exemplo completo da criação de um clone entre VPCs

A criação de um clone em uma VPC diferente da original usa as mesmas etapas gerais dos exemplos anteriores. Como o ID da VPC é uma propriedade das sub-redes, você não especifica o ID da VPC como um parâmetro ao executar qualquer um dos comandos da CLI do RDS. A principal diferença é que é mais provável que você precise criar sub-redes, sub-redes associadas a AZs específicas, um grupo de segurança de VPC e um grupo de sub-redes de banco de dados. Isso se aplica principalmente se esse for o primeiro cluster do Aurora que está sendo criado nessa VPC.

É possível consultar nessa lista de verificação se todos os requisitos estão em vigor para realizar esse tipo de operação de clonagem.

- Registre as três AZs associadas ao cluster original. Para obter instruções, consulte Etapa 1: verificar as zonas de disponibilidade do cluster original.
- Registre as três ou duas AZs associadas às sub-redes no grupo de sub-redes de banco de dados do cluster original. Para obter instruções, consulte Etapa 2: verificar o grupo de sub-redes de banco de dados do cluster original.
- Crie sub-redes que correspondam às três AZs associadas ao cluster original. Para obter instruções, consulte Etapa 1: criar sub-redes para o clone.
- Faça qualquer outra configuração de rede, como configurar um grupo de segurança da VPC, para sistemas cliente, servidores de aplicações e assim por diante para estabelecer comunicação com as instâncias de banco de dados no clone. Para obter instruções, consulte Controlar acesso com grupos de segurança.
- Crie um grupo de sub-redes de banco de dados contendo três ou duas das sub-redes associadas às AZs desde o primeiro ponto. Para obter instruções, consulte Etapa 2: criar o grupo de sub-redes de banco de dados para o clone.

Quando todos os pré-requisitos forem cumpridos, será possível pausar a atividade do banco de dados no cluster original enquanto você cria o clone e troca a aplicação para usá-lo. Depois que o clone for criado e você confirmar que consegue se conectar a ele, executar o código da aplicação e assim por diante, considere se deseja manter o original e os clones em execução ou interromper o uso do cluster original.

Os exemplos do Linux a seguir mostram a sequência de operações da AWS CLI para clonar um cluster de banco de dados do Aurora de uma VPC para outra. Alguns campos que não são relevantes para os exemplos não são mostrados na saída do comando.

Primeiro, verificamos os IDs das VPCs de origem e destino. O nome descritivo que você atribui a uma VPC ao criá-la é representado como uma tag nos metadados da VPC.

```
$ aws ec2 describe-vpcs --query '*[].[VpcId,Tags]'
[
    [
        'vpc-0f0c0fc0000b0ffb0',
        [
            {
                'Key': 'Name',
```

```
                'Value': 'clone-vpc-source'
            }
        ]
    ],
    [
        'vpc-9e99d9f99a999bd99',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-dest'
            }
        ]
    ]
]
```

O cluster original já existe na VPC de origem. Para configurar o clone usando o mesmo conjunto de AZs para o armazenamento do Aurora, selecionamos as AZs usadas pelo cluster original.

```
$ aws rds describe-db-clusters --db-cluster-identifier original-cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Garantimos que haja sub-redes que correspondam às AZs usadas pelo cluster original: `us-east-1c`, `us-east-1d` e `us-east-1f`.

```
$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
  --availability-zone us-east-1c --cidr-block 10.0.0.128/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1c',
        'SubnetId': 'subnet-3333a33be3ef3e333',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1d --cidr-block 10.0.0.160/28
{
    'Subnet': {
```

```
        'AvailabilityZone': 'us-east-1d',
        'SubnetId': 'subnet-4eeb444cd44b4d444',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1f --cidr-block 10.0.0.224/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1f',
        'SubnetId': 'subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}
```

Este exemplo confirma que há sub-redes que correspondem às AZs necessárias na VPC de destino.

```
aws ec2 describe-subnets --query 'sort_by(*[] | [?VpcId == `vpc-9e99d9f99a999bd99`] |
[].{SubnetId:SubnetId,VpcId:VpcId,AvailabilityZone:AvailabilityZone},
 &AvailabilityZone)' --output table

-------------------------------------------------------------------------
|                            DescribeSubnets                            |
+------------------+---------------------------+-------------------------+
| AvailabilityZone |         SubnetId          |          VpcId          |
+------------------+---------------------------+-------------------------+
|  us-east-1a      |  subnet-000ff0e00000c0aea  |  vpc-9e99d9f99a999bd99  |
|  us-east-1b      |  subnet-1111d111111ca11b1  |  vpc-9e99d9f99a999bd99  |
|  us-east-1c      | subnet-3333a33be3ef3e333   |  vpc-9e99d9f99a999bd99  |
|  us-east-1d      | subnet-4eeb444cd44b4d444   |  vpc-9e99d9f99a999bd99  |
|  us-east-1f      | subnet-66eea6666fb66d66c   |  vpc-9e99d9f99a999bd99  |
+------------------+---------------------------+-------------------------+
```

Antes de criar um cluster de banco de dados do Aurora na VPC, você deve ter um grupo de sub-redes de banco de dados com sub-redes que correspondam às AZs usadas para armazenamento do Aurora. Ao criar um cluster normal, você pode usar qualquer conjunto de três AZs. Quando você clona um cluster existente, o grupo de sub-redes deve corresponder a pelo menos duas das três AZs que ele usa para armazenamento do Aurora.

```
$ aws rds create-db-subnet-group \
  --db-subnet-group-name subnet-group-in-other-vpc \
```

```
  --subnet-ids
 '["subnet-3333a33be3ef3e333","subnet-4eeb444cd44b4d444","subnet-66eea6666fb66d66c"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c'

{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'subnet-group-in-other-vpc',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'subnet-4eeb444cd44b4d444',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1d' }
            },
            {
                'SubnetIdentifier': 'subnet-3333a33be3ef3e333',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1c' }
            },
            {
                'SubnetIdentifier': 'subnet-66eea6666fb66d66c',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1f' }
            }
        ]
    }
}
```

Agora as sub-redes e o grupo de sub-redes de banco de dados estão prontos. O exemplo a seguir mostra o `restore-db-cluster-to-point-in-time` que clona o cluster. A opção `--db-subnet-group-name` associa o clone ao conjunto correto de sub-redes que correspondem ao conjunto correto de AZs do cluster original.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier original-cluster \
  --db-cluster-identifier clone-in-other-vpc \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name subnet-group-in-other-vpc

{
  'DBClusterIdentifier': 'clone-in-other-vpc',
  'DBSubnetGroup': 'subnet-group-in-other-vpc',
```

```
  'Engine': 'aurora-postgresql',
  'EngineVersion': '15.4',
  'Status': 'creating',
  'Endpoint': 'clone-in-other-vpc.cluster-c0abcdef.us-east-1.rds.amazonaws.com'
}
```

O exemplo a seguir confirma que o armazenamento do Aurora no clone usa o mesmo conjunto de AZs do cluster original.

```
$ aws rds describe-db-clusters --db-cluster-identifier clone-in-other-vpc \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Neste ponto, você poderá criar instâncias de banco de dados para o clone. O grupo de segurança da VPC associado a cada instância deve permitir conexões dos intervalos de endereços IP que você usa para as instâncias do EC2, os servidores de aplicações e assim por diante que estão na VPC de destino.

Clonar entre contas com o AWS RAM e Amazon Aurora

Ao usar o AWS Resource Access Manager (AWS RAM) com o Amazon Aurora, você pode compartilhar clusters e clones de banco de dados do Aurora que pertencem a sua conta da AWS com outra conta da AWS ou organização. Essa clonagem entre contas do que a criação e a restauração de um snapshot de banco de dados. Você pode criar um clone a partir de um de seus clusters de banco de dados do Aurora e compartilhar o clone. Ou você pode compartilhar seu cluster de banco de dados do Aurora com outra conta da AWS e deixar o titular da conta criar o clone. A abordagem escolhida depende de seu caso de uso.

Por exemplo, talvez seja necessário compartilhar regularmente um clone do banco de dados financeiro com a equipe de auditoria interna de sua organização. Nesse caso, sua equipe de auditoria tem o própria conta da AWS para as aplicações que ela usa. Você pode dar permissão à conta da AWS da equipe de auditoria para acessar seu cluster de banco de dados do Aurora e cloná-lo conforme necessário.

Por outro lado, se um fornecedor externo auditar seus dados financeiros, talvez você mesmo prefira criar o clone. Em seguida, conceda acesso somente ao clone para o fornecedor externo.

Você também pode usar a clonagem entre contas para oferecer suporte a muitos dos mesmos casos de uso para clonagem dentro da mesma conta da AWS, como desenvolvimento e teste. Por exemplo, sua organização pode usar diferentes contas da AWSpara produção, desenvolvimento, teste etc. Para obter mais informações, consulte Visão geral da clonagem do Aurora.

Assim, recomenda-se compartilhar um clone com outra conta da AWSou permitir que outra conta da AWS crie clones de seus clusters de banco de dados do Aurora. Em ambos os casos, comece usando o AWS RAM para criar um objeto de compartilhamento. Para obter informações completas sobre o compartilhamento de recursos da AWS entre contas da AWS, consulte o Manual do usuário do AWS RAM.

A criação de um clone entre contas requer ações na conta da AWS que tem o cluster original e na conta da AWS que cria o clone. Primeiro, o proprietário do cluster original modifica o cluster para permitir que uma ou mais outras contas o clonem. Se qualquer uma das contas estiver em uma organização da AWSdiferente, a AWS gerará um convite de compartilhamento. A outra conta deve aceitar o convite antes de prosseguir. Depois, cada conta autorizada pode clonar o cluster. Durante todo esse processo, o cluster é identificado por seu nome de recurso da Amazon (ARN) exclusivo.

Como acontece com a clonagem dentro da mesma conta da AWS, espaço de armazenamento adicional será usado somente se as alterações forem feitas nos dados pelo original ou pelo clone. Então as cobranças de armazenamento serão aplicadas nesse momento. Se o cluster de origem for excluído, os custos de armazenamento serão igualmente distribuídos entre os clusters clonados restantes.

Tópicos

- Limitações da clonagem entre contas
- Permissão para que outras contas da AWS clonem seu cluster
- Clonar um cluster que pertence a outra conta da AWS

Limitações da clonagem entre contas

A clonagem entre contas do Aurora tem as seguintes limitações:

- Não é possível clonar um cluster Aurora Serverless v1 entre contas da AWS.
- Não é possível visualizar nem aceitar convites para recursos compartilhados com o Console de gerenciamento da AWS. Use a AWS CLI, a API do Amazon RDS ou o console do AWS RAM para exibir e aceitar convites para recursos compartilhados.

- É possível criar apenas um clone de um recurso compartilhado com sua Conta da AWS. Isso se aplica independentemente de o recurso compartilhado ser um cluster de banco de dados original do Aurora ou um clone criado anteriormente.
- É possível criar um clone apenas de um clone que tenha sido compartilhado com sua conta da AWS.
- Você não pode compartilhar recursos (clones ou clusters de banco de dados do Aurora) que foram compartilhados com sua conta da AWS.
- Não é possível criar mais de 15 clones entre contas em qualquer cluster de banco de dados do Aurora.
- Cada um desses 15 clones deve pertencer a uma conta da AWS. Ou seja, é possível criar apenas um clone entre contas de um cluster em qualquer conta da AWS.
- Depois de clonar um cluster, o cluster original e o respectivo clone são considerados os mesmos a fim de impor limites em clones entre contas. Você não pode criar clones entre contas do cluster original e do cluster clonado dentro da mesma conta da AWS. O número total de clones entre contas para o cluster original e qualquer um de seus clones não pode ser superior a 15.
- Não é possível compartilhar um cluster de banco de dados do Aurora com outras contas da AWS, a menos que o cluster esteja em estado `ACTIVE`.
- Não é possível renomear um cluster de banco de dados do Aurora que tenha sido compartilhado com outras contas da AWS.
- Não é possível criar um clone entre contas de um cluster que é criptografado com a chave padrão do RDS.
- Não é possível criar clones não criptografados em uma conta da AWS em clusters de banco de dados do Aurora criptografados compartilhados por outra conta da AWS. O proprietário do cluster deve conceder permissão de acesso à do cluster de fonte AWS KMS key. No entanto, é possível usar uma chave diferente ao criar o clone.

Permissão para que outras contas da AWS clonem seu cluster

Para permitir que outras contas da AWS clonem um cluster de sua propriedade, use o AWS RAM para definir a permissão de compartilhamento. Fazer isso também envia um convite para cada conta que está em uma organização da AWS diferente.

Para ver os procedimentos necessários para compartilhar recursos de sua propriedade no console do AWS RAM, consulte Sharing resources owned by you (Compartilhar recursos de sua propriedade) no Manual do usuário do AWS RAM.

Tópicos

- Concessão de permissão para que outras contas da AWS clonem seu cluster
- Verificar se um cluster de sua propriedade está compartilhado com outras contas da AWS

Concessão de permissão para que outras contas da AWS clonem seu cluster

Se o cluster que você está compartilhando estiver criptografado, também será necessário compartilhar a AWS KMS key do cluster. Você pode permitir que os usuários ou as funções do AWS Identity and Access Management (IAM) de uma conta da AWS usem uma chave do KMS em conta distinta.

Para fazê-lo, primeiro adicione a conta externa (usuário raiz) à política de chave do KMS por meio do AWS KMS. Você não adiciona os usuários ou perfis individuais à política de chaves, somente a conta externa que os possui. Você só pode compartilhar a chave do KMS que criar, não a chave de serviço padrão do RDS. Para obter informações sobre o controle de acesso para chaves do KMS, consulte Autenticação e controle de acesso para o AWS KMS.

Console

Para conceder permissão para clonar o cluster

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados que deseja compartilhar para ver a página Details (Detalhes) e escolha a guia Connectivity & security (Conectividade e segurança).
4. Na seção Compartilhar cluster de banco de dados com outras contas da AWS, insira o ID numérico da conta da AWS que deseja autorizar para clonar esse cluster. Para IDs de conta na mesma organização, você pode começar digitando na caixa e escolher a opção no menu.

> **Important**
>
> Em alguns casos, convém não ter uma conta na mesma organização da AWS que sua conta para clonar um cluster. Nesses casos, por motivos de segurança, o console não relata que possui esse ID de conta ou se a conta existe.

> Tome cuidado ao inserir números de conta que não estão na mesma organização da AWS que sua conta da AWS. Verifique imediatamente se você compartilhou com a conta desejada.

5. Na página de confirmação, verifique se o ID da conta especificado está correto. Insira `share` na caixa de confirmação para confirmar.

   Na página Details (Detalhes), uma entrada exibe o ID da conta da AWS especificado em Accounts that this DB cluster is shared with (Contas com as quais esse cluster de banco de dados está compartilhado). A coluna Status mostra inicialmente o status Pending (Pendente).

6. Entre em contato com o proprietário da outra conta da AWS ou faça login nessa conta, se for proprietário das duas. Instrua o proprietário da outra conta a aceitar o convite de compartilhamento e clonar o cluster de banco de dados, conforme descrito a seguir.

AWS CLI

Para conceder permissão para clonar o cluster

1. Colete as informações para os parâmetros obrigatórios. Você precisa do ARN do cluster e do ID numérico da outra conta da AWS.
2. Execute o comando da CLI do AWS RAM `create-resource-share`.

   Para Linux, macOS ou Unix:

```
aws ram create-resource-share --name descriptive_name \
  --region region \
  --resource-arns cluster_arn \
  --principals other_account_ids
```

   Para Windows:

```
aws ram create-resource-share --name descriptive_name ^
  --region region ^
  --resource-arns cluster_arn ^
  --principals other_account_ids
```

   Para incluir vários IDs de conta para o parâmetro `--principals`, separe os IDs de cada uma com espaços. Para especificar se os IDs de conta permitidos podem estar fora da sua

organização da AWS, inclua o parâmetro `--allow-external-principals` ou `--no-allow-external-principals` para `create-resource-share`.

AWS RAMAPI do

Para conceder permissão para clonar o cluster

1. Colete as informações para os parâmetros obrigatórios. Você precisa do ARN do cluster e do ID numérico da outra conta da AWS.
2. Chame a operação de API do AWS RAM CreateResourceShare e especifique os seguintes valores:
   - Especifique o ID de uma ou mais contas da AWS como o parâmetro `principals`.
   - Especifique o ARN de um ou mais clusters de banco de dados do Aurora como o parâmetro `resourceArns`.
   - Especifique se os IDs de conta permitidos podem estar fora de sua organização da AWS, incluindo um valor booliano para o parâmetro `allowExternalPrincipals`.

Recriar um cluster que usa a chave padrão do RDS

Se o cluster criptografado que você planeja compartilhar usar a chave padrão do RDS, recrie o cluster. Para isso, crie um snapshot manual do seu cluster de banco de dados, use uma AWS KMS key e restaure o cluster como um novo. Então, compartilhe o novo cluster. Para realizar esse processo, execute as seguintes etapas:

Para recriar um cluster criptografado que usa a chave padrão do RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Snapshots.
3. Escolha seu snapshot.
4. Em Actions (Ações), escolha Copy Snapshot (Copiar snapshot) e Enable encryption (Ativar criptografia).
5. Em AWS KMS key, escolha a nova chave de criptografia que deseja usar.

6. Restaure o snapshot copiado. Para fazer isso, siga o procedimento em Restauração de um snapshot de um cluster de banco de dados. A nova instância de banco de dados usará a nova chave de criptografia.
7. (Opcional) Exclua o cluster de banco de dados antigo se não precisar mais dele. Para fazer isso, siga o procedimento em Excluir um snapshot de cluster de banco de dados. Antes de fazer isso, confirme se o novo cluster tem todos os dados necessários e se seu aplicativo consegue acessá-lo.

Verificar se um cluster de sua propriedade está compartilhado com outras contas da AWS

Você pode verificar se outros usuários têm permissão para compartilhar um cluster. Fazer isso pode ajudá-lo a entender se o cluster está se aproximando do limite de número máximo de clones entre contas.

Para ver os procedimentos necessários para compartilhar recursos usando o console do AWS RAM, consulte Sharing resources owned by you (Compartilhar recursos de sua propriedade) no Manual do usuário do AWS RAM.

AWS CLI

Para descobrir se um cluster seu está compartilhado com outras contas da AWS

- Execute o comando da CLI do AWS RAM `list-principals` usando o ID da conta como o proprietário do recurso e o ARN do cluster como o ARN do recurso. Você pode ver todos os compartilhamentos com o seguinte comando. Os resultados indicam quais contas da AWS têm permissão para clonar o cluster.

```
aws ram list-principals \
    --resource-arns your_cluster_arn \
    --principals your_aws_id
```

AWS RAMAPI do

Para descobrir se um cluster seu está compartilhado com outras contas da AWS

- Chame a operação de API do AWS RAM ListPrincipals. Use o ID da conta como o proprietário do recurso e o ARN do cluster como o ARN do recurso.

Clonar um cluster que pertence a outra conta da AWS

Para clonar um cluster que é de propriedade de outra conta da AWS, use o AWS RAM para obter permissão para fazer o clone. Depois de obter a permissão necessária, utilize o procedimento padrão para clonar um cluster do Aurora.

Também é possível verificar se um cluster seu é um clone de um cluster de propriedade de outra conta da AWS.

Para ver os procedimentos necessários para trabalhar com recursos de propriedade de outros no console do AWS RAM, consulte Accessing resources shared with you (Acesso aos recursos compartilhados com você) no Manual do usuário do AWS RAM.

Tópicos

- Visualizar convites para clonar clusters quer pertencem a outras contas da AWS
- Aceitação de convites para compartilhar clusters de propriedade de outras contas da AWS
- Clonar um cluster do Aurora de propriedade de outra conta da AWS
- Verificação de se um cluster de banco de dados é um clone entre contas

Visualizar convites para clonar clusters quer pertencem a outras contas da AWS

Para trabalhar com convites para clonar clusters de propriedade de contas da AWS em outras organizações da AWS, use a AWS CLI, o console do AWS RAM ou a API do AWS RAM. No momento, não é possível realizar esse procedimento usando o console do Amazon RDS.

Para ver os procedimentos necessários para trabalhar com convites no console do AWS RAM, consulte Accessing resources shared with you (Acesso aos recursos compartilhados com você) no Manual do usuário do AWS RAM.

AWS CLI

Para ver convites para clonar clusters de propriedade de outras contas da AWS

1. Execute o comando da CLI do AWS RAM `get-resource-share-invitations`.

   ```
   aws ram get-resource-share-invitations --region region_name
   ```

   Os resultados do comando anterior mostram todos os convites para clonar clusters, incluindo aqueles que você já aceitou ou rejeitou.

2. (Opcional) Filtre a lista para ver apenas os convites que precisam de alguma ação da sua parte. Para fazer isso, adicione o parâmetro `--query 'resourceShareInvitations[?status==`PENDING`]'`.

AWS RAMAPI do

Para ver convites para clonar clusters de propriedade de outras contas da AWS

1. Chame a operação da API do AWS RAM `GetResourceShareInvitations`. Essa operação retorna todos esses convites, incluindo aqueles que você já aceitou ou rejeitou.
2. (Opcional) Encontre apenas os convites que precisam de alguma ação da sua parte procurando no campo de retorno `resourceShareAssociations` o valor de `status` como `PENDING`.

Aceitação de convites para compartilhar clusters de propriedade de outras contas da AWS

Você pode aceitar convites para compartilhar clusters de propriedade de outras contas da AWS que estão em organizações da AWS diferentes. Para trabalhar com esses convites, use a AWS CLI, o AWS RAM ou as APIs do RDS, do console do AWS RAM. No momento, não é possível realizar esse procedimento usando o console do RDS.

Para ver os procedimentos necessários para trabalhar com convites no console do AWS RAM, consulte Accessing resources shared with you (Acesso aos recursos compartilhados com você) no Manual do usuário do AWS RAM.

AWS CLI

Para aceitar um convite para compartilhar um cluster de outra conta da AWS

1. Encontre o ARN do convite executando o comando da CLI do AWS RAM `get-resource-share-invitations`, conforme mostrado antes.
2. Aceite o convite chamando o comando da CLI do AWS RAM `accept-resource-share-invitation`, conforme mostrado a seguir.

   Para Linux, macOS ou Unix:

```
aws ram accept-resource-share-invitation \
  --resource-share-invitation-arn invitation_arn \
  --region region
```

Para Windows:

```
aws ram accept-resource-share-invitation ^
  --resource-share-invitation-arn invitation_arn ^
  --region region
```

AWS RAM e API do RDS

Para aceitar convites para compartilhar o cluster de alguém

1. Encontre o ARN do convite chamando a operação de API do AWS RAM `GetResourceShareInvitations`, conforme mostrado antes.
2. Transmita esse ARN como o parâmetro `resourceShareInvitationArn` para a operação de API do RDS AcceptResourceShareInvitation.

Clonar um cluster do Aurora de propriedade de outra conta da AWS

Depois de aceitar o convite da conta da AWS que tem o cluster de banco de dados, conforme mostrado antes, você pode clonar o cluster.

Console

Para clonar um cluster do Aurora de propriedade de outra conta da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).

   Na parte posterior da lista de bancos de dados, você deve ver um ou mais itens com o valor de Role (Função) igual a `Shared from account #account_id`. Por motivos de segurança, você pode ver apenas informações limitadas sobre os clusters originais. As propriedades que podem ser vistas são aquelas como mecanismo de banco de dados e versão que devem ser iguais aos do cluster clonado.
3. Escolha o cluster que você pretende clonar.
4. Em Actions (Ações), escolha Create alias (Criar alias).
5. Siga o procedimento descrito em Console para terminar de configurar o cluster clonado.

6. Conforme necessário, ative a criptografia para o cluster clonado. Se o cluster que estiver sendo clonado for criptografado, você deverá ativar a criptografia para o cluster clonado. A conta da AWS que compartilhou o cluster com você também deve compartilhar a chave do KMS que foi usada para criptografar o cluster. Você pode usar a mesma chave do KMS para criptografar o clone ou a sua própria chave do KMS. Não é possível criar um clone entre contas de um cluster que é criptografado com a chave do KMS padrão.

   A conta proprietária da chave de criptografia deve conceder permissão para usar a chave para a conta de destino usando uma política de chave. Esse processo é semelhante a como os snapshots criptografados são compartilhados, usando uma política de chave que concede permissão para a conta de destino usar a chave.

AWS CLI

Para clonar um cluster do Aurora de propriedade de outra conta da AWS

1. Aceite o convite da conta da AWS que tem o cluster de banco de dados, conforme mostrado antes.
2. Clone o cluster especificando o ARN completo do cluster de origem no parâmetro `source-db-cluster-identifier` do comando da CLI do RDS `restore-db-cluster-to-point-in-time`, conforme mostrado a seguir.

   Se o ARN transmitido como o `source-db-cluster-identifier` não tiver sido compartilhado, o mesmo erro será retornado como se o cluster especificado não existisse.

   Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier=arn:aws:rds:arn_details \
  --db-cluster-identifier=new_cluster_id \
  --restore-type=copy-on-write \
  --use-latest-restorable-time
```

   Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
```

```
  --use-latest-restorable-time
```

3. Se o cluster que está sendo clonado for criptografado, criptografe o cluster clonado incluindo um parâmetro `kms-key-id`. Esse valor de `kms-key-id` pode ser igual ao usado para criptografar o cluster de banco de dados original ou sua própria chave do KMS. Sua conta deve ter permissão para usar essa chave de criptografia.

   Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier=arn:aws:rds:arn_details \
  --db-cluster-identifier=new_cluster_id \
  --restore-type=copy-on-write \
  --use-latest-restorable-time \
  --kms-key-id=arn:aws:kms:arn_details
```

   Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
  --use-latest-restorable-time ^
  --kms-key-id=arn:aws:kms:arn_details
```

   A conta proprietária da chave de criptografia deve conceder permissão para usar a chave para a conta de destino usando uma política de chave. Esse processo é semelhante a como os snapshots criptografados são compartilhados, usando uma política de chave que concede permissão para a conta de destino usar a chave. Veja a seguir um exemplo de uma política de chave.

   JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> O comando restore-db-cluster-to-point-in-time da AWS CLI restaura o cluster de banco de dados, não as instâncias de banco de dados para esse cluster de banco de dados. Para criar instâncias de banco de dados para o cluster de banco de dados restaurado, chame o comando create-db-instance. Especifique o identificador do cluster de banco de dados restaurado em `--db-cluster-identifier`.
> Você pode criar instâncias de banco de dados somente após o comando `restore-db-cluster-to-point-in-time` tiver sido concluído e o cluster de banco de dados estiver disponível.

API do RDS

Para clonar um cluster do Aurora de propriedade de outra conta da AWS

1. Aceite o convite da conta da AWS que tem o cluster de banco de dados, conforme mostrado antes.
2. Clone o cluster especificando o ARN completo do cluster de origem no parâmetro `SourceDBClusterIdentifier` da operação de API do RDS `RestoreDBClusterToPointInTime`.

   Se o ARN transmitido como o `SourceDBClusterIdentifier` não tiver sido compartilhado, o mesmo erro será retornado como se o cluster especificado não existisse.
3. Se o cluster que está sendo clonado for criptografado, inclua um parâmetro `KmsKeyId` para criptografar o cluster clonado. Esse valor de `kms-key-id` pode ser igual ao usado para criptografar o cluster de banco de dados original ou sua própria chave do KMS. Sua conta deve ter permissão para usar essa chave de criptografia.

   Ao clonar um volume, a conta de destino deve ter permissão para usar a chave de criptografia usada para criptografar o cluster de origem. O Aurora criptografa o novo cluster clonado com a chave de criptografia especificada em `KmsKeyId`.

   A conta proprietária da chave de criptografia deve conceder permissão para usar a chave para a conta de destino usando uma política de chave. Esse processo é semelhante a como os snapshots criptografados são compartilhados, usando uma política de chave que concede permissão para a conta de destino usar a chave. Veja a seguir um exemplo de uma política de chave.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
```

```
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> A operação de API RestoreDBClusterToPointInTime do RDS restaura o cluster de banco de dados, não as instâncias de banco de dados desse cluster. Para criar instâncias primárias para o cluster de banco de dados, chame a operação de API CreateDBInstance do RDS. Especifique o identificador do cluster de banco de dados restaurado em `DBClusterIdentifier`. É possível criar instâncias de banco de dados somente após a operação `RestoreDBClusterToPointInTime` ter sido concluída e o cluster de banco de dados estar disponível.

Verificação de se um cluster de banco de dados é um clone entre contas

O objeto `DBClusters` identifica se cada cluster é um clone entre contas. Você pode ver os clusters que têm permissão para clonar usando a opção `include-shared` ao executar o comando da CLI do RDS `describe-db-clusters`. No entanto, não é possível ver a maioria dos detalhes de configuração desses clusters.

AWS CLI

Para verificar se um cluster de banco de dados é um clone entre contas

- Chame o comando da CLI do RDS `describe-db-clusters`.

  O exemplo a seguir mostra como clusters de banco de dados de clones entre contas reais ou em potencial aparecem na saída `describe-db-clusters`. Para os clusters existentes de propriedade da sua conta da AWS, o campo `CrossAccountClone` indica se o cluster é um clone de um cluster de banco de dados que é de propriedade de outra conta da AWS.

  Em alguns casos, uma entrada pode ter um número de conta da AWS diferente do seu no campo `DBClusterArn`. Nesse caso, essa entrada representa um cluster que é de propriedade de outra conta da AWS e que você pode clonar. Essas entradas têm alguns campos

diferentes de `DBClusterArn`. Ao criar o cluster clonado, especifique os mesmos valores de `StorageEncrypted`, `Engine` e `EngineVersion` do cluster original.

```
$aws rds describe-db-clusters --include-shared --region us-east-1
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0
  ]
}
```

API do RDS

Para verificar se um cluster de banco de dados é um clone entre contas

- Chame a operação de API do RDS DescribeDBClusters.

  Para os clusters existentes de propriedade da sua conta da AWS, o campo `CrossAccountClone` indica se o cluster é um clone de um cluster de banco de dados que é de propriedade de outra conta da AWS. As entradas com um número de conta da AWS diferente no campo `DBClusterArn` representam clusters que você pode clonar e que são de propriedade de outras contas da AWS. Essas entradas têm alguns campos diferentes de `DBClusterArn`.

Ao criar o cluster clonado, especifique os mesmos valores de `StorageEncrypted`, `Engine` e `EngineVersion` do cluster original.

O exemplo a seguir mostra um valor de retorno que demonstra clusters clonados reais e em potencial.

```
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0"
      }
  ]
}
```

Integração do Aurora com outros produtos da AWS

Integre o Amazon Aurora a outros produtos da AWS para que você possa estender seu cluster de banco de dados Aurora de forma a usar recursos adicionais na Nuvem AWS.

Tópicos

- Integrar produtos da AWS com o Amazon Aurora MySQL
- Integrar produtos da AWS com o Amazon Aurora PostgreSQL

Integrar produtos da AWS com o Amazon Aurora MySQL

O Amazon Aurora MySQL integra-se a outros produtos da AWS para que você possa estender seu cluster de banco de dados do Aurora MySQL de forma a usar recursos adicionais na Nuvem AWS. O cluster de banco de dados Aurora MySQL pode usar produtos da AWS para fazer o seguinte:

- Invoque de forma síncrona ou assíncrona uma função do AWS Lambda usando as funções nativas `lambda_sync` ou `lambda_async`. Ou invoque de forma assíncrona uma função do AWS Lambda usando o procedimento `mysql.lambda_async`.
- Carregue os dados de arquivos de texto ou XML armazenados em um bucket do Amazon S3 no seu cluster de banco de dados usando o comando `LOAD DATA FROM S3` ou `LOAD XML FROM S3`.
- Salve os dados em arquivos de texto armazenados em um bucket do Amazon S3 a partir de seu cluster de banco de dados usando o comando `SELECT INTO OUTFILE S3`.
- Adicione ou remova réplicas do Aurora automaticamente com o Application Auto Scaling. Para obter mais informações, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.

Para obter mais informações sobre como integrar o Aurora MySQL a outros produtos da AWS, consulte Integração do Amazon Aurora MySQL com outros produtos da AWS.

Integrar produtos da AWS com o Amazon Aurora PostgreSQL

O Amazon Aurora PostgreSQL integra-se a outros produtos da AWS para que você possa estender seu cluster de banco de dados Aurora PostgreSQL de forma a usar recursos adicionais na Nuvem AWS. Seu cluster de banco de dados Aurora PostgreSQL pode usar produtos da AWS para fazer o seguinte:

- Coletar, visualizar e avaliar rapidamente a performance nas workloads do banco de dados relacional com o Performance Insights.
- Adicione ou remova réplicas do Aurora automaticamente com o Aurora Auto Scaling. Para obter mais informações, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.

Para obter mais informações sobre como integrar o Aurora PostgreSQL a outros produtos da AWS, consulte Integração do Amazon Aurora PostgreSQL com outros produtos da AWS.

Manutenção de um cluster de banco de dados do Amazon Aurora

Periodicamente, o Amazon RDS realiza a manutenção em seus recursos. Os tópicos a seguir descrevem essas ações de manutenção e como aplicá-las.

Visão geral das atualizações de manutenção do cluster de banco de dados

A manutenção geralmente envolve atualizações dos seguintes atributos no cluster de banco de dados:

- Hardware subjacente
- Sistema operacional subjacente
- Versão do mecanismo de banco de dados

As atualizações no sistema operacional geralmente ocorrem para problemas de segurança. Recomendamos que você os faça o quanto antes. Para ter mais informações sobre os sistemas operacionais compatíveis, consulte the section called “Atualizações do sistema operacional”.

Tópicos

- Recursos off-line durante as atualizações de manutenção
- Modificações adiadas da instância de banco de dadose do cluster de banco de dados
- Consistência eventual da API DescribePendingMaintenanceActions

Recursos off-line durante as atualizações de manutenção

Alguns itens de manutenção exigem que o Amazon RDS coloque o cluster de banco de dados off-line por um curto período. Entre os itens de manutenção que exigem um recurso esteja offline estão sistema operacional obrigatório ou patches de banco de dados. A aplicação obrigatória de patches é automaticamente programada somente para patches relacionados à segurança e à confiabilidade da instância. Essa correção ocorre com pouca frequência, normalmente uma vez a cada poucos meses. Raramente requer mais do que uma fração de sua janela de manutenção.

Modificações adiadas da instância de banco de dadose do cluster de banco de dados

As modificações adiadas no cluster e na instância de banco de dados que você optou por não aplicar imediatamente são aplicadas durante a janela de manutenção. Por exemplo, você pode optar por

alterar classes de instância de banco de dados ou grupos de parâmetros de cluster ou banco de dados durante a janela de manutenção. Essas modificações especificadas usando a configuração pending reboot (reinicialização pendente) não aparecem na lista Pending maintenance (Manutenção pendente). Para obter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Para ver as modificações pendentes para a próxima janela de manutenção, use o comando describe-db-clusters da AWS CLI e selecione o campo `PendingModifiedValues`.

Consistência eventual da API DescribePendingMaintenanceActions

A API `DescribePendingMaintenanceActions` do Amazon RDS segue o modelo de consistência eventual. Isso significa que o resultado do comando `DescribePendingMaintenanceActions` poderá não ser imediatamente visível para todos os comandos subsequentes do RDS. Lembre-se disso ao usar `DescribePendingMaintenanceActions` imediatamente após usar um comando de API anterior.

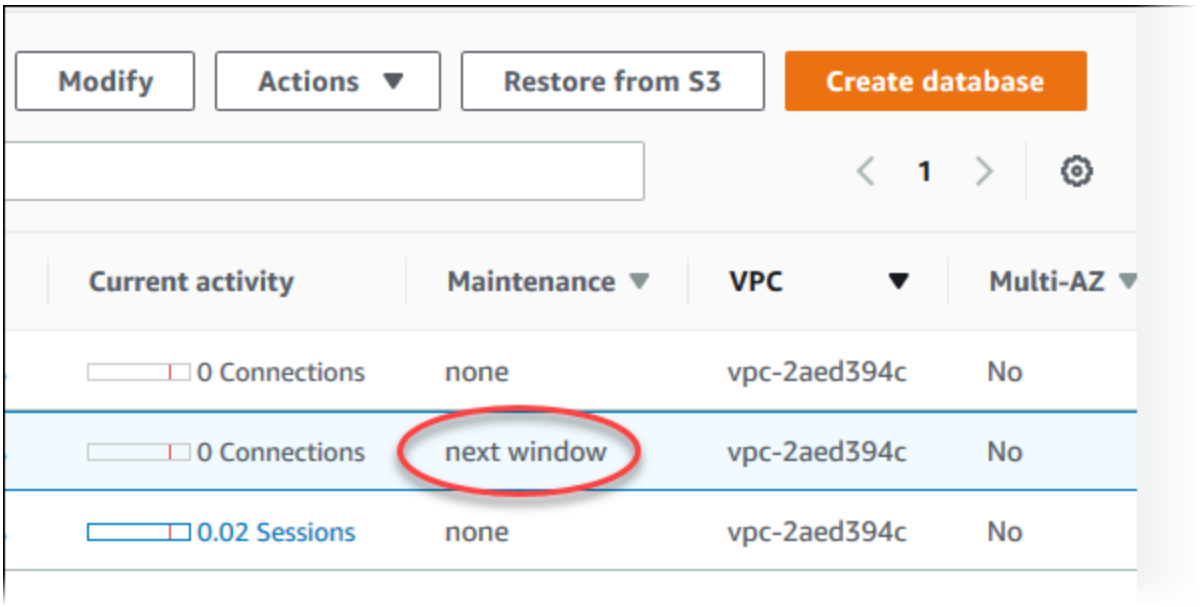
A consistência eventual pode afetar a maneira como você gerencia suas atualizações de manutenção. Por exemplo, se você executar o comando `ApplyPendingMaintenanceActions` para atualizar a versão do mecanismo de banco de dados para um cluster de banco de dados, ela ficará visível para `DescribePendingMaintenanceActions`. Nesse cenário, `DescribePendingMaintenanceActions` pode mostrar que a ação de manutenção não foi aplicada, embora tenha sido.

Para gerenciar a consistência eventual, é possível fazer o seguinte:

- Confirme o estado do cluster de banco de dados antes de executar um comando para modificá-lo. Execute o comando `DescribePendingMaintenanceActions` apropriado usando um algoritmo de recuo exponencial para garantir tempo suficiente para que o comando anterior se propague pelo sistema. Para fazer isso, execute o comando `DescribePendingMaintenanceActions` repetidamente, começando com alguns segundos de espera e aumentando gradualmente até cinco minutos de espera.
- Adicione o tempo de espera entre os comandos subsequentes, mesmo que um comando `DescribePendingMaintenanceActions` exiba uma resposta precisa. Aplique um algoritmo de recuo exponencial começando com alguns segundos de espera e aumente gradualmente até cerca de cinco minutos de espera.

Visualizar atualizações de manutenção pendentes

Veja se uma atualização de manutenção está disponível para seu cluster de banco de dados utilizando o console do RDS, a AWS CLI ou a API do RDS. Se estiver disponível, uma atualização será indicada na coluna Manutenção do cluster de banco de dados no console do Amazon RDS, conforme mostrado nesta figura.



Se nenhuma atualização de manutenção estiver disponível para um cluster de banco de dados, o valor da coluna será none.

Se uma atualização de manutenção estiver disponível para um cluster de banco de dados, os seguintes valores de coluna serão possíveis:

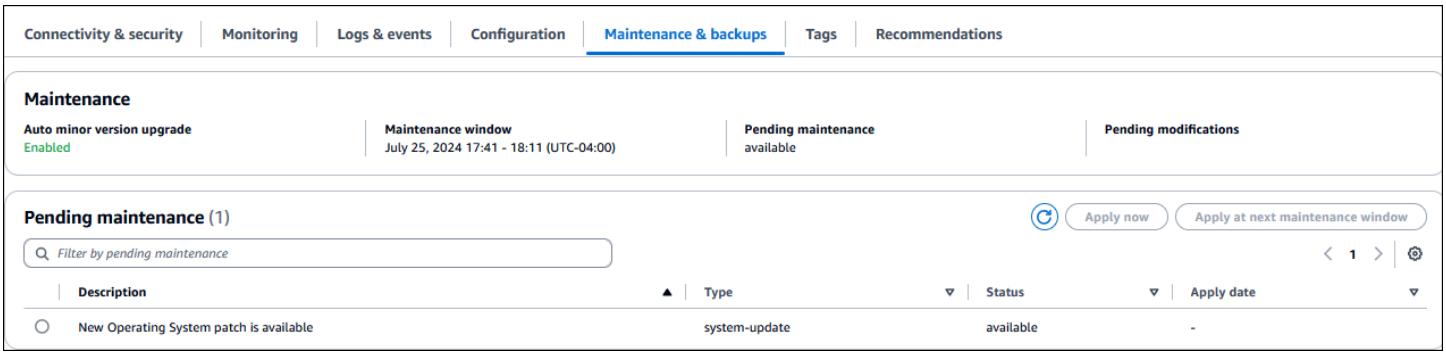
- obrigatório – a ação de manutenção será aplicada ao recurso e não pode ser adiada indefinidamente.
- available (disponível) – a ação de manutenção está disponível, mas não será aplicada automaticamente ao recurso. Você pode aplicá-la manualmente.
- next window (próxima janela) – a ação de manutenção será aplicada ao recurso durante a próxima janela de manutenção.
- Em andamento: a ação de manutenção está sendo aplicada ao recurso.

Se uma atualização estiver disponível, será possível executar uma das seguintes ações:

- Se o valor de manutenção for próxima janela, adie as ações de manutenção escolhendo Adiar atualização em Ações. Não é possível adiar uma ação de manutenção que já tiver sido iniciada.
- Aplicar as ações de manutenção imediatamente.
- Aplique as ações de manutenção durante a próxima janela de manutenção.
- Não tome nenhuma ação.

Como realizar uma ação usando o Console de gerenciamento da AWS

1. Escolha a instância ou cluster de banco de dados para mostrar os detalhes.
2. Escolha Maintenance & backups (Manutenção e backups). As ações de manutenção pendentes são exibidas.
3. Escolha a ação a ser tomada e, em seguida, escolha quando aplicá-la.



A janela de manutenção determina quando as operações pendentes começam, mas não limita o tempo total de execução dessas operações. Não há garantia de que as operações de manutenção terminem antes de a janela de manutenção se encerrar, podendo continuar além do tempo de encerramento especificado. Para obter mais informações, consulte Janela de manutenção do Amazon RDS.

Você pode ver se uma atualização de manutenção está disponível para seu cluster de banco de dados executando o comando describe-pending-maintenance-actions da AWS CLI.

Para ter informações sobre a aplicação de atualizações de manutenção, consulte Aplicar atualizações para um clusterde banco de dados.

Ações de manutenção do Amazon Aurora

As ações de manutenção a seguir se aplicam a clusters de banco de dados do Aurora:

- `os-upgrade`: atualize os sistemas operacionais de todas as instâncias de banco de dados no cluster de banco de dados, usando atualizações progressivas. Para obter mais informações, consulte the section called “Atualizações do sistema operacional”.
- `system-update`: aplique patch no mecanismo de banco de dados para o Aurora PostgreSQL.

As ações de manutenção a seguir se aplicam a instâncias de banco de dados do Aurora:

- `ca-certificate-rotation`: atualize o certificado da Autoridade Certificadora do Amazon RDS para a instância de banco de dados.
- `hardware-maintenance`: execute a manutenção no hardware subjacente da instância de banco de dados.
- `system-update`: atualize o sistema operacional da instância de banco de dados.

Escolher a frequência das atualizações de manutenção do Aurora MySQL

Você pode controlar se as atualizações do Aurora MySQL ocorrem com frequência ou raramente para cada cluster de banco de dados. A melhor opção depende do seu uso do Aurora MySQL e das prioridades das aplicações que são executadas no Aurora. Para saber mais sobre as versões de estabilidade a longo prazo (LTS) do Aurora MySQL que exigem upgrades menos frequentes, consulte Versões de suporte de longo prazo (LTS) do Aurora MySQL.

Você pode optar por atualizar um cluster Aurora MySQL raramente se algumas ou todas as seguintes condições se aplicarem:

- O ciclo de testes para o aplicativo exige muito tempo para cada atualização no mecanismo de banco de dados Aurora MySQL.
- Você tem muitos clusters de banco de dados ou muitos aplicativos, todos em execução na mesma versão do Aurora MySQL. Você prefere atualizar todos os clusters de banco de dados e aplicativos associados ao mesmo tempo.
- Use o Aurora MySQL e o RDS para MySQL. Você prefere manter os clusters do Aurora MySQL e as instâncias de banco de dados do RDS para MySQL compatíveis com o mesmo nível do MySQL.
- Seu aplicativo Aurora MySQL está em produção ou é essencial para os negócios. Não é possível permitir tempo de inatividade para atualizações fora de ocorrências raras para patches importantes.

- O aplicativo Aurora MySQL não é limitado por problemas de performance ou falhas de recursos que são abordados nas versões subsequentes do Aurora MySQL.

Se os fatores anteriores se aplicarem à situação, você poderá limitar o número de atualizações forçadas para um cluster de banco de dados do Aurora MySQL. Você faz isso escolhendo uma versão específica do Aurora MySQL conhecida como versão de "suporte de longo prazo" (LTS) quando cria ou atualiza esse cluster de banco de dados. Isso minimiza o número de ciclos de atualização, ciclos de testes e interrupções relacionadas à atualização para esse cluster de banco de dados.

Você pode optar por atualizar um cluster Aurora MySQL frequentemente se algumas ou todas as seguintes condições se aplicarem:

- O ciclo de testes para o aplicativo é simples e breve.
- O aplicativo ainda está na fase de desenvolvimento.
- O ambiente de banco de dados usa diversas versões do Aurora MySQL ou as versões do Aurora MySQL e do RDS for MySQL. Cada cluster do Aurora MySQL tem seu próprio ciclo de atualização.
- Você está esperando por melhorias específicas de performance ou recursos antes de aumentar o uso do Aurora MySQL.

Se os fatores anteriores se aplicarem à situação, você poderá habilitar o Aurora para aplicar atualizações importantes com maior frequência. Para fazer isso, atualize um cluster de banco de dados do Aurora MySQL para uma versão mais recente do Aurora MySQL que a versão LTS. Fazer isso disponibiliza os mais recentes aprimoramentos de performance, correções de erros e recursos mais rapidamente.

Janela de manutenção do Amazon RDS

A janela de manutenção é um intervalo de tempo semanal durante o qual todas as alterações do sistema são aplicadas. Cada cluster de banco de dados tem uma janela de manutenção semanal. A janela de manutenção é uma oportunidade de controlar quando as modificações e a aplicação de patches de software ocorrem. Para ter mais informações sobre como ajustar a janela de manutenção, consulte Ajustar a janela de manutenção do cluster de banco de dados preferencial.

O RDS consome alguns dos recursos em seu cluster de banco de dados enquanto a manutenção é aplicada. Você poderá observar um impacto mínimo na performance. Quanto a uma instância de

banco de dados, em raras ocasiões, pode ser necessário realizar um failover Multi-AZ para concluir uma atualização de manutenção.

Se um evento de manutenção estiver programado para determinada semana, ele será iniciado durante a janela de manutenção de 30 minutos que você identificar. A maioria dos eventos de manutenção também é concluída durante a janela de manutenção de 30 minutos, embora os eventos de manutenção mais longos possam levar mais de 30 minutos para serem concluídos. A janela de manutenção é pausada quando o cluster de banco de dados é interrompido.

A janela de manutenção de 30 minutos é selecionada aleatoriamente de um bloco de tempo de 8 horas por região. Se você não especificar uma janela de manutenção ao criar o cluster de banco de dados, o RDS atribuirá uma janela de manutenção de 30 minutos em um dia da semana selecionado aleatoriamente.

A tabela a seguir mostra os blocos de tempo de cada Região da AWS dos quais as janelas de manutenção padrão são atribuídas.

Nome da Região	Região	Bloco de hora
Leste dos EUA (Norte da Virgínia)	us-east-1	Das 3h às 11h (UTC)
Leste dos EUA (Ohio)	us-east-2	De 03:00 a 11:00 UTC
US West (N. Californi a)	us-west-1	De 06:00 a 14:00 UTC
US West (Oregon)	us-west-2	De 06:00 a 14:00 UTC
Africa (Cape Town)	af-south-1	De 03:00 a 11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	6h às 14h (UTC)
Ásia-Pacífico (Hyderabad)	ap-south-2	06h30 a 14h30 UTC
Ásia-Pacífico (Jacarta)	ap-southeast-3	Das 08h às 16h UTC

Nome da Região	Região	Bloco de hora
Ásia-Pacífico (Malásia)	ap-southeast-5	Das 9h às 17h (UTC)
Ásia-Pacífico (Melbourne)	ap-southeast-4	Das 11h às 19h UTC
Ásia-Pacífico (Mumbai)	ap-south-1	6h às 14h (UTC)
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	Das 13h às 21h (UTC)
Ásia-Pacífico (Osaka)	ap-northeast-3	De 22:00 a 23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	De 13:00 a 21:00 UTC
Ásia-Pacífico (Singapura)	ap-southeast-1	De 14:00 a 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	De 12:00 a 20:00 UTC
Ásia-Pacífico (Taipei)	ap-east-2	Das 9h às 17h (UTC)
Ásia-Pacífico (Tailândia)	ap-southeast-7	Das 8h às 16h (UTC)
Ásia-Pacífico (Tóquio)	ap-northeast-1	De 13:00 a 21:00 UTC
Canada (Central)	ca-central-1	Das 3h às 11h (UTC)
Oeste do Canadá (Calgary)	ca-west-1	Das 18h às 2h (UTC)
China (Pequim)	cn-north-1	De 06:00 a 14:00 UTC
China (Ningxia)	cn-northwest-1	De 06:00 a 14:00 UTC
Europe (Frankfurt)	eu-central-1	De 21:00 a 05:00 UTC

Nome da Região	Região	Bloco de hora
Europe (Ireland)	eu-west-1	De 22:00 a 06:00 UTC
Europe (London)	eu-west-2	Das 22:00 às 06:00 (UTC)
Europa (Milão)	eu-south-1	De 02:00 a 10:00 UTC
Europa (Paris)	eu-west-3	De 23:59 a 07:29 UTC
Europa (Espanha)	eu-south-2	De 02:00 a 10:00 UTC
Europe (Stockholm)	eu-north-1	De 23:00 a 07:00 UTC
Europa (Zurique)	eu-central-2	De 02:00 a 10:00 UTC
Israel (Tel Aviv)	il-central-1	Das 3h às 11h (UTC)
México (Central)	mx-central-1	Das 19h às 3h (UTC)
Oriente Médio (Barém)	me-south-1	6h às 14h (UTC)
Oriente Médio (Emirados Árabes Unidos)	me-central-1	Das 5h às 13h UTC
América do Sul (São Paulo)	sa-east-1	De 00:00 a 08:00 UTC
AWS GovCloud (Leste dos EUA)	us-gov-east-1	De 17:00 a 01:00 UTC
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	De 06:00 a 14:00 UTC

Tópicos

- Ajustar a janela de manutenção do cluster de banco de dados preferencial

Ajustar a janela de manutenção do cluster de banco de dados preferencial

A janela de manutenção do cluster de banco de dados do Aurora deve ser definida no horário de menor utilização e, portanto, talvez precise ser modificada de vez em quando. Seu cluster de banco de dados ficará indisponível durante esse período apenas se as atualizações aplicadas exigirem uma interrupção. A interrupção ocorre durante o tempo mínimo necessário para fazer as atualizações necessárias.

> **Note**
>
> Para atualizações no mecanismo de banco de dados, o Amazon Aurora gerencia a janela de manutenção preferencial de um cluster de banco de dados e não de instâncias individuais.

Console

Para ajustar a janela de manutenção do cluster de banco de dados preferencial

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados para o qual deseja alterar a janela de manutenção.
4. Selecione Modify.
5. Na seção Maintenance (Manutenção), atualize a janela de manutenção.
6. Escolha Continue.

   Na página de confirmação, revise suas alterações.
7. Para aplicar as alterações à janela de manutenção imediatamente, escolha Immediately (Imediatamente) na seção Schedule of modifications (Programação de modificações).
8. Selecione Modify cluster (Modificar cluster) para salvar suas alterações.

   Como alternativa, escolha Back (Voltar) para editar suas alterações ou escolha Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para ajustar a janela de manutenção do cluster de banco de dados preferencial, use o comando AWS CLI da `modify-db-cluster` com os seguintes parâmetros:

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

O exemplo de código a seguir define a janela de manutenção como terças, das 4h–4h30 UTC.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Para Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API do RDS

Para ajustar a janela de manutenção do cluster de banco de dados de sua preferência, use a operação `ModifyDBCluster` da API do Amazon RDS com os seguintes parâmetros:

- `DBClusterIdentifier`
- `PreferredMaintenanceWindow`

Aplicar atualizações para um clusterde banco de dados

Com o Amazon RDS, você pode escolher quando aplicar operações de manutenção. Você pode decidir quando o Amazon RDS aplicará atualizações usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para gerenciar uma atualização de um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados que exige uma atualização obrigatória.
4. Em Actions (Ações), escolha uma das seguinte opções:

   - Aplicar patch agora
   - Aplicar patch na próxima janela

   > **Note**
   >
   > Se escolher Aplicar patch na próxima janela e depois quiser atrasar a atualização do sistema operacional, você poderá escolher Adiar atualização. Não é possível adiar uma ação de manutenção que já tiver sido iniciada.
   > Para cancelar uma ação de manutenção, modifique a instância de banco de dados e desative Auto minor version upgrade (Atualização automática da versão secundária).

AWS CLI

Para aplicar uma atualização pendente a um cluster de banco de dados, use o comando apply-pending-maintenance-action da AWS CLI.

Example

Para Linux, macOS ou Unix:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
```

```
    --apply-action system-update \
    --opt-in-type immediate
```

Para Windows:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> Para adiar uma ação de manutenção, especifique `undo-opt-in` para `--opt-in-type`. Não será possível especificar `undo-opt-in` para `--opt-in-type` se a ação de manutenção já tiver sido iniciada.
> Para cancelar uma ação de manutenção, execute o comando modify-db-instance da AWS CLI e especifique `--no-auto-minor-version-upgrade`.

Para retornar uma lista de recursos que tenham pelo menos uma atualização pendente, use o comando describe-pending-maintenance-actions da AWS CLI.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Para Windows:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Você também pode exibir uma lista de recursos para um cluster de banco de dados especificando o parâmetro `--filters` do comando `describe-pending-maintenance-actions` da AWS CLI. O formato do comando `--filters` é `Name=filter-name,Value=resource-id,...`.

Os valores a seguir são os valores aceitos para o parâmetro `Name` de um filtro:

- `db-instance-id` – aceita uma lista de identificadores de instância de banco de dados ou nomes de recurso da Amazon (ARNs). A lista retornada inclui apenas ações de manutenção pendentes para as instâncias de bancos de dados identificadas por esses identificadores ou ARNs.
- `db-cluster-id` – aceita uma lista de identificadores de cluster de banco de dados ou ARNs para o Amazon Aurora. A lista retornada inclui apenas ações de manutenção pendentes para os clusters de bancos de dados identificados por esses identificadores ou ARNs.

Por exemplo, o exemplo a seguir retorna as ações de manutenção pendentes para os clusters de banco de dados `sample-cluster1` e `sample-cluster2`.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Para Windows:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API do RDS

Para aplicar uma atualização a um cluster de banco de dados, chame a operação `ApplyPendingMaintenanceAction` da API do Amazon RDS.

Para retornar uma lista de recursos que tenham pelo menos uma atualização pendente, chame a operação `DescribePendingMaintenanceActions` da API do Amazon RDS.

Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora

A configuração Upgrade automático de versões secundárias especifica se o Aurora aplica upgrades automaticamente em seu cluster. Esses upgrades incluem novas versões secundárias que contêm atributos adicionais e patches que contêm correções de bugs.

As atualizações automáticas de versões secundárias atualizam periodicamente o banco de dados para versões recentes do respectivo mecanismo. No entanto, a atualização nem sempre inclui

a versão mais recente do mecanismo de banco de dados. Se precisar manter seus bancos de dados em determinadas versões em momentos específicos, recomendamos atualizar manualmente para as versões de banco de dados necessárias de acordo com o cronograma exigido. Em casos de problemas críticos de segurança ou quando uma versão atinge a data de fim do suporte, o Amazon Aurora pode aplicar uma atualização de versão secundária, mesmo que você não tenha habilitado a opção Upgrade automático de versões secundárias. Para ter mais informações, consulte a documentação sobre atualização do seu mecanismo de banco de dados específico.

Consulte Atualizando a versão secundária ou o nível de patch de um cluster de banco de dados de Aurora MySQL e Realizar uma atualização da versão secundária.

> **Note**
>
> O Aurora Global Database não é compatível com atualizações automáticas de versões secundárias.

Essa configuração está ativada por padrão. Para cada novo cluster de banco de dados, selecione o valor apropriado para essa configuração. Esse valor é baseado em sua importância, tempo de vida esperado e a quantidade de testes de verificação que você faz após cada atualização.

Para obter instruções sobre como ativar ou desativar a configuração Upgrade automático de versões secundárias, consulte o seguinte:

- Ativar upgrades automáticos de versões secundárias para um cluster de banco de dados do Aurora
- Ativar upgrades automáticos de versões secundárias para instâncias individuais em um cluster de banco de dados do Aurora

> **Important**
>
> É altamente recomendável que, para clusters de banco de dados novos e existentes, você aplique essa configuração ao cluster de banco de dados e não às instâncias de banco de dados do cluster individualmente. Se alguma instância de banco de dados em seu cluster tiver essa configuração desativada, o cluster de banco de dados não receberá upgrade automaticamente.

A tabela a seguir mostra como a configuração Upgrade automático de versões secundárias funciona quando aplicada nos níveis de cluster e instância.

Ação	Configuração do cluster	Configurações das instâncias	O cluster recebe upgrade automatic amente?
Você define como True no cluster de banco de dados.	Verdadeiro	True para todas as instâncias novas e existentes	Sim
Você define como False no cluster de banco de dados.	Falso	False para todas as instâncias novas e existentes	Não
Já estava definido como True no cluster de banco de dados. Você define como False em pelo menos uma instância de banco de dados.	Muda para False	False para uma ou mais instâncias	Não
Já estava definido como False no cluster de banco de dados. Você define como True em pelo menos uma instância de banco de dados, mas não em todas as instâncias.	Falso	True para uma ou mais instâncias, mas não para todas as instâncias	Não
Já estava definido como False no cluster de banco de dados.	Muda para True	True para todas as instâncias	Sim

Ação	Configuração do cluster	Configurações das instâncias	O cluster recebe upgrade automatic amente?
Você define como True em todas as instâncias de banco de dados.			

As atualizações automáticas de versões secundárias são comunicadas com antecedência por meio de um evento de cluster de banco de dados do Amazon RDS com a categoria `maintenance` e o ID `RDS-EVENT-0156`. Para obter mais informações, consulte Categorias de eventos e mensagens de eventos para o Amazon RDS.

Os upgrades automáticos ocorrem durante as janelas de manutenção. Se as instâncias de banco de dados individuais no cluster de banco de dados tiverem janelas de manutenção diferentes da janela de manutenção do cluster, a janela de manutenção do cluster terá precedência.

Para ter mais informações sobre atualizações de mecanismos para o Aurora PostgreSQL, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL.

Para ter mais informações sobre a configuração Auto minor version upgrade (Atualização automática de versão secundária) para o Aurora MySQL, consulte Habilitar atualizações automáticas entre versões secundárias do Aurora MySQL. Para obter informações gerais sobre atualizações de mecanismos para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

Ativar upgrades automáticos de versões secundárias para um cluster de banco de dados do Aurora

Siga o procedimento geral em Modificar o cluster de banco de dados usando o console, a CLI e a API.

Console

Na seção Manutenção da página Modificar cluster de banco de dados, marque a caixa de seleção Habilitar o upgrade automático da versão secundária.

AWS CLI

Chame o comando modify-db-cluster da AWS CLI. Especifique o nome do cluster de banco de dados para a opção `--db-cluster-identifier` e `true` para a opção `--auto-minor-version-upgrade`. Opcionalmente, especifique a opção `--apply-immediately` para habilitar imediatamente essa configuração para o cluster de banco de dados.

API do RDS

Chame a operação de API ModifyDBCluster e especifique o nome do cluster de banco de dados para o parâmetro `DBClusterIdentifier` e `true` para o parâmetro `AutoMinorVersionUpgrade`. Opcionalmente, defina o parâmetro `ApplyImmediately` como `true` para habilitar imediatamente essa configuração para o cluster de banco de dados.

Ativar upgrades automáticos de versões secundárias para instâncias individuais em um cluster de banco de dados do Aurora

Siga o procedimento geral em Modificar uma instância de banco de dados em um cluster de banco de dados.

Console

Na seção Manutenção da página Modificar instância de banco de dados, marque a caixa de seleção Habilitar o upgrade automático da versão secundária.

AWS CLI

Chame o comando modify-db-instance da AWS CLI. Especifique o nome da instância de banco de dados para a opção `--db-instance-identifier` e `true` para a opção `--auto-minor-version-upgrade`. Opcionalmente, especifique a opção `--apply-immediately` para habilitar imediatamente essa configuração para sua instância de banco de dados. Execute um comando `modify-db-instance` separado para cada instância de banco de dados no cluster.

API do RDS

Chame a operação de API ModifyDBInstance e especifique o nome do cluster de banco de dados para o parâmetro `DBInstanceIdentifier` e `true` para o parâmetro `AutoMinorVersionUpgrade`. Opcionalmente, defina o parâmetro `ApplyImmediately` como `true` para habilitar imediatamente essa configuração para sua instância de banco de dados. Chame uma operação `ModifyDBInstance` separada para cada instância de banco de dados no cluster.

É possível usar um comando de CLI como o seguinte para conferir o status da configuração `AutoMinorVersionUpgrade` para todas as instâncias de banco de dados em seus clusters do Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Esse comando gerará uma saída semelhante à seguinte:

```
[
  {
      "DBInstanceIdentifier": "db-writer-instance",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-reader-instance1",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "db-writer-instance2",
      "DBClusterIdentifier": "my-db-cluster-80",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

Neste exemplo, a opção Habilitar atualização automática da versão secundária está desativada para o cluster de banco de dados `my-db-cluster-57` porque está desativada para uma das instâncias de banco de dados no cluster.

Atualizações do sistema operacional para clusters de banco de dados do Aurora

As instâncias de banco de dados nos clusters de banco de dados do Aurora MySQL e do Aurora PostgreSQL ocasionalmente exigem atualizações do sistema operacional. O Amazon RDS faz upgrade do sistema operacional para uma versão mais recente para melhorar o desempenho do banco de dados e o procedimento de segurança geral dos clientes. Em geral, essas atualizações

demoram cerca de dez minutos. As atualizações do sistema operacional não alteram a versão do mecanismo de banco de dados nem a classe de uma instância de banco de dados.

Há dois tipos de atualização do sistema operacional, diferenciados pela descrição da ação de manutenção pendente:

- Atualização da distribuição do sistema operacional: usada para migrar para a versão principal mais recente compatível do Amazon Linux. A descrição é `New Operating System upgrade is available`.
- Patch do sistema operacional: usado para aplicar várias correções de segurança e, às vezes, para melhorar a performance do banco de dados. A descrição é `New Operating System patch is available`.

As atualizações do sistema operacional podem ser opcionais ou obrigatórias:

- Uma atualização opcional pode ser aplicada a qualquer momento. Embora essas atualizações sejam opcionais, recomendamos que você as aplique periodicamente para manter sua frota do RDS atualizada. O RDS não aplica essas atualizações automaticamente.

  Para ser notificado quando um novo patch opcional do sistema operacional estiver disponível, você poderá assinar o RDS-EVENT-0230 na categoria de evento de aplicação de patch de segurança. Para obter informações sobre como se inscrever em eventos do RDS, consulte Inscrever-se em notificações de eventos do Amazon RDS.

  > **Note**
  >
  > `RDS-EVENT-0230` não se aplica às atualizações de distribuição do sistema operacional.

- É necessária uma atualização obrigatória e enviamos uma notificação antes da atualização obrigatória. A notificação pode conter um prazo. Agende sua atualização para antes desse prazo. Após a data de prazo especificada, o Amazon RDS atualiza automaticamente o sistema operacional da instância de banco de dados para a versão mais recente durante uma das janelas de manutenção atribuídas.

  Os upgrades da distribuição do sistema operacional são obrigatórios.

> **Note**
>
> A aplicação de todas as atualizações opcionais e obrigatórias pode ser necessária para cumprir várias obrigações de conformidade. Recomendamos que você aplique todas as atualizações disponibilizadas pelo RDS rotineiramente durante suas janelas de manutenção.

Para clusters de banco de dados do Aurora, você pode usar a opção de manutenção em nível de cluster para realizar atualizações do sistema operacional (SO). Encontre a opção de realizar atualizações em nível de cluster na guia Manutenção e backups ao selecionar o nome do cluster de banco de dados no console ou usar o comando `os-upgrade` na AWS CLI. Esse método preserva a disponibilidade de leitura com atualizações contínuas que aplicam automaticamente as atualizações a algumas instâncias de banco de dados de leitura por vez. Para evitar vários failovers e reduzir o tempo de inatividade desnecessário, o Aurora atualiza a instância de banco de dados de gravador por último.

As atualizações do sistema operacional em nível de cluster ocorrem durante a janela de manutenção especificada para o cluster. Isso garante atualizações coordenadas para todo o cluster.

Para oferecer compatibilidade com versões anteriores, o Aurora também mantém a opção de manutenção em nível de instância. No entanto, recomendamos o uso de atualizações em nível de cluster. Se você precisar usar as atualizações em nível de instância, atualize primeiro as instâncias de banco de dados de leitor em um cluster de banco de dados e, depois, a instância de banco de dados de gravador. Se você atualizar as instâncias de leitor e de gravador simultaneamente, aumentará a chance de tempo de inatividade relacionado ao failover. Encontre a opção de realizar atualizações em nível de instância na guia Manutenção e backups ao selecionar o nome da instância de banco de dados no console ou usar o comando `system-update` na AWS CLI.

As atualizações do sistema operacional em nível da instância ocorrem durante a janela de manutenção que você especificou para cada instância em questão. Por exemplo, se o cluster e duas instâncias de leitor tiverem horários de janela de manutenção diferentes, a atualização do SO em nível de cluster se alinhará à janela de manutenção do cluster.

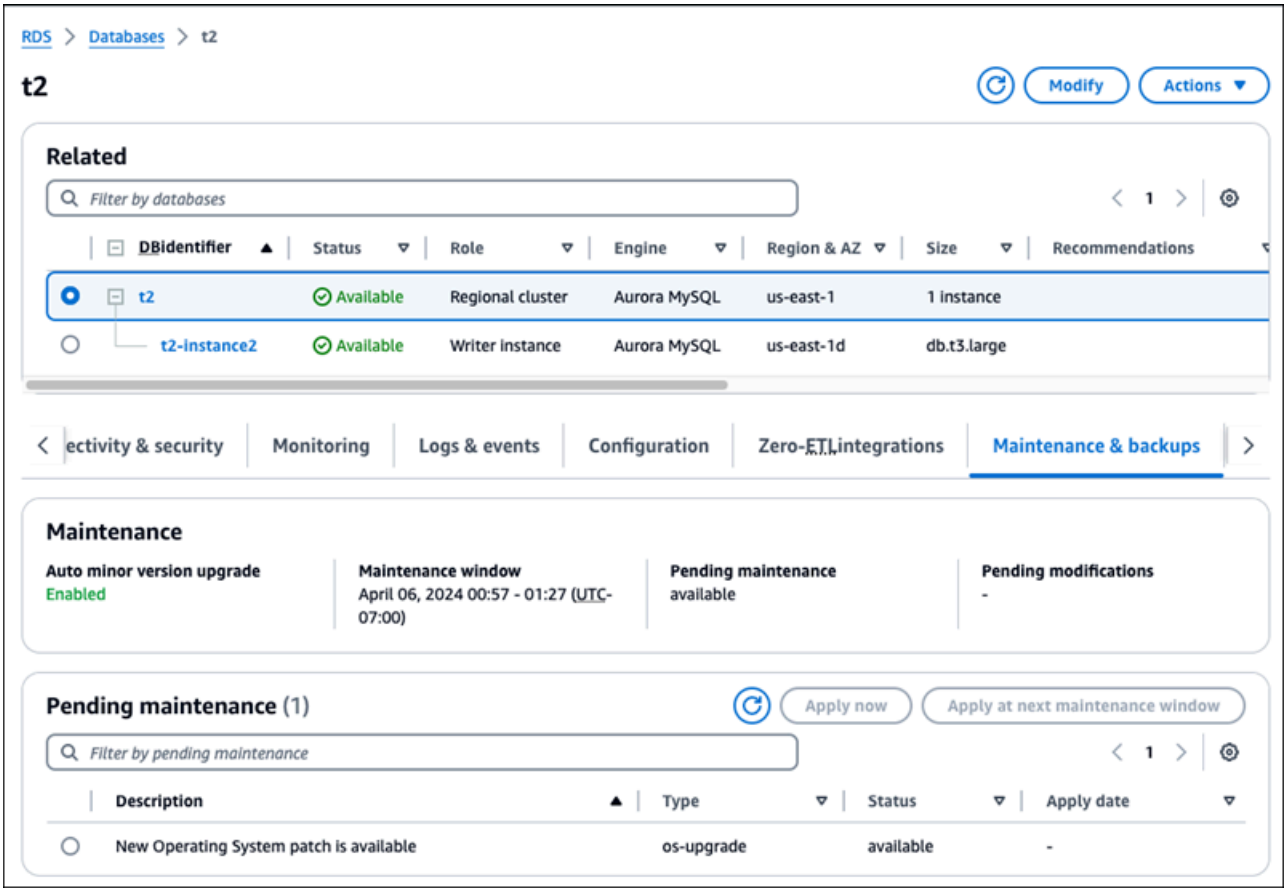
É possível usar o Console de gerenciamento da AWS ou a AWS CLI para obter informações sobre o tipo de atualização do sistema operacional.

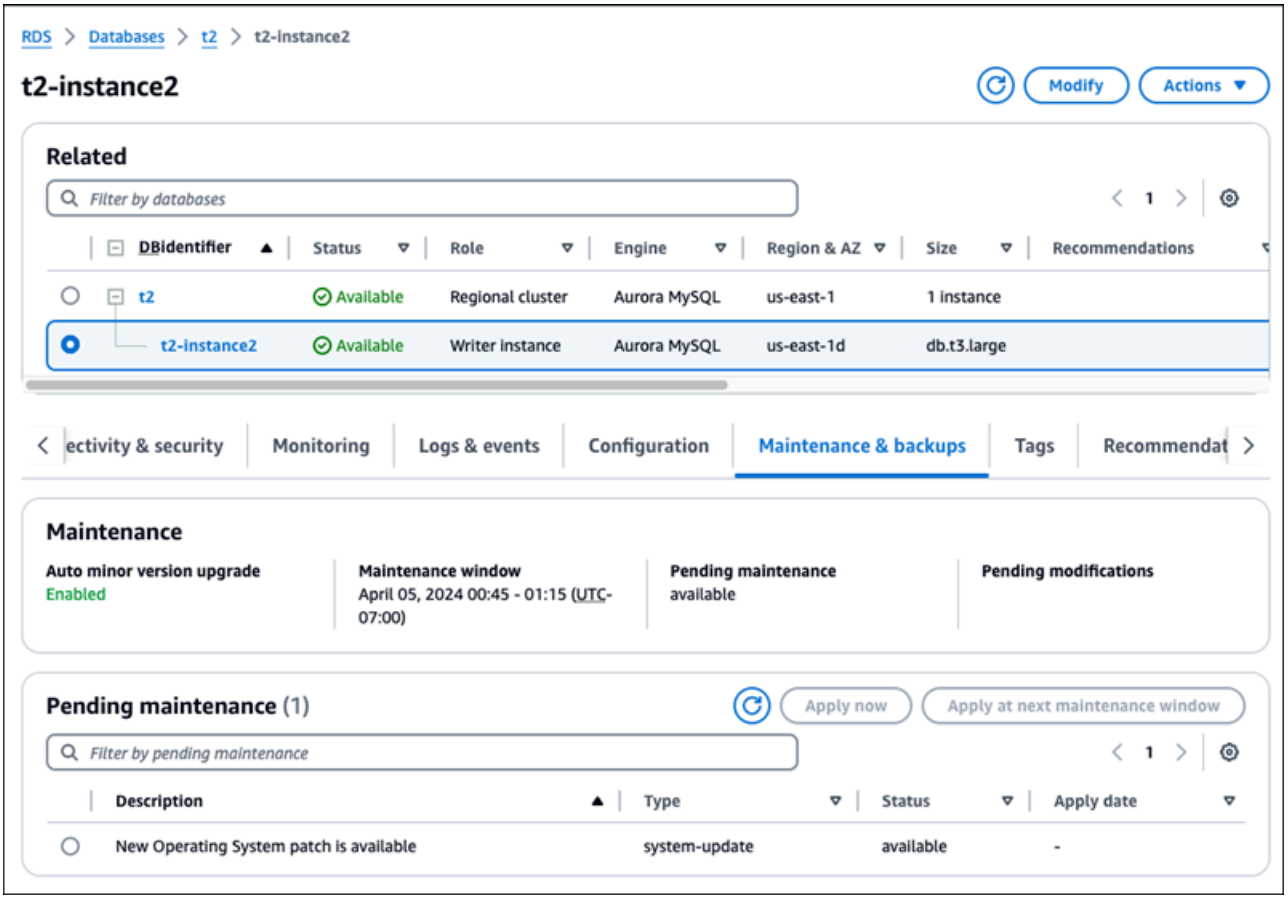
Console

Como obter informações de atualização usando o Console de gerenciamento da AWS

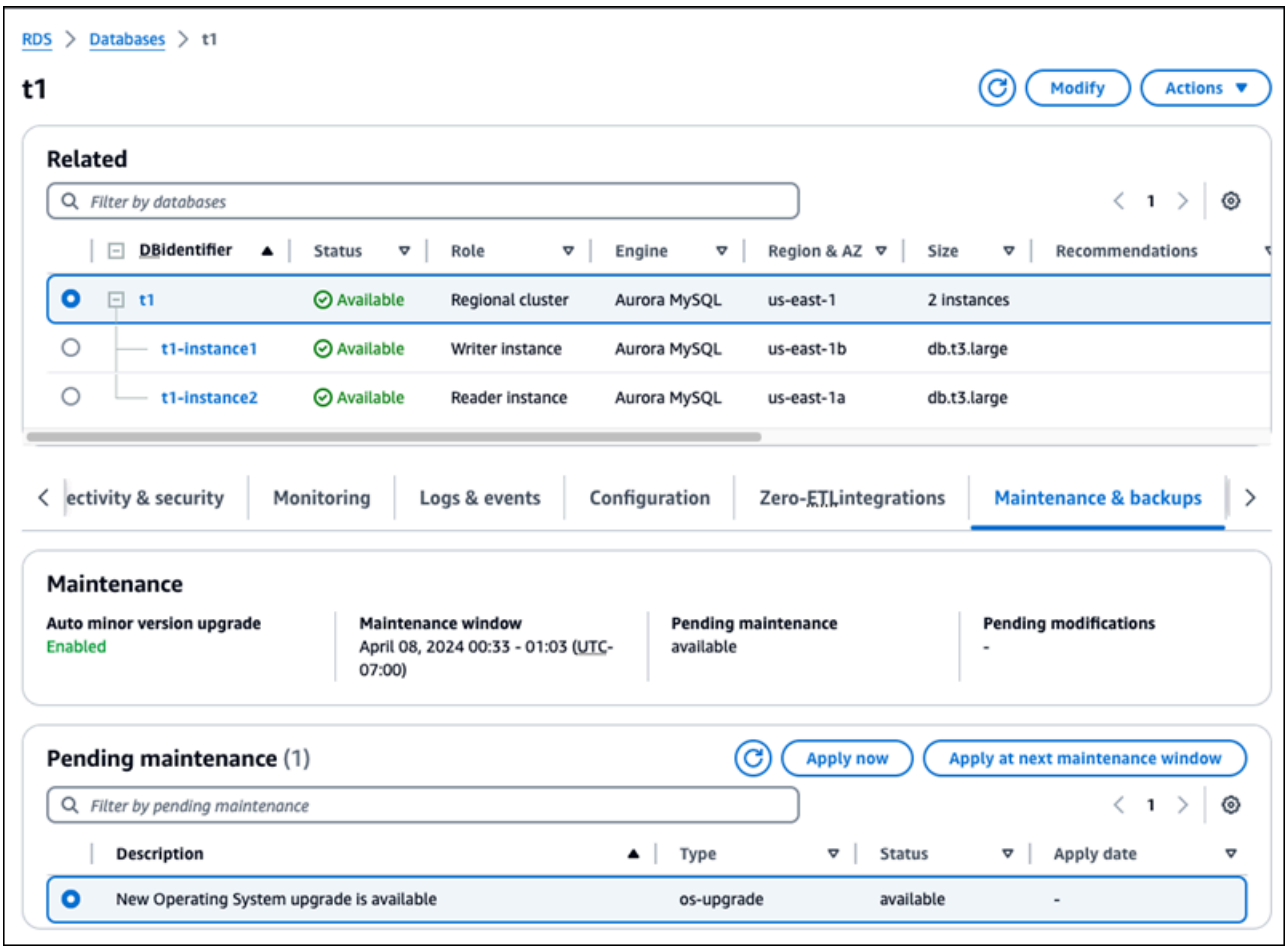
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e, depois, a instância de banco de dados.
3. Escolha Maintenance & backups (Manutenção e backups).
4. Na seção Manutenção pendente, encontre a atualização do sistema operacional e confira o valor Descrição.

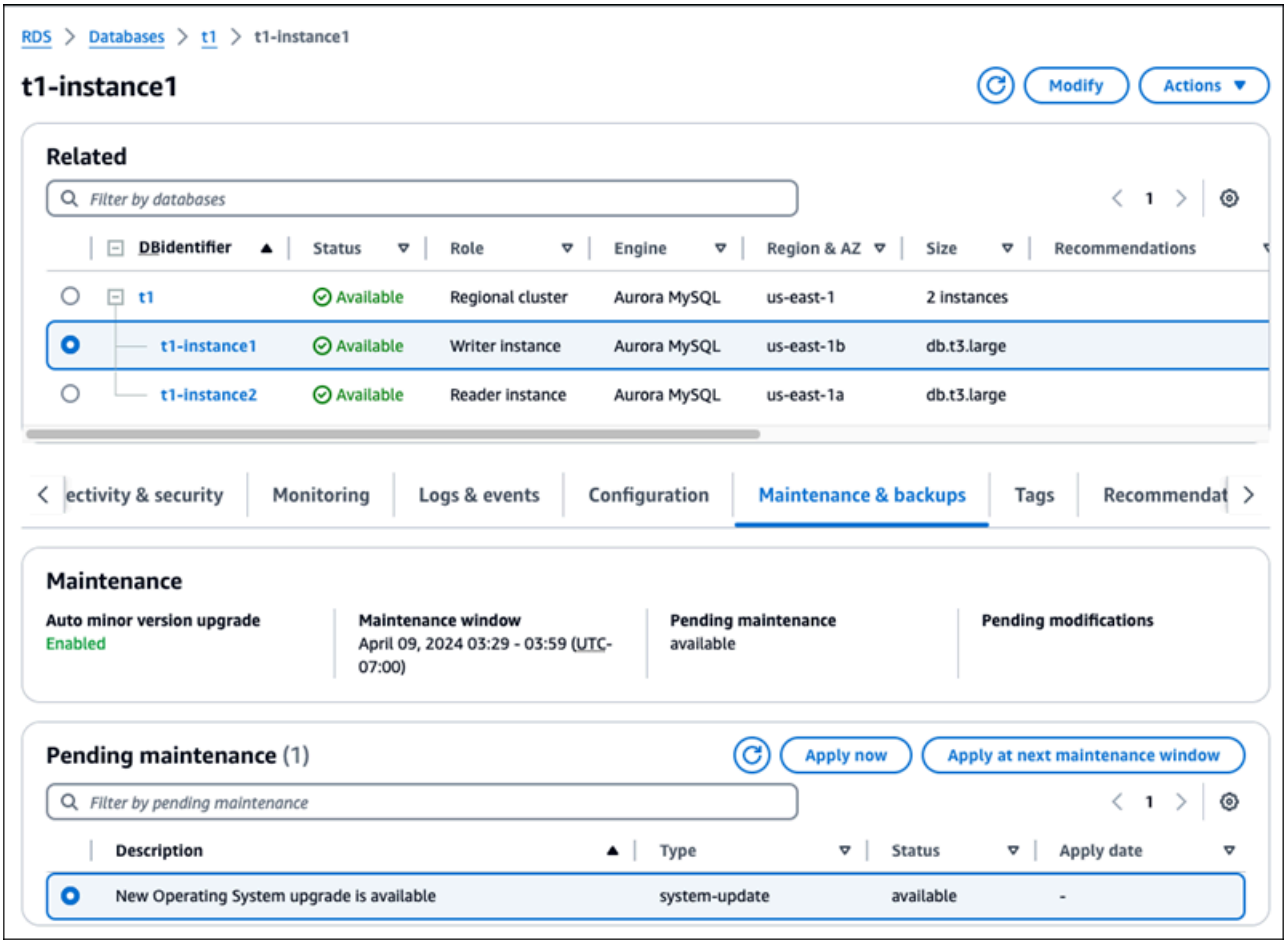
As imagens a seguir mostram um cluster de banco de dados com uma instância de banco de dados de gravador que tem um patch de sistema operacional disponível.

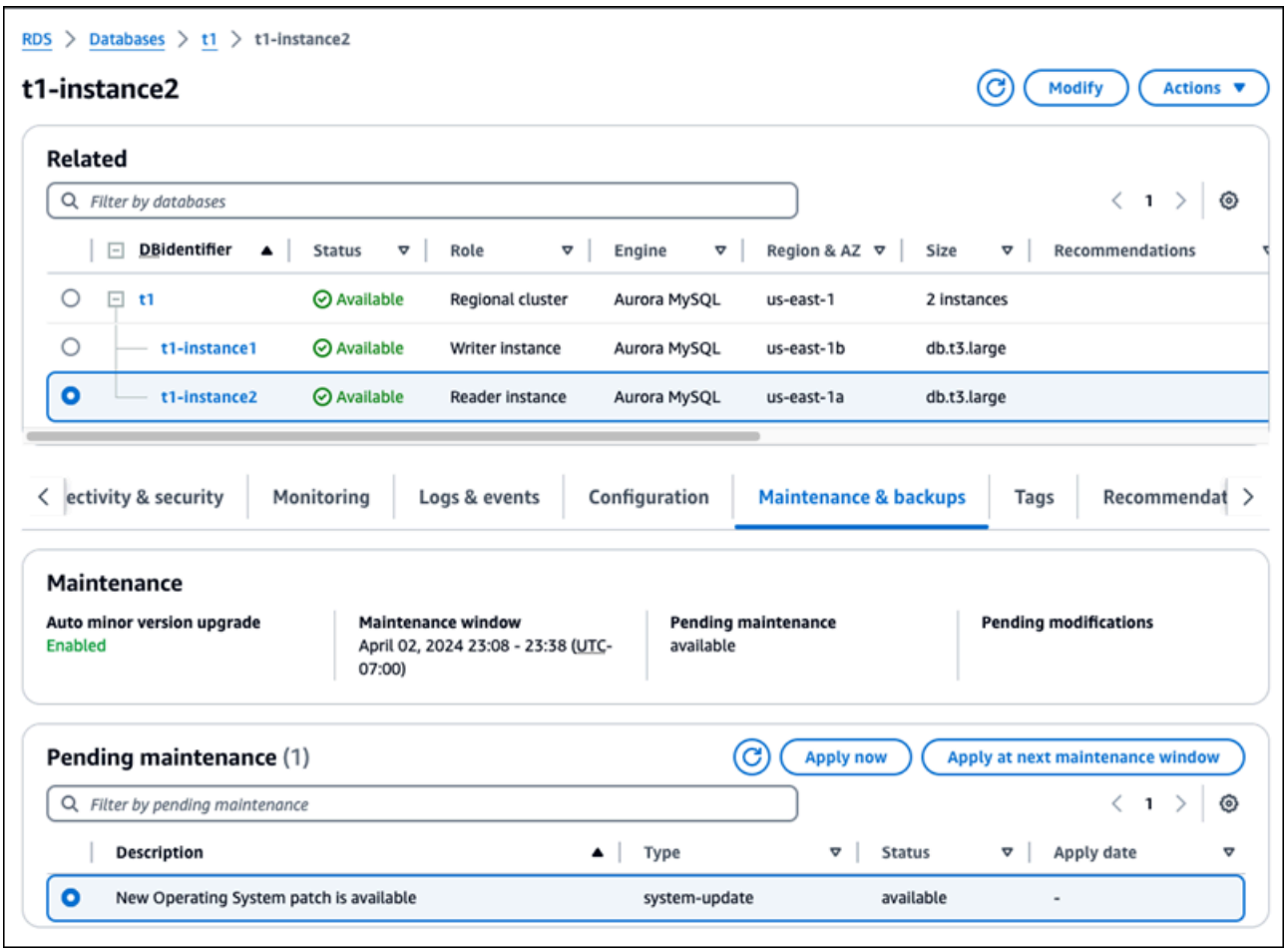




As imagens a seguir mostram um cluster de banco de dados com uma instância de banco de dados de gravador e uma instância de banco de dados de leitor. A instância do gravador tem uma atualização obrigatória do sistema operacional disponível. A instância do leitor tem um patch do sistema operacional disponível.


RDS > Databases > t1
t1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 08, 2024 00:33 - 01:03 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
os-upgrade
available
-


RDS > Databases > t1 > t1-instance1
t1-instance1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Recommendat
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 09, 2024 03:29 - 03:59 (UTC-07:00)
Pending maintenance
available
Pending modifications
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
system-update
available
-



AWS CLI

Para obter informações de atualização da AWS CLI, use o comando [describe-pending-maintenance-actions](#).

```
aws rds describe-pending-maintenance-actions
```

A saída a seguir mostra uma atualização da distribuição do sistema operacional para um cluster de banco de dados e uma instância de banco de dados.

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:t3",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "os-upgrade",
          "Description": "New Operating System upgrade is available"
```

```
        }
      ]
    },
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:t3-instance1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "New Operating System upgrade is available"
        }
      ]
    },
  ]
}
```

A saída a seguir mostra um patch do sistema operacional para uma instância de banco de dados.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System patch is available"
    }
  ]
}
```

Disponibilidade de atualizações do sistema operacional

As atualizações do sistema operacional são específicas da versão do mecanismo de banco de dados e da classe de instância de banco de dados. Portanto, as instâncias de banco de dados recebem ou exigem atualizações em momentos diferentes. Quando uma atualização do sistema operacional estiver disponível para sua instância de banco de dados com base na versão do mecanismo e na classe de instância de banco de dados, essa atualização aparecerá no console. Ela também pode ser visualizada executando o comando describe-pending-maintenance-actions AWS CLI ou chamando a operação DescribePendingMaintenanceActions da API do RDS. Se houver uma atualização disponível para sua instância, você poderá atualizar o sistema operacional seguindo as instruções em Aplicar atualizações para um clusterde banco de dados.

Usar a política de implementação de atualização do AWS Organizations para atualizações automáticas de versões secundárias

O Aurora comporta a política de implementação de atualizações do AWS Organizations para gerenciar atualizações automáticas de versões secundárias em vários recursos de banco de dados e Contas da AWS. Essa política ajuda você a implementar uma estratégia de atualização controlada para seus clusters por meio de:

Como funciona a política de implementação de atualizações

Quando uma nova versão secundária do mecanismo se torna elegível para atualização automática, a política controla a sequência de atualização com base em ordens definidas:

- Os recursos marcados como [primeiros] (normalmente ambientes de desenvolvimento) se tornam elegíveis para atualizações durante suas janelas de manutenção.
- Depois de um tempo de incorporação designado, os recursos marcados como [segundo] se tornam elegíveis.
- Depois de outro tempo de incorporação designado, os recursos marcados como [últimos] (normalmente ambientes de produção) se tornam elegíveis.
- Monitorar o andamento da atualização por meio de notificações de integridade da AWS.

Você pode definir a ordem das atualizações da seguinte forma:

- Políticas em nível de conta: aplicam-se a todos os recursos elegíveis em contas específicas.
- Tags de recursos: aplicam-se a recursos específicos com base em tags.

> **Note**
>
> Os recursos não configurados com uma política de atualização ou excluídos da política recebem automaticamente uma ordem de atualização de [segundo].

Pré-requisitos

- Sua Conta da AWS deve fazer parte de uma organização em Organizations com a política de implementação de atualizações habilitada.
- Habilitar atualizações automáticas da versão secundária de clusters

- As tags não são estritamente necessárias para a política de implementação de atualizações. Se quiser definir ordens de atualização específicas para diferentes ambientes (por exemplo, desenvolvimento, teste, controle de qualidade e produção), você pode usar tags. Se você não incluir configurações de tag em sua política, todos os recursos dela seguirão a ordem de atualização padrão. No caso de recursos do Aurora, somente tags em nível de cluster são usadas para a política de implementação de atualizações, mesmo se você tiver tags definidas em nível de instância.

Como marcar seus recursos

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o cluster que você deseja marcar.
4. Escolha Actions e, em seguida, escolha Manage tags.
5. Escolha Adicionar Tag.
6. Insira a chave da tag (por exemplo, “Ambiente”) e o valor (por exemplo, “Desenvolvimento”).
7. Escolha Adicionar tag e, depois, Salvar.

Você também pode adicionar tags usando a AWS CLI:

```
aws rds add-tags-to-resource \
    --resource-name arn:aws:rds:region:account-number:cluster:cluster-name \
    --tags Key=Environment,Value=Development
```

Ordem e fases de atualização

A política de implementação de atualizações aceita três ordens de atualização:

- [primeiro]: normalmente usado para ambientes de desenvolvimento ou teste.
- [segundo]: normalmente usado para ambientes de controle de qualidade. Ordem padrão dos recursos se a política não estiver configurada especificamente.
- [último]: normalmente reservado para ambientes de produção.

Quando uma nova versão secundária do mecanismo se torna elegível para atualização automática:

- Os recursos com ordem de atualização [primeiro] se tornam elegíveis para atualizações durante as janelas de manutenção configuradas.
- Após um tempo de incorporação, os recursos com ordem de atualização [primeiro] se tornam elegíveis para atualizações durante as janelas de manutenção.
- Após outro tempo de incorporação, os recursos com ordem de atualização [último] se tornam elegíveis para atualizações durante as janelas de manutenção.
- A campanha automática de atualização de versão secundária é encerrada após todos os recursos elegíveis com ordens de atualização [primeiro], [segundo] e [último] terem sido atualizados, ou quando a campanha atinge a data de término programada, o que ocorrer primeiro.

> **Note**
>
> Todas as atualizações automáticas de versões secundárias são realizadas durante a janela de manutenção configurada de cada cluster para minimizar o possível impacto em suas aplicações.

Observabilidade

Integridade e monitoramento da AWS

Você recebe notificações de integridade da AWS:

- Antes de iniciar uma campanha de atualização automática de versão secundária.
- Entre cada transição de fase para ajudar a acompanhar e monitorar o andamento da atualização.
- Atualizações de andamento mostrando o número de recursos atualizados em sua frota no console de integridade da AWS.

Notificações de eventos do Amazon RDS:

- Notificações de recursos habilitados para atualizações automáticas de versões secundárias, incluindo:
  - Quando seu recurso se torna elegível para atualização com base em sua ordem de atualização ([primeiro], [segundo] ou [último]).
  - Cronograma de atualizações programadas durante a janela de manutenção
  - Status de conclusão e início de atualização de banco de dados individual

- Assine esses eventos por meio do Amazon EventBridge0 para monitoramento automatizado.

Considerações

Algumas questões a serem levadas em consideração:

- A política se aplica a todas as futuras campanhas automáticas de atualização de versões secundárias, incluindo alterações de política feitas durante campanhas ativas.
- Se você participar de uma campanha de atualização em andamento, seus recursos seguirão a ordem atual de atualização em execução e não esperarão por uma política configurada.
- Os recursos não configurados com uma política de atualização ou excluídos da política recebem automaticamente uma ordem de atualização de [segundo].
- A política fornece períodos de validação entre as fases de atualização antes de prosseguir para a próxima fase.
- As alterações na política ou nas tags de recursos exigem tempo para serem propagadas antes que a nova ordem de atualização seja aplicada.
- A política se aplica somente aos recursos do Aurora com atualizações automáticas de versões secundárias habilitadas.
- Se você detectar um problema em um ambiente, poderá desativar as atualizações automáticas de versões secundárias para ambientes subsequentes ou usar o período de validação para resolver problemas antes que as atualizações prossigam para a próxima ordem de atualização.

Para acessar mais informações sobre como marcar recursos do RDS, consulte Marcar recursos do Amazon Aurora e do Amazon RDS. Para acessar instruções detalhadas sobre como configurar e usar a política de implementação de atualizações, consulte Introdução ao AWS Organizations no Guia do usuário do AWS Organizations.

Reinicializar um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora

Talvez seja necessário reinicializar seu cluster de banco de dados ou algumas instâncias dentro do cluster, geralmente por motivos de manutenção. Por exemplo, suponha que você modifique os parâmetros dentro de um grupo de parâmetros ou associe um grupo de parâmetros diferente ao cluster. Nesses casos, você deve reinicializar o cluster para que as alterações entrem em vigor. Da mesma forma, você pode reinicializar uma ou mais instâncias de banco de dados de leitor dentro do cluster. Você pode organizar as operações de reinicialização para instâncias individuais para minimizar o tempo de inatividade de todo o cluster.

O tempo necessário para reinicializar cada instância de banco de dados no cluster depende da atividade do banco de dados no momento da reinicialização. Também depende do processo de recuperação do seu mecanismo de banco de dados específico. Se for prático, reduza a atividade do banco de dados nessa instância específica antes de iniciar o processo de reinicialização. Isso pode reduzir o tempo necessário para reiniciar o banco de dados.

Você só pode reinicializar cada instância de banco de dados em seu cluster quando ela estiver no estado disponível. Uma instância de banco de dados pode estar indisponível por vários motivos. Isso inclui o estado do cluster que está sendo interrompido, uma modificação sendo aplicada à instância e uma ação da janela de manutenção, como uma atualização de versão.

Reiniciar uma instância de banco de dados reinicia o processo do mecanismo de banco de dados. Reinicializar uma instância de banco de dados resulta em uma interrupção momentânea, durante a qual o status da instância de banco de dados é definido como rebooting (reinicialização).

> **Note**
>
> Se uma instância de banco de dados não estiver usando as últimas alterações de seu grupo de parâmetros de banco de dados associado, o Console de gerenciamento da AWS mostrará o grupo de parâmetros de banco de dados com um status de pending-reboot (reinicialização pendente). O status pending-reboot (reinicialização pendente) do grupo de parâmetros não resultará em uma reinicialização automática durante a próxima janela de manutenção. Para aplicar as alterações de parâmetro mais recentes a essa instância de banco de dados, reinicialize-a manualmente. Para obter mais informações sobre parameter groups, consulte Grupos de parâmetros para Amazon Aurora.

Tópicos

- Reinicializar uma instância de banco de dados em um cluster do Aurora
- Reinicializar um cluster do Aurora com disponibilidade de leitura
- Reinicializar um cluster do Aurora sem disponibilidade de leitura
- Verificar o tempo de atividade para clusters e instâncias do Aurora
- Exemplos de operações de reinicialização do Aurora

Reinicializar uma instância de banco de dados em um cluster do Aurora

Esse procedimento é a operação mais importante que você realiza ao executar reinicializações com o Aurora. Muitos dos procedimentos de manutenção envolvem a reinicialização de uma ou mais instâncias de banco de dados do Aurora em uma ordem específica.

Console

Para reiniciar uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e a instância de banco de dados que você deseja reiniciar.
3. Em Actions (Ações), escolha Reboot (Reiniciar).

   A página Reboot DB Instance (Reinicializar instância de banco de dados) é exibida.
4. Escolha Reboot (Reinicializar) para reinicializar sua instância de banco de dados.

   Ou escolha Cancel (Cancelar).

AWS CLI

Para reinicializar uma instância de banco de dados usando a AWS CLI, chame o comando `reboot-db-instance`.

Example

Para Linux, macOS ou Unix:

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier mydbinstance
```

Para Windows:

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

API do RDS

Para reinicializar uma instância de banco de dados usando a API do Amazon RDS, chame a operação `RebootDBInstance`.

Reinicializar um cluster do Aurora com disponibilidade de leitura

Com o recurso de disponibilidade de leitura, é possível reinicializar a instância do gravador do cluster do Aurora sem reinicializar as instâncias do leitor no cluster de banco de dados primário ou secundário. Isso pode ajudar a manter a alta disponibilidade do cluster para operações de leitura enquanto você reinicializa a instância do gravador. Você pode reinicializar as instâncias do leitor mais tarde, em um cronograma que seja conveniente para você. Por exemplo, em um cluster de produção, você pode reinicializar as instâncias do leitor uma de cada vez, começando somente após a conclusão da reinicialização da instância primária. Para cada instância de banco de dados que você reinicializar, siga o procedimento em Reinicializar uma instância de banco de dados em um cluster do Aurora.

O recurso de disponibilidade de leitura de clusters de banco de dados primários está disponível no Aurora MySQL versão 2.10 e posterior. A disponibilidade de leitura para clusters de banco de dados secundários está disponível no Aurora MySQL versão 3.06 e posterior.

Esse atributo está disponível para as seguintes versões do Aurora PostgreSQL:

- Versão 15.2 e versões 15 posteriores
- Versão 14.7 e versões 14 posteriores
- Versão 13.10 e versões 13 posteriores
- Versão 12.14 e versões 12 posteriores

Para obter mais informações sobre o atributo de disponibilidade de leitura no Aurora PostgreSQL, consulte Melhorar a disponibilidade de leitura das réplicas do Aurora.

Antes desse recurso, a reinicialização da instância principal fazia com que cada instância do leitor fosse reinicializada ao mesmo tempo. Se o cluster do Aurora estiver executando uma versão mais antiga, use o procedimento de reinicialização em Reinicializar um cluster do Aurora sem disponibilidade de leitura.

> **Note**
>
> A mudança no comportamento de reinicialização nos clusters de banco de dados do Aurora com disponibilidade de leitura é diferente para banco de dados globais do Aurora no Aurora MySQL versões anteriores a 3.06. Se você reinicializar a instância do gravador para o cluster principal em um banco de dados global do Aurora, as instâncias do leitor no cluster principal permanecerão disponíveis. No entanto, as instâncias de banco de dados em qualquer cluster secundário são reinicializadas ao mesmo tempo.
>
> Uma versão limitada do recurso avançado de disponibilidade de leitura é compatível com o bancos de dados globais do Aurora para o Aurora PostgreSQL versões 12.16, 13.12, 14.9, 15.4 e posteriores.

Você reinicializa frequentemente o cluster depois de fazer alterações nos grupos de parâmetros do cluster. Você faz alterações nos parâmetros seguindo os procedimentos em Grupos de parâmetros para Amazon Aurora. Suponha que você reinicialize a instância de banco de dados do gravador em um cluster do Aurora para aplicar alterações aos parâmetros do cluster. Algumas ou todas as instâncias de banco de dados do leitor podem continuar usando as configurações de parâmetros antigas. No entanto, as diferentes configurações de parâmetros não afetam a integridade dos dados do cluster. Todos os parâmetros de cluster que afetam a organização dos arquivos de dados são usados apenas pela instância de banco de dados do gravador.

Por exemplo, em um cluster do Aurora MySQL, você pode atualizar parâmetros de cluster, como `binlog_format` e `innodb_purge_threads`, na instância do gravador antes das instâncias do leitor. Somente a instância do gravador está gravando logs binários e limpando registros de desfazer. Para parâmetros que alteram a forma como as consultas interpretam instruções SQL ou a saída da consulta, talvez seja necessário ter cuidado para reinicializar as instâncias do leitor imediatamente. Você faz isso para evitar comportamentos inesperados de aplicações durante consultas. Por exemplo, suponha que você altere o parâmetro `lower_case_table_names` e reinicie a instância do gravador. Nesse caso, as instâncias do leitor talvez não consigam acessar uma tabela recém-criada até que todas sejam reinicializadas.

Para obter uma lista de todos os parâmetros do cluster do Aurora MySQL, consulte Parâmetros no nível do cluster.

Para obter uma lista de todos os parâmetros de cluster do Aurora PostgreSQL, consulte Parâmetros no nível do cluster do Aurora PostgreSQL.

> **Tip**
>
> O Aurora MySQL ainda pode reinicializar algumas das instâncias do leitor junto com a instância do gravador se o cluster estiver processando uma workload com alta taxa de transferência.
> A redução no número de reinicializações também se aplica durante as operações de failover. O Aurora MySQL reinicia somente a instância de banco de dados do gravador e o destino de failover durante um failover. Outras instâncias de banco de dados do leitor no cluster permanecem disponíveis para continuar processando consultas por meio de conexões com o endpoint do leitor. Assim, você pode melhorar a disponibilidade durante um failover com mais de uma instância de banco de dados de leitor em um cluster.

Reinicializar um cluster do Aurora sem disponibilidade de leitura

Sem o atributo de disponibilidade de leitura, você reinicializa um cluster de banco de dados do Aurora inteiro ao reinicializar a instância de banco de dados do leitor desse cluster. Para fazer isso, siga o procedimento em Reinicializar uma instância de banco de dados em um cluster do Aurora.

A reinicialização da instância de banco de dados do gravador também inicia uma reinicialização para cada instância de banco de dados do leitor no cluster. Dessa forma, todas as alterações de parâmetros em todo o cluster são aplicadas a todas as instâncias de banco de dados ao mesmo tempo. No entanto, a reinicialização de todas as instâncias de banco de dados causa uma breve interrupção para o cluster. As instâncias de banco de dados do leitor permanecem indisponíveis até que a instância de banco de dados do gravador termine a reinicialização e fique disponível

Esse comportamento de reinicialização aplica-se a todos os clusters de banco de dados criados no Aurora MySQL versão 2.09 e anteriores.

Para o Aurora PostgreSQL, esse comportamento se aplica às seguintes versões:

- Versão 14.6 e versões 14 anteriores
- Versão 13.9 e versões 13 anteriores

- Versão 12.13 e versões 12 anteriores
- Todas as versões 11 do PostgreSQL

No console do RDS, a instância de banco de dados do gravador tem o valor Writer na colunaFunction (Função) na página Databases (Bancos de dados). Na CLI do RDS, a saída do comando `describe-db-clusters` inclui uma seção `DBClusterMembers`. O elemento `DBClusterMembers` que representa a instância de banco de dados do gravador tem um valor `true` para o campo `IsClusterWriter`.

> **Important**
>
> Com o atributo de disponibilidade de leitura, o comportamento de reinicialização é diferente no Aurora MySQL e Aurora PostgreSQL: as instâncias de banco de dados do leitor geralmente permanecem disponíveis enquanto você reinicia a instância do gravador. Em seguida, você pode reinicializar as instâncias do leitor em um momento conveniente. Você pode reinicializar as instâncias do leitor em uma programação escalonada se quiser que algumas instâncias do leitor estejam sempre disponíveis. Para ter mais informações, consulte Reinicializar um cluster do Aurora com disponibilidade de leitura.

Verificar o tempo de atividade para clusters e instâncias do Aurora

Você pode verificar e monitorar o período de tempo desde a última reinicialização para cada instância de banco de dados em seu cluster do Aurora. A métrica `EngineUptime` do Amazon CloudWatch informa o número de segundos desde a última vez que uma instância de banco de dados foi iniciada. Você pode examinar essa métrica em um momento para descobrir o tempo de atividade da instância de banco de dados. Você também pode monitorar essa métrica ao longo do tempo para detectar quando a instância é reinicializada.

Você também pode examinar a métrica `EngineUptime` no nível do cluster. As dimensões `Minimum` e `Maximum` relatam os menores e maiores valores de tempo de atividade para todas as instâncias de banco de dados no cluster. Para verificar o momento mais recente em que qualquer instância do leitor em um cluster foi reinicializada ou reiniciada por outro motivo, monitore a métrica no nível do cluster usando a dimensão `Minimum`. Para verificar qual instância no cluster passou mais tempo sem uma reinicialização, monitore a métrica no nível do cluster usando a dimensão `Maximum`. Por exemplo, talvez você queira confirmar que todas as instâncias de banco de dados no cluster foram reinicializadas após uma alteração de configuração.

> **Tip**
>
> Para monitoramento de longo prazo, recomendamos monitorar a métrica `EngineUptime` para instâncias individuais em vez de no nível do cluster. A métrica `EngineUptime` no nível do cluster é definida como zero quando uma nova instância de banco de dados é adicionada ao cluster. Essas alterações de cluster podem ocorrer como parte das operações de manutenção e escalabilidade, como as realizadas pelo Auto Scaling.

Os exemplos da CLI a seguir mostram como examinar a métrica `EngineUptime` para as instâncias do gravador e do leitor em um cluster. Os exemplos usam um cluster chamado `tpch100g`. Este cluster tem uma instância de banco de dados do gravador `instance-1234`. Ele também tem duas instâncias de banco de dados de leitor, `instance-7448` e `instance-6305`.

Primeiro, o comando `reboot-db-instance` reinicializa uma das instâncias do leitor. O comando `wait` aguarda até que a instância termine a reinicialização.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-6305
```

O comando `get-metric-statistics` do CloudWatch examina a métrica `EngineUptime` nos últimos cinco minutos em intervalos de um minuto. O tempo de atividade da instância `instance-6305` é redefinido para zero e começa a contar novamente. Esse exemplo da AWS CLI para Linux usa substituição de variáveis `$( )` para inserir os carimbos de data/hora apropriados nos comandos da CLI. Ele também usa o comando `sort` para Linux para ordenar a saída no momento em que a métrica foi coletada. Esse valor de carimbo de data/hora é o terceiro campo em cada linha de saída.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 231.0 2021-03-16T18:19:00+00:00 Seconds
```

```
DATAPOINTS 291.0 2021-03-16T18:20:00+00:00 Seconds
DATAPOINTS 351.0 2021-03-16T18:21:00+00:00 Seconds
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
```

O tempo de atividade mínimo para o cluster é redefinido para zero porque uma das instâncias no cluster foi reinicializada. O tempo de atividade máximo para o cluster não é redefinido porque pelo menos uma das instâncias de banco de dados no cluster permaneceu disponível.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Minimum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63099.0 2021-03-16T18:12:00+00:00 Seconds
DATAPOINTS 63159.0 2021-03-16T18:13:00+00:00 Seconds
DATAPOINTS 63219.0 2021-03-16T18:14:00+00:00 Seconds
DATAPOINTS 63279.0 2021-03-16T18:15:00+00:00 Seconds
DATAPOINTS 51.0 2021-03-16T18:16:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63389.0 2021-03-16T18:16:00+00:00 Seconds
DATAPOINTS 63449.0 2021-03-16T18:17:00+00:00 Seconds
DATAPOINTS 63509.0 2021-03-16T18:18:00+00:00 Seconds
DATAPOINTS 63569.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 63629.0 2021-03-16T18:20:00+00:00 Seconds
```

Em seguida, outro comando `reboot-db-instance` reinicializa a instância do gravador do cluster. Outro comando `wait` pausa até que a instância do gravador termine a reinicialização.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
  "DBInstanceIdentifier": "instance-1234",
  "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

Agora, a métrica `EngineUptime` para a instância do gravador mostra que a instância `instance-1234` foi reinicializada recentemente. A instância do leitor `instance-6305` também foi reinicializada automaticamente junto com a instância do gravador. Este cluster está executando o Aurora MySQL 2.09, o que não mantém as instâncias do leitor em execução à medida que a instância do gravador é reinicializada.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-1234 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63749.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 63809.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 63869.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 41.0 2021-03-16T18:25:00+00:00 Seconds
DATAPOINTS 101.0 2021-03-16T18:26:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 531.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 49.0 2021-03-16T18:26:00+00:00 Seconds
```

Exemplos de operações de reinicialização do Aurora

Os exemplos do Aurora MySQL a seguir mostram diferentes combinações de operações de reinicialização para instâncias de banco de dados de leitor e gravador em um cluster de banco de dados do Aurora. Após cada reinicialização, as consultas SQL demonstram o tempo de atividade das instâncias no cluster.

Tópicos

- Localizar as instâncias do gravador e do leitor para um cluster do Aurora
- Reinicializar uma única instância do leitor
- Reinicializar a instância do gravador

- Reinicializar o gravador e os leitores independentemente
- Aplicando uma alteração de parâmetro de cluster a um cluster do Aurora MySQL versão 2.10

Localizar as instâncias do gravador e do leitor para um cluster do Aurora

Em um cluster do Aurora MySQL com várias instâncias de banco de dados, é importante saber qual delas é o gravador e quais são os leitores. As instâncias do gravador e do leitor também podem mudar de função quando ocorre uma operação de failover. Assim, é melhor executar uma verificação como a seguinte antes de fazer qualquer operação que exija uma instância de gravador ou leitor. Nesse caso, os valores de `False` para `IsClusterWriter` identificam as instâncias do leitor `instance-6305` e `instance-7448`. O valor `True` identifica a instância do gravador, `instance-1234`.

```
$ aws rds describe-db-clusters --db-cluster-id tpch100g \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:     tpch100g
Instance:    instance-6305    False
Instance:    instance-7448    False
Instance:    instance-1234    True
```

Antes de iniciarmos os exemplos de reinicialização, a instância do gravador tem um tempo de atividade de aproximadamente uma semana. A consulta SQL neste exemplo mostra uma maneira específica do MySQL de verificar o tempo de atividade. Você pode usar essa técnica em uma aplicação de banco de dados. Para outra técnica que usa a AWS CLI e funciona para os dois mecanismos do Aurora, consulte Verificar o tempo de atividade para clusters e instâncias do Aurora.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-08 17:49:06.000000 | 174h 42m|
+----------------------------+---------+
```

Reinicializar uma única instância do leitor

Esse exemplo reinicia uma das instâncias de banco de dados do leitor. Talvez essa instância tenha sido sobrecarregada por uma consulta enorme ou por muitas conexões simultâneas. Ou talvez tenha ficado atrás da instância do escritor por causa de um problema de rede. Depois de iniciar a operação de reinicialização, o exemplo usa um comando `wait` para pausar até que a instância fique disponível. A essa altura, a instância tem um tempo de atividade de alguns minutos.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-6305
$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:35:02.000000 | 00h 03m |
+----------------------------+---------+
```

A reinicialização da instância do leitor não afetou o tempo de atividade da instância do gravador. Ele ainda tem um tempo de atividade de cerca de uma semana.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+----------+
| Last Startup               | Uptime   |
+----------------------------+----------+
| 2021-03-08 17:49:06.000000 | 174h 49m |
+----------------------------+----------+
```

Reinicializar a instância do gravador

Este exemplo reinicia a instância do gravador. Esse cluster está executando o Aurora MySQL versão 2.09. Como a versão do Aurora MySQL é inferior à 2.10, a reinicialização da instância do gravador também reinicializa qualquer instância de leitor no cluster.

Um comando `wait` pausa até que a reinicialização seja concluída. Agora, o tempo de atividade dessa instância é redefinido para zero. É possível que uma operação de reinicialização possa levar momentos substancialmente diferentes para instâncias de banco de dados de gravador e leitor. As instâncias de banco de dados do gravador e do leitor executam diferentes tipos de operações de limpeza, dependendo de suas funções.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-1234",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-1234
$ mysql -h instance-1234.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:27.000000 | 00h 00m |
+----------------------------+---------+
```

Após a reinicialização da instância de banco de dados do gravador, ambas as instâncias de banco de dados do leitor também têm seu tempo de atividade redefinido. A reinicialização da instância do gravador fazia com que as instâncias do leitor fossem reinicializadas também. Esse comportamento se aplica a clusters do Aurora PostgreSQL e a clusters do Aurora MySQL antes da versão 2.10.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
```

```
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:35.000000 | 00h 00m |
+----------------------------+---------+

$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:33.000000 | 00h 01m |
+----------------------------+---------+
```

Reinicializar o gravador e os leitores independentemente

Esses próximos exemplos mostram um cluster que executa o Aurora MySQL versão 2.10. Nessa versão do Aurora MySQL e nas posteriores, você pode reinicializar a instância do gravador sem causar reinicializações para todas as instâncias do leitor. Dessa forma, suas aplicações com uso intensivo de consultas não sofrem nenhuma interrupção quando você reinicializa a instância do gravador. Você pode reinicializar as instâncias do leitor mais tarde. Você pode fazer essas reinicializações em um momento de baixo tráfego de consulta. Você também pode reinicializar as instâncias do leitor uma de cada vez. Dessa forma, pelo menos uma instância do leitor está sempre disponível para o tráfego de consulta da aplicação.

O exemplo a seguir usa um cluster chamado `cluster-2393`, executando a versão `5.7.mysql_aurora.2.10.0` do Aurora MySQL. Esse cluster tem uma instância de gravador chamada `instance-9404` e três instâncias de leitor chamadas `instance-6772`, `instance-2470` e `instance-5138`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:        cluster-2393
Instance:       instance-5138        False
Instance:       instance-2470        False
```

```
Instance:       instance-6772        False
Instance:       instance-9404        True
```

Verificar o valor `uptime` de cada instância de banco de dados por meio do comando `mysql` mostra que cada uma tem aproximadamente o mesmo tempo de atividade. Por exemplo, aqui está o tempo de atividade para `instance-5138`.

```
mysql> SHOW GLOBAL STATUS LIKE 'uptime';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Uptime        | 3866  |
+---------------+-------+
```

Ao usar o CloudWatch, podemos obter as informações de tempo de atividade correspondentes sem realmente fazer login nas instâncias. Dessa forma, um administrador pode monitorar o banco de dados, mas não pode exibir ou alterar nenhum dado da tabela. Nesse caso, especificamos um período de tempo que abrange cinco minutos e verificamos o valor do tempo de atividade a cada minuto. Os valores crescentes de tempo de atividade demonstram que as instâncias não foram reiniciadas durante esse período.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4648.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4708.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4768.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4828.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4888.0 2021-03-17T23:46:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4315.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4375.0 2021-03-17T23:43:00+00:00 Seconds
```

```
DATAPOINTS 4435.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4495.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4555.0 2021-03-17T23:46:00+00:00 Seconds
```

Agora reinicializamos uma das instâncias do leitor, `instance-5138`. Esperamos que a instância fique disponível novamente após a reinicialização. Agora, monitorar o tempo de atividade durante um período de cinco minutos mostra que o tempo de atividade foi redefinido para zero durante esse tempo. O valor de tempo de atividade mais recente foi medido cinco segundos após o término da reinicialização.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4500.0 2021-03-17T23:46:00+00:00 Seconds
DATAPOINTS 4560.0 2021-03-17T23:47:00+00:00 Seconds
DATAPOINTS 4620.0 2021-03-17T23:48:00+00:00 Seconds
DATAPOINTS 4680.0 2021-03-17T23:49:00+00:00 Seconds
DATAPOINTS  5.0 2021-03-17T23:50:00+00:00 Seconds
```

Em seguida, realizamos uma reinicialização para a instância do gravador, `instance-9404`. Comparamos os valores de tempo de atividade para a instância do gravador e uma das instâncias do leitor. Ao fazer isso, podemos ver que a reinicialização do escritor não causou uma reinicialização para os leitores. Nas versões anteriores a Aurora MySQL 2.10, os valores de tempo de atividade para todos os leitores seriam redefinidos ao mesmo tempo que o gravador.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
{
  "DBInstanceIdentifier": "instance-9404",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 371.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 431.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 491.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 551.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS  37.0  2021-03-18T00:01:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5215.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 5275.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 5335.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 5395.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS 5455.0 2021-03-18T00:01:00+00:00 Seconds
```

Para garantir que todas as instâncias do leitor tenham todas as mesmas alterações nos parâmetros de configuração que a instância do gravador, reinicie todas as instâncias do leitor após o gravador. Este exemplo reinicia todos os leitores e aguarda até que todos eles estejam disponíveis antes de prosseguir.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6772
{
  "DBInstanceIdentifier": "instance-6772",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-5138
```

```
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}

$ aws rds wait db-instance-available --db-instance-id instance-6772
$ aws rds wait db-instance-available --db-instance-id instance-2470
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

Agora podemos ver que a instância de banco de dados do gravador tem o maior tempo de atividade. O valor do tempo de atividade dessa instância aumentou de forma constante durante todo o período de monitoramento. As instâncias de banco de dados do leitor foram todas reinicializadas após o leitor. Podemos ver o ponto dentro do período de monitoramento em que cada leitor foi reinicializado e seu tempo de atividade foi redefinido para zero.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 457.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 517.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 577.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 637.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS  697.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-2470 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5819.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS  35.0  2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 95.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 155.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 215.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
```

```
  --output text | sort -k 3
EngineUptime
DATAPOINTS 1085.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 1145.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 1205.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS  49.0  2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 109.0 2021-03-18T00:12:00+00:00 Seconds
```

Aplicando uma alteração de parâmetro de cluster a um cluster do Aurora MySQL versão 2.10

O exemplo a seguir demonstra como aplicar uma alteração de parâmetro a todas as instâncias de banco de dados em seu cluster do Aurora MySQL 2.10. Com essa versão do Aurora MySQL, você reinicializa a instância do gravador e todas as instâncias do leitor de forma independente.

O exemplo usa o parâmetro de configuração `lower_case_table_names` do MySQL para ilustração. Quando essa configuração de parâmetro é diferente entre as instâncias de banco de dados do gravador e do leitor, talvez uma consulta não consiga acessar uma tabela declarada com um nome maiúsculo ou misto. Ou se dois nomes de tabela diferirem apenas em termos de letras maiúsculas e minúsculas, uma consulta pode acessar a tabela errada.

Este exemplo mostra como determinar as instâncias do gravador e do leitor no cluster examinando o atributo `IsClusterWriter` de cada instância. O cluster é chamado `cluster-2393`. O cluster tem uma instância de gravador chamada `instance-9404`. As instâncias do leitor no cluster são chamadas `instance-5138` e `instance-2470`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
cluster-2393
instance-5138        False
instance-2470        False
instance-9404        True
```

Para demonstrar os efeitos da alteração do parâmetro `lower_case_table_names`, configuramos dois grupos de parâmetros de cluster de banco de dados. O grupo de parâmetros `lower-case-table-names-0` tem esse parâmetro definido como 0. O grupo de parâmetros `lower-case-table-names-1` tem esse grupo de parâmetros definido como 1.

```
$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-0' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-0
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-0",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-0"
    }
}

$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-1' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-1",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-1"
    }
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-0 \
  --parameters
 ParameterName=lower_case_table_names,ParameterValue=0,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-0"
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-1 \
    --parameters
 ParameterName=lower_case_table_names,ParameterValue=1,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-1"
}
```

O valor padrão de `lower_case_table_names` é 0. Com essa configuração de parâmetro, a tabela `foo` é distinta da tabela F00. Este exemplo verifica se o parâmetro ainda está em sua configuração padrão. Em seguida, o exemplo cria três tabelas que diferem apenas em letras maiúsculas e minúsculas em seus nomes.

```
mysql> create database lctn;
Query OK, 1 row affected (0.07 sec)

mysql> use lctn;
Database changed
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> create table foo (s varchar(128));
mysql> insert into foo values ('Lowercase table name foo');

mysql> create table Foo (s varchar(128));
mysql> insert into Foo values ('Mixed-case table name Foo');

mysql> create table FOO (s varchar(128));
mysql> insert into FOO values ('Uppercase table name FOO');

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+---------------------------+
| s                         |
+---------------------------+
| Mixed-case table name Foo |
+---------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

Em seguida, associamos o grupo de parâmetros de banco de dados ao cluster para definir o parâmetro `lower_case_table_names` como 1. Essa alteração só entra em vigor após cada instância de banco de dados ser reinicializada.

```
$ aws rds modify-db-cluster --db-cluster-identifier cluster-2393 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
  "DBClusterIdentifier": "cluster-2393",
  "DBClusterParameterGroup": "lower-case-table-names-1",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.0"
}
```

A primeira reinicialização que fazemos é para a instância de banco de dados do gravador. Em seguida, esperamos que a instância se torne disponível novamente. Nesse ponto, nos conectamos ao endpoint do gravador e verificamos se a instância do gravador tem o valor do parâmetro alterado. O comando `SHOW TABLES` confirma que o banco de dados contém as três tabelas diferentes. No entanto, as consultas que se referem às tabelas chamadas `foo`, `Foo` ou `F00`, todas acessam a tabela com o nome todo em minúsculas, `foo`.

```
# Rebooting the writer instance
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

Agora, as consultas que usam o endpoint do cluster mostram os efeitos da alteração de parâmetro. Se o nome da tabela na consulta estiver em maiúsculas, minúsculas ou combinação de ambas, a instrução SQL acessa a tabela cujo nome está em minúsculas.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+

mysql> use lctn;
mysql> show tables;
+----------------+
| Tables_in_lctn |
+----------------+
```

```
| FOO            |
| Foo            |
| foo            |
+----------------+

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

O próximo exemplo mostra as mesmas consultas que a anterior. Nesse caso, as consultas usam o endpoint do leitor e são executadas em uma das instâncias de banco de dados do leitor. Essas instâncias ainda não foram reinicializadas. Assim, eles ainda têm a configuração original para o parâmetro `lower_case_table_names`. Isso significa que as consultas podem acessar cada uma das tabelas `foo`, `Foo` e `F00`.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> use lctn;

mysql> select * from foo;
+--------------------------+
```

```
| s                         |
+---------------------------+
| Lowercase table name foo |
+---------------------------+

mysql> select * from Foo;
+----------------------------+
| s                           |
+----------------------------+
| Mixed-case table name Foo |
+----------------------------+

mysql> select * from FOO;
+---------------------------+
| s                         |
+---------------------------+
| Uppercase table name FOO |
+---------------------------+
```

Em seguida, reinicializamos uma das instâncias do leitor e aguardamos que ela se torne disponível novamente.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-2470
```

Enquanto conectado ao endpoint da instância para `instance-2470`, uma consulta mostra que o novo parâmetro está em vigor.

```
mysql> select @@lower_case_table_names;
+---------------------------+
| @@lower_case_table_names |
+---------------------------+
|                         1 |
+---------------------------+
```

Nesse ponto, as duas instâncias do leitor no cluster estão sendo executadas com configurações `lower_case_table_names` diferentes. Assim, qualquer conexão com o endpoint do leitor

do cluster usa um valor para essa configuração que é imprevisível. É importante reinicializar imediatamente a outra instância do leitor para que ambas tenham configurações consistentes.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

O exemplo a seguir confirma que todas as instâncias do leitor têm a mesma configuração para o parâmetro `lower_case_table_names`. Os comandos verificam o valor da configuração `lower_case_table_names` em cada instância do leitor. Em seguida, o mesmo comando que usa o endpoint do leitor demonstra que cada conexão com o endpoint do leitor usa uma das instâncias do leitor, mas qual delas não é previsível.

```
# Check lower_case_table_names setting on each reader instance.

$ mysql -h instance-5138.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

$ mysql -h instance-2470.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-2470            |                        1 |
+--------------------------+--------------------------+

# Check lower_case_table_names setting on the reader endpoint of the cluster.

$ mysql -h cluster-2393.cluster-ro-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
```

```
+--------------------------+--------------------------+

# Run query on writer instance

$ mysql -h cluster-2393.cluster-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-9404            |                        1 |
+--------------------------+--------------------------+
```

Com a mudança de parâmetro aplicada em todos os lugares, podemos ver o efeito da configuração `lower_case_table_names=1`. Se a tabela é referida como `foo`, `Foo` ou `F00`, a consulta converte o nome para `foo` e acessa a mesma tabela em cada caso.

```
mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from F00;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

Fazer failover de um cluster de banco de dados do Amazon Aurora

Você poderá fazer failover manual de um cluster de banco de dados do Aurora quando, por exemplo, quiser substituir uma instância de banco de dados do gravador provisionada por uma instância de gravador do Aurora Serverless v2.

O Aurora fará failover para uma nova instância do banco de dados primário de uma das duas seguintes maneiras:

- Ao promover uma réplica existente do banco de dados de leitor à nova instância primária.
- Ao criar uma nova instância primária.

Se o cluster de banco de dados tiver uma ou mais réplicas do leitor, um leitor será promovido a instância primária durante um evento de falha. Para aumentar a disponibilidade do cluster de banco de dados, recomendamos criar pelo menos uma ou mais instâncias do leitor em duas ou mais zonas de disponibilidade diferentes. Para ter informações sobre o mecanismo de failover, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

É possível usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para fazer failover manual.

Console

Como fazer failover de um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Bancos de dados e, no cluster de banco de dados, escolha uma instância de banco de dados da qual você deseja fazer failover.
3. Em Actions (Ações), selecione Failover.

   A página de confirmação é exibida.
4. Escolha Failover.

   A página Bancos de dados mostra que o status do cluster de banco de dados é Em failover. O status retorna para Disponível quando o failover é concluído e os perfis da nova e da antiga instância do banco de dados primário são exibidos.

AWS CLI

Para fazer failover de um cluster de banco de dados usando a AWS CLI, chame o comando failover-db-cluster. Especifique os seguintes parâmetros:

- `--db-cluster-identifier`: o cluster de banco de dados do qual você deseja fazer failover.
- `--target-db-instance-identifier`: o nome da instância de banco de dados a ser promovida a instância do banco de dados primário.

Example

Para Linux, macOS ou Unix:

```
aws rds failover-db-cluster \
    --db-cluster-identifier mydbcluster \
    --target-db-instance-identifier mydbcluster-instance-2
```

Para Windows:

```
aws rds failover-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --target-db-instance-identifier mydbcluster-instance-2
```

API do RDS

Para modificar um cluster de banco de dados usando a API do Amazon RDS, chame a operação FailoverDBCluster. Especifique os seguintes parâmetros:

- DBClusterIdentifier
- TargetDBInstanceIdentifier

Excluir clusters de banco de dados e instâncias de banco de dados do Aurora

Será possível excluir um cluster de banco de dados do Aurora quando não for mais necessário. A exclusão do cluster remove o volume do cluster que contém todos os seus dados. Antes de excluir o cluster, você pode salvar um instantâneo de seus dados. Você pode restaurar o snapshot posteriormente e criar um cluster com os mesmos dados.

Você também pode excluir instâncias de banco de dados de um cluster e preservar o próprio cluster e os dados que ele contém. A exclusão de instâncias de banco de dados poderá ajudar a reduzir as cobranças se o cluster não estiver ocupado ou você não precisar da capacidade computacional de várias instâncias de banco de dados.

Tópicos

- Excluir um cluster de banco de dados do Aurora
- Proteção contra exclusão para clusters do Aurora.
- Excluir um cluster parado do Aurora
- Excluir clusters do Aurora MySQL que são réplicas de leitura
- O snapshot final ao excluir um cluster
- Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora

Excluir um cluster de banco de dados do Aurora

O Aurora não fornece um método de etapa única para excluir um cluster de banco de dados. Essa opção de design se destina a evitar que você perca dados ou deixe sua aplicação offline acidentalmente. As aplicações do Aurora geralmente são essenciais à missão e exigem alta disponibilidade. Assim, o Aurora facilita escalar a capacidade do cluster acrescentando e removendo instâncias de banco de dados. A remoção do cluster de banco de dados em si requer que você faça uma exclusão separada.

Use o procedimento geral a seguir para remover todas as instâncias de banco de dados de um cluster e então excluir o próprio cluster.

1. Exclua todas as instâncias do leitor no cluster. Use o procedimento em Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.

Se o cluster tiver alguma instância do leitor, excluir uma das instâncias apenas reduzirá a capacidade computacional do cluster. A exclusão das instâncias de leitor garante primeiro que o cluster permaneça disponível durante todo o procedimento e não execute operações de failover desnecessárias.

2. Exclua a instância do gravador do cluster. Use novamente o procedimento em Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.

   Quando você exclui as instâncias de banco de dados, o cluster e o volume do cluster associado permanecem mesmo após a exclusão de todas as respectivas instâncias.

3. Exclua o cluster de banco de dados.

   - Console de gerenciamento da AWS: selecione o cluster e escolha Excluir do menu Ações. Você pode escolher as seguintes opções para preservar os dados do cluster caso precise deles posteriormente:
     - Crie um snapshot final do volume do cluster. A configuração padrão é criar um snapshot final.
     - Reter backups automatizados. A configuração padrão é não reter backups automatizados.

     > **Note**
     >
     > Os backups automatizados para clusters de banco de dados do Aurora Serverless v1 não são retidos.

     O Aurora também pede para você confirmar que pretende excluir o cluster.
   - CLI e API: chame o comando `delete-db-cluster` da CLI ou a operação de API `DeleteDBCluster`. Você pode escolher as seguintes opções para preservar os dados do cluster caso precise deles posteriormente:
     - Crie um snapshot final do volume do cluster.
     - Reter backups automatizados.

     > **Note**
     >
     > Os backups automatizados para clusters de banco de dados do Aurora Serverless v1 não são retidos.

Tópicos

- Excluir um cluster do Aurora vazio
- Excluir um cluster do Aurora com uma única instância de banco de dados
- Excluir um cluster do Aurora com várias instâncias de banco de dados

Excluir um cluster do Aurora vazio

Você pode excluir um cluster de banco de dados vazio usando oConsole de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

> **Tip**
>
> É possível manter um cluster sem instâncias de banco de dados para preservar seus dados sem incorrer em cobranças de CPU pelo cluster. Comece a usar rapidamente o cluster novamente criando uma ou mais instâncias novas de banco de dados para o cluster. No entanto, você só pode adicionar novas instâncias de banco de dados usando a AWS CLI ou a API do RDS. Você não pode adicionar novas instâncias de banco de dados usando o console do Amazon RDS. Você pode executar operações administrativas específicas do Aurora no cluster enquanto ele não tiver nenhuma instância de banco de dados associada. Você só não pode acessar os dados ou executar quaisquer operações que exijam a conexão a uma instância de banco de dados.

Console

Para excluir um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados e o cluster de banco de dados que você deseja excluir.
3. Em Ações, escolha Excluir.
4. Para criar um snapshot de banco de dados final para o cluster de banco de dados, escolha Criar snapshot final?. Essa é a configuração padrão.
5. Se você optar por criar um snapshot final, insira o Final snapshot name (Nome do snapshot final).

6. Para reter backups automatizados, escolha Retain automated backups (Reter backups automatizados). Essa não é a configuração padrão.
7. Digite **`delete me`** na caixa.
8. Escolha Delete (Excluir).

CLI

Para excluir um cluster de banco de dados do Aurora vazio usando a AWS CLI, chame o comando delete-db-cluster.

Suponha que o cluster vazio `deleteme-zero-instances` foi usado apenas para desenvolvimento e teste e não contenha dados importantes. Nesse caso, você não precisa preservar um snapshot do volume instância do banco de dados primárior ao excluir o cluster. O exemplo a seguir demonstra que um cluster não contém nenhuma instância de banco de dados e, depois, exclui o cluster vazio sem criar um snapshot final nem reter backups automatizados.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-zero-instances --output
 text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-zero-instances

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-zero-instances \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-zero-instances",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

API do RDS

Para excluir um cluster de banco de dados vazio do Aurora usando a API do Amazon RDS, chame a operação de API DeleteDBCluster.

Excluir um cluster do Aurora com uma única instância de banco de dados

Você pode excluir a última instância de banco de dados mesmo que o cluster de banco de dados tenha proteção contra exclusão habilitada. Nesse caso, o cluster de banco de dados propriamente

dito ainda existirá, e seus dados serão preservados. Para acessar esses dados novamente, conecte uma nova instância de banco de dados ao cluster.

O exemplo a seguir mostra como o comando `delete-db-cluster` não funciona quando o cluster ainda tem quaisquer instâncias de banco de dados associadas. Esse cluster tem uma instância de banco de dados de gravador único. Quando examinamos as instâncias de banco de dados no cluster, verificamos o atributo `IsClusterWriter` de cada uma. O cluster pode ter zero ou uma instância de banco de dados de gravador. Um valor `true` significa uma instância de banco de dados de gravador. Um valor `false` significa uma instância de banco de dados de leitor. O cluster pode ter zero, uma ou várias instâncias de banco de dados de leitor. Nesse caso, excluímos a instância de banco de dados de gravador usando o comando `delete-db-instance`. Assim que a instância de banco de dados entrar no estado `deleting`, podemos excluir o cluster também. Para este exemplo também, suponha que o cluster não contenha dados que valham a pena preservar. Portanto, não criamos um snapshot do volume do cluster nem retemos backups automatizados.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only --skip-final-
snapshot
An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-only \
  --query '*[].[DBClusterIdentifier,Status,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]'
[
    [
        "deleteme-writer-only",
        "available",
        [
            [
                "instance-2130",
                true
            ]
        ]
    ]
]

$ aws rds delete-db-instance --db-instance-identifier instance-2130
{
  "DBInstanceIdentifier": "instance-2130",
  "DBInstanceStatus": "deleting",
```

```
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-only",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Excluir um cluster do Aurora com várias instâncias de banco de dados

Se o cluster contém várias instâncias de banco de dados, normalmente há uma única instância de gravador e uma ou mais instâncias de leitor. As instâncias de leitor ajudam com a alta disponibilidade por ficarem em espera para assumir se a instância de gravador encontrar um problema. Também é possível usar instâncias de leitor para dimensionar o cluster a fim de lidar com uma workload com uso intenso de leitura sem sobrecarregar a instância de gravador.

Para excluir um cluster com várias instâncias de banco de dados de leitor, exclua as instâncias de leitor primeiro e, em seguida, a instância de gravador. A exclusão das instâncias de gravador deixa o cluster e os respectivos dados intocados. Você exclui o cluster por meio de uma ação separada.

- Consulte para conferir o procedimento de exclusão de uma instância de banco de dados do Aurora Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.
- Consulte para ver o procedimento de exclusão da instância de banco de dados de gravador em um cluster do Aurora Excluir um cluster do Aurora com uma única instância de banco de dados.
- Consulte para ver o procedimento de exclusão de um cluster do Aurora vazio Excluir um cluster do Aurora vazio.

Este exemplo da CLI mostra como excluir um cluster que contém uma instância de banco de dados de gravador e uma única instância de banco de dados de leitor. A saída de `describe-db-clusters` mostra que `instance-7384` é a instância de gravador e `instance-1039` é a instância de leitor. No exemplo, a instância de leitor é excluída primeiro porque a exclusão da instância de gravador enquanto ainda existe uma instância de leitor causaria uma operação de failover. Não faz sentido promover a instância de leitor para uma instância de gravador se você planeja excluí-la também. Novamente, suponha que essas instâncias `db.t2.small` sejam usadas apenas para

desenvolvimento e teste. Desse modo, a operação de exclusão ignora o snapshot final e não retém backups automatizados.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader --skip-
final-snapshot

An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-and-reader --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-writer-and-reader
Instance:       instance-1039  False
Instance:       instance-7384   True

$ aws rds delete-db-instance --db-instance-identifier instance-1039
{
  "DBInstanceIdentifier": "instance-1039",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-7384
{
  "DBInstanceIdentifier": "instance-7384",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-and-reader",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

O exemplo a seguir mostra como excluir um cluster de banco de dados que contém uma instância de banco de dados e várias instâncias de banco de leitor. Ele usa uma saída concisa obtida do

comando `describe-db-clusters` para obter um relatório das instâncias de gravador e de leitor. Novamente, excluímos todas as instâncias de banco de dados de leitor antes de excluirmos a instância de banco de dados de gravador. Não importa a ordem em que excluímos as instâncias de banco de dados de leitor.

Suponha que esse cluster com várias instâncias de banco de dados contenha dados que devem ser preservados. Assim, o comando `delete-db-cluster` neste exemplo inclui os parâmetros `--no-skip-final-snapshot` e `--final-db-snapshot-identifier` para especificar os detalhes do snapshot a ser criado. Também inclui o parâmetro `--no-delete-automated-backups` para reter backups automatizados.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-multiple-readers --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-multiple-readers
Instance:       instance-1010   False
Instance:       instance-5410   False
Instance:       instance-9948   False
Instance:       instance-8451   True

$ aws rds delete-db-instance --db-instance-identifier instance-1010
{
  "DBInstanceIdentifier": "instance-1010",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-5410
{
  "DBInstanceIdentifier": "instance-5410",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-9948
{
  "DBInstanceIdentifier": "instance-9948",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}
```

```
$ aws rds delete-db-instance --db-instance-identifier instance-8451
{
  "DBInstanceIdentifier": "instance-8451",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-multiple-readers \
  --no-delete-automated-backups \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
  "DBClusterIdentifier": "deleteme-multiple-readers",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

O exemplo a seguir mostra como confirmar que o Aurora criou o snapshot solicitado. Você pode solicitar detalhes do snapshot específico definindo seu identificador `deleteme-multiple-readers-final-snapshot`. Também é possível obter um relatório de todos os snapshots do cluster que foi excluído especificando o identificador do cluster `deleteme-multiple-readers`. Os dois comandos retornam informações sobre o mesmo snapshot.

```
$ aws rds describe-db-cluster-snapshots \
  --db-cluster-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...

$ aws rds describe-db-cluster-snapshots --db-cluster-identifier deleteme-multiple-
readers
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
```

```
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...
```

Proteção contra exclusão para clusters do Aurora.

Não é possível excluir clusters com a proteção contra exclusão habilitada. É possível excluir instâncias de banco de dados dentro do cluster, mas não o cluster em si. Dessa maneira, o volume do cluster que contém todos os seus dados está protegido contra exclusão acidental. O Aurora aplica a proteção contra exclusão para um cluster de banco de dados se você tentar excluir o cluster usando o console, a AWS CLI ou a API do RDS.

Por padrão, a proteção contra exclusão permanece habilitada quando você cria um cluster de banco de dados de produção usando o Console de gerenciamento da AWS. Porém, a proteção contra exclusão será desabilitada por padrão, se você criar um cluster usando a AWS CLI ou a API. A ativação ou desativação da proteção contra exclusão não causa uma interrupção. Modifique o cluster e desabilite a proteção contra exclusão para conseguir excluí-lo. Para obter mais informações sobre como ativar e desativar a proteção contra exclusão, consulte Modificar o cluster de banco de dados usando o console, a CLI e a API.

> **Tip**
>
> Mesmo que todas as instâncias de banco de dados sejam excluídas, é possível acessar os dados criando uma nova instância de banco de dados no cluster.

Excluir um cluster parado do Aurora

Não é possível excluir um cluster se ele estiver no estado `stopped`. Nesse caso, inicie o cluster antes de excluí-lo. Para obter mais informações, consulte Iniciar um cluster de banco de dados do Aurora.

Excluir clusters do Aurora MySQL que são réplicas de leitura

No Aurora MySQL, você não pode excluir uma instância de banco de dados em um cluster do banco de dados se ambas as condições a seguir forem verdadeiras:

- O cluster de banco de dados é uma réplica de leitura de outro cluster de banco de dados do Aurora.

- A instância de banco de dados é a única instância no cluster de banco de dados.

Para excluir uma instância de banco de dados nesse caso, primeiro promova o cluster de banco de dados para que ele deixe de ser uma réplica de leitura. Quando a promoção for concluída, você poderá excluir a instância de banco de dados final do cluster de banco de dados. Para obter mais informações, consulte Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS.

O snapshot final ao excluir um cluster

Ao longo desta seção, os exemplos mostram como escolher se deseja tirar um snapshot final ao excluir um cluster do Aurora. Se você escolher fazer um snapshot final, mas o nome especificado corresponder a um snapshot existente, a operação será interrompida com erro. Nesse caso, examine os detalhes do snapshot para confirmar se ele representa o detalhe atual ou se é um snapshot mais antigo. Se o snapshot existente não tiver os dados mais recentes que você deseja preservar, renomeie o snapshot e tente novamente ou especifique um nome diferente para o parâmetro do snapshot final.

Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora

É possível excluir uma instância de banco de dados de um cluster de banco de dados do Aurora como parte do processo de exclusão de todo o cluster. Se o cluster contiver determinado número de instâncias de banco de dados, a exclusão do cluster exige a exclusão de cada uma dessas instâncias. Também é possível excluir uma ou mais instâncias de leitor em um cluster e deixá-lo em execução. Faça isso para reduzir a capacidade computacional e os encargos associados se o seu cluster não estiver ocupado.

Para excluir uma instância de banco de dados, é necessário especificar o nome da instância.

Você pode excluir uma instância de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Note**
>
> Quando a réplica do Aurora é excluída, o endpoint da instância é removido imediatamente, e a réplica do Aurora é removida do endpoint leitor. Se houver instruções em execução na réplica do Aurora que está sendo excluída, haverá um período de carência de três minutos.

As instruções existentes podem ser concluídas durante o período de carência. Quando o período de carência termina, a réplica do Aurora é fechada e excluída.

Para clusters de banco de dados do Aurora, a exclusão de uma instância de banco de dados não exclui necessariamente todo o cluster. Você pode excluir uma instância de banco de dados em um cluster do Aurora para reduzir a capacidade computacional e os encargos associados quando o cluster não estiver ocupado. Para obter informações sobre as circunstâncias especiais para clusters do Aurora que têm uma ou nenhuma instância de banco de dados, consulte Excluir um cluster do Aurora com uma única instância de banco de dados e Excluir um cluster do Aurora vazio.

Note

Não é possível excluir um cluster de banco de dados quando a proteção contra exclusão está habilitada nele. Para obter mais informações, consulte Proteção contra exclusão para clusters do Aurora..
Modifique o cluster de banco de dados para desabilitar a proteção contra exclusão. Para obter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Console

Como excluir uma instância de banco de dados em um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e a instância de banco de dados que você deseja excluir.
3. Em Ações, escolha Excluir.
4. Digite **`delete me`** na caixa.
5. Escolha Delete (Excluir).

AWS CLI

Para excluir uma instância de banco de dados usando a AWS CLI, chame o comando delete-db-instance e especifique o valor `--db-instance-identifier`.

Example

Para Linux, macOS ou Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance
```

Para Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance
```

API do RDS

Para excluir uma instância de banco de dados usando a API do Amazon RDS, chame a operação DeleteDBInstance e especifique o parâmetro `DBInstanceIdentifier`.

> **Note**
>
> Quando o status de uma instância de banco de dados for `deleting`, seu valor de certificado CA não será exibido no console do RDS nem na saída de comandos da AWS CLI ou de operações de API do RDS. Para obter mais informações sobre certificados CA, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

Marcar recursos do Amazon Aurora e do Amazon RDS

Uma tag do Amazon RDS é um par de nome-valor que você define e associa a um recurso do Amazon RDS, como uma instância ou um snapshot de banco de dados. O nome é referido como chave. Opcionalmente, é possível fornecer um valor para a chave.

Você pode usar o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS para adicionar, listar e excluir tags em recursos do Amazon RDS. Ao usar a CLI ou a API, forneça o nome do recurso da Amazon (ARN) do recurso do RDS com o qual deseja trabalhar. Para obter mais informações sobre a criação de um ARN, consulte Criar um ARN para o Amazon RDS.

É possível usar etiquetas para adicionar metadados aos recursos do Aurora e Amazon RDS. Você pode usar as tags para adicionar suas próprias anotações sobre instâncias de banco de dados, snapshots, Aurora clusters e assim por diante. Fazer isso pode ajudar você a documentar os recursos do Aurora e Amazon RDS. Você também pode usar as tags com procedimentos de manutenção automatizada.

Especificamente, você pode usar essas tags com políticas do IAM. É possível usá-las para gerenciar o acesso aos recursos do Aurora e Amazon RDS e controlar quais ações podem ser aplicadas a esses recursos. Você também pode usar essas tags para monitorar custos agrupando despesas de recursos marcados com tags semelhantes.

É possível marcar os seguintes recursos do Aurora e do Amazon RDS:

- Instâncias de banco de dados
- clusters de banco de dados
- Clusters globais do Aurora
- Endpoints do cluster de banco de dados
- Réplicas de leitura
- DB snapshots
- Snapshots de cluster de banco de dados
- Instâncias de bancos de dados reservadas
- Assinaturas de eventos
- Grupos de opções de banco de dados
- Grupos de parâmetros do banco de dados

- Grupos de parâmetros de cluster de banco de dados
- Grupos de sub-redes de banco de dados
- RDS Proxies
- Endpoint do RDS Proxy
- Implantações azuis/verdes
- Integrações ETL zero
- Backups automatizados
- Backups automatizados do cluster

> **Note**
>
> Quando você marca uma instância de banco de dados, o Aurora aplica automaticamente essas tags aos recursos associados do Insights de Performance. Atualmente, você não pode etiquetar RDS Proxies e endpoints do RDS Proxy usando o Console de gerenciamento da AWS.

Tópicos

- Por que usar tags de recurso do Amazon RDS?
- Como as tags de recurso do Amazon RDS funcionam
- Práticas recomendadas para marcação de recursos do Amazon RDS
- Copiar tags para snapshots de cluster de banco de dados
- Marcar recursos de backup automatizado
- Adicionar e excluir tags no Amazon RDS
- Tutorial: Uso de tags para especificar quais clusters de bancos de dados Aurora interromper

Por que usar tags de recurso do Amazon RDS?

É possível usar tags para fazer o seguinte:

- Categorize recursos do RDS por aplicação, projeto, departamento, ambiente e assim por diante. Por exemplo, é possível usar uma chave de tag para definir uma categoria, na qual o valor da tag é um item nessa categoria. É possível criar a tag `environment=prod`. Ou você pode definir

uma chave de tag como `project` e um valor de tag de `Salix`, o que indica que um recurso do Amazon RDS está atribuído ao projeto Salix.

- Automatize tarefas de gerenciamento de recursos. Por exemplo, é possível criar uma janela de manutenção para instâncias marcadas com `environment=prod` que seja diferente da janela para instâncias marcadas com `environment=test`. Também é possível configurar snapshots de banco de dados automáticos para instâncias marcadas com `environment=prod`.
- Controle o acesso aos recursos do RDS em uma política do IAM. Faça isso usando a chave de condição global `aws:ResourceTag/tag-key`. Por exemplo, uma política pode permitir que somente usuários do grupo `DBAdmin` modifiquem instâncias de banco de dados marcadas com `environment=prod`. Consulte informações sobre como gerenciar o acesso a recursos marcados com políticas do IAM em Gerenciamento de identidade e acesso no Amazon Aurora e em Controlar o acesso aos recursos da AWS no Guia do usuário do AWS Identity and Access Management.
- Monitore recursos com base em uma tag. Por exemplo, é possível criar um painel do Amazon CloudWatch para instâncias de banco de dados marcadas com `environment=prod`.
- Monitore custos agrupando despesas de recursos marcados com tags semelhantes. Por exemplo, se marcar os recursos do RDS associados ao projeto Salix com `project=Salix`, você poderá gerar relatórios de custos e alocar despesas para esse projeto. Para obter mais informações, consulte Como o faturamento da AWS funciona com tags no Amazon RDS.

Como as tags de recurso do Amazon RDS funcionam

AWSA não aplica nenhum significado semântico às suas tags. Tags são interpretadas estritamente como sequências de caracteres.

Tópicos

- Conjuntos de tags no Amazon RDS
- Estrutura de tags no Amazon RDS
- Recursos do Amazon RDS elegíveis para marcação
- Como o faturamento da AWS funciona com tags no Amazon RDS

Conjuntos de tags no Amazon RDS

Cada recurso do Amazon RDS tem um contêiner chamado conjunto de tags. O contêiner inclui todas as tags atribuídas ao recurso. Um recurso tem exatamente um conjunto de tags.

Um conjunto de tags contém de 0 a 50 tags. Se você adicionar uma tag a um recurso do RDS que tenha a mesma chave que uma tag existente no recurso, o novo valor substituirá o antigo.

Estrutura de tags no Amazon RDS

A estrutura de uma tag do RDS é a seguinte:

Chave de tag

A chave é o nome obrigatório da tag. O valor da string deve ter de 1 a 128 caracteres Unicode e não pode ter os prefixos `aws:` ou `rds:`. A string pode conter apenas o conjunto de letras, números e espaços em branco Unicode, _, ., :, /, =, +, - e @. O regex Java é `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. As chaves de tag fazem distinção entre maiúsculas e minúsculas. Assim, as chaves `project` e `Project` são distintas.

Uma chave é exclusiva a um conjunto de tags. Por exemplo, não pode haver um par de chaves em um conjunto de tags com a mesma chave mas com valores diferentes, como `project=Trinity` e `project=Xanadu`.

Valor da tag

O valor é um valor de string opcional da tag. O valor da string deve ter de 1 a 256 caracteres Unicode. A string pode conter apenas o conjunto de letras, números e espaços em branco Unicode, _, ., :, /, =, +, - e @. O regex Java é `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Os valores de tags diferenciam maiúsculas de minúsculas. Assim, os valores `prod` e `Prod` são distintos.

Os valores não precisam ser exclusivos em um conjunto de tags e podem ser nulos. Por exemplo, você pode ter um par de valor-chave em um conjunto de tag de `project=Trinity` e `cost-center=Trinity`.

Recursos do Amazon RDS elegíveis para marcação

Você pode marcar os seguintes recursos do Amazon RDS:

- Instâncias de banco de dados
- clusters de banco de dados
- Endpoints do cluster de banco de dados
- Réplicas de leitura
- DB snapshots

- Snapshots de cluster de banco de dados
- Instâncias de bancos de dados reservadas
- Assinaturas de eventos
- Grupos de opções de banco de dados
- Grupos de parâmetros do banco de dados
- Grupos de parâmetros de cluster de banco de dados
- Grupos de sub-redes de banco de dados
- RDS Proxies
- Endpoint do RDS Proxy

> **Note**
>
> Atualmente, você não pode etiquetar RDS Proxies e endpoints do RDS Proxy usando o Console de gerenciamento da AWS.

- Implantações azuis/verdes
- Integrações ETL zero (pré-visualização)
- Backups automatizados
- Backups automatizados do cluster

Como o faturamento da AWS funciona com tags no Amazon RDS

Também é possível utilizar tags para organizar sua fatura da AWS para refletir sua própria estrutura de custo. Para fazer isso, inscreva-se para obter a fatura da sua Conta da AWS com os valores de chave de tag incluídos. Então, para ver o custo de recursos combinados, organize suas informações de faturamento de acordo com recursos com os mesmos valores de chave de tags. Por exemplo, é possível marcar vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento para ver o custo total daquela aplicação em vários serviços. Para obter mais informações, consulte Usar tags de alocação de custos no Guia do usuário do AWS Billing.

Como as tags de alocação de custos funcionam com snapshots de cluster de banco de dados

É possível adicionar uma tag a um snapshot decluster de banco de dados. No entanto, sua conta não refletirá esse agrupamento. Para que tags de alocação de custo sejam aplicadas a snapshots de cluster de banco de dados, as seguintes condições devem ser atendidas:

- As tags devem ser anexadas à instância de banco de dados principal.
- A instância de banco de dados principal deve estar na mesma Conta da AWS que o snapshot do cluster de banco de dados.
- A instância de banco de dados principal deve estar na mesma Região da AWS que o snapshot do cluster de banco de dados.

Os snapshots do cluster de banco de dados serão considerados órfãos se não estiverem na mesma região da instância de banco de dados principal ou se esta for excluída. Os snapshots de banco de dados órfãos não comportam tags de alocação de custos. Os custos de snapshots órfãos são agregados em um único item de linha não marcado. Os snapshots do cluster de banco de dados entre contas não são considerados órfãos quando as seguintes condições são atendidas:

- Eles estão na mesma região da instância de banco de dados principal.
- A instância de banco de dados principal pertence à conta de origem.

> **Note**
>
> Se a instância de banco de dados principal pertencer a uma conta diferente, as tags de alocação de custos não se aplicarão aos snapshots de várias contas na conta de destino.

Práticas recomendadas para marcação de recursos do Amazon RDS

Ao usar tags, sugerimos seguir estas práticas recomendadas:

- Convenções de documentos para uso de tags que são seguidas por todas as equipes da organização. Acima de tudo, os nomes devem ser descritivos e consistentes. Por exemplo, padronize o formato `environment:prod` em vez de marcar alguns recursos com `env:production`.

> **Important**
>
> Não armazene informações de identificação pessoal (PII) nem outras informações confidenciais ou sensíveis em tags.

- Automatize a marcação para garantir a consistência. Por exemplo, é possível usar as seguintes técnicas:

  - Inclua tags em um modelo do CloudFormation. Ao criar recursos com o modelo, eles são marcados automaticamente.
  - Defina e aplique tags usando funções do AWS Lambda.
  - Crie um documento do SSM que inclua etapas para adicionar tags aos recursos do RDS.
- Use tags somente quando necessário. É possível adicionar até 50 tags a um único recurso do RDS, mas a prática recomendada é evitar a proliferação e a complexidade desnecessárias de tags.
- Revise as tags periodicamente para verificar a relevância e a precisão. Remova ou modifique as tags desatualizadas conforme necessário.
- Considere criar tags com o Editor de Tags da AWS no Console de gerenciamento da AWS. É possível usar o Editor de Tags para adicionar tags a vários recursos compatíveis da AWS ao mesmo tempo, incluindo os recursos do RDS. Para obter mais informações, consulte o Tag Editor no Guia do usuário dos Grupos de recursos da AWS.

Copiar tags para snapshots de cluster de banco de dados

Ao criar ou restaurar um cluster de banco de dados, você pode especificar que as tags do cluster sejam copiadas para snapshots do cluster de banco de dados. A cópia de tags garante que os metadados dos snapshots de banco de dados correspondam aos do cluster de banco de dados de origem. Também garante que quaisquer políticas de acesso do snapshot de banco de dados também correspondam às do cluster de banco de dados de origem. Tags não são copiadas por padrão.

Você pode especificar que as tags sejam copiados para snapshots de banco de dados para as seguintes ações:

- Criar um cluster de banco de dados
- Restaurar um cluster de banco de dados
- Como criar uma réplica de leitura
- Copiar um snapshot de cluster de banco de dados

> **Note**
>
> Em alguns casos, você pode incluir um valor para o parâmetro `--tags` do comando create-db-snapshot da AWS CLI. Ou pode fornecer pelo menos uma tag à operação da API

CreateDBSnapshot. Nesses casos, o RDS não copia tags da instância de banco de dados de origem para o novo snapshot de banco de dados. Essa funcionalidade é aplicável mesmo que a instância de banco de dados de origem tenha a opção `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) ativada.
Se você seguir essa abordagem, poderá criar uma cópia de uma instância de banco de dados de um snapshot de banco de dados. Essa abordagem evita adicionar tags que não se aplicam à nova instância de banco de dados. Você cria o snapshot de banco de dados com o comando `create-db-snapshot` da AWS CLI (ou a operação `CreateDBSnapshot` da API do RDS). Depois de criar o snapshot de banco de dados, é possível adicionar etiquetas conforme descrito posteriormente neste tópico.

Marcar recursos de backup automatizado

Os recursos de backup automatizado são criados quando você define o valor do período de retenção de backup de 0 a maior que 0. É possível marcar recursos de backup automatizado da instância ou do cluster durante a criação usando o parâmetro `--tag-specifications`.

Parâmetro de especificações de tag

As APIs que aceitam o parâmetro de solicitação `--tag-specifications` (como create-db-instance, restore-db-instance-from-db-snapshot, create-db-cluster etc.) podem marcar backups automatizados (Tipo de recurso: `auto-backup` ou `cluster-auto-backup`) durante a criação.

Marcar backups automatizados do cluster

Use `--tag-specifications` com `ResourceType=cluster-auto-backup` ao criar clusters de banco de dados com backups automatizados habilitados.

> **Note**
>
> - As tags de backup automatizado são independentes das tags de instância de banco de dados de origem, tags de cluster de banco de dados ou tags de snapshot de banco de dados.

Adicionar e excluir tags no Amazon RDS

Você pode fazer o seguinte:

- Crie tags ao criar um recurso, por exemplo, ao executar o comando `create-db-instance` da AWS CLI.
- Adicione tags a um recurso existente usando o comando `add-tags-to-resource`.
- Liste as tags associadas a um recurso específico usando o comando `list-tags-for-resource`.
- Atualize as tags usando o comando `add-tags-to-resource`.
- Remova as tags de um recurso usando o comando `remove-tags-from-resource`.

Os procedimentos a seguir mostram como executar operações típicas de marcação em recursos relacionados a instâncias de banco de dados e clusters de bancos de dados do Aurora. Observe que as tags ficam armazenadas em cache para fins de autorização. Por esse motivo, ao adicionar ou atualizar tags nos recursos do Amazon RDS, pode demorar vários minutos para que as modificações sejam disponibilizadas.

Console

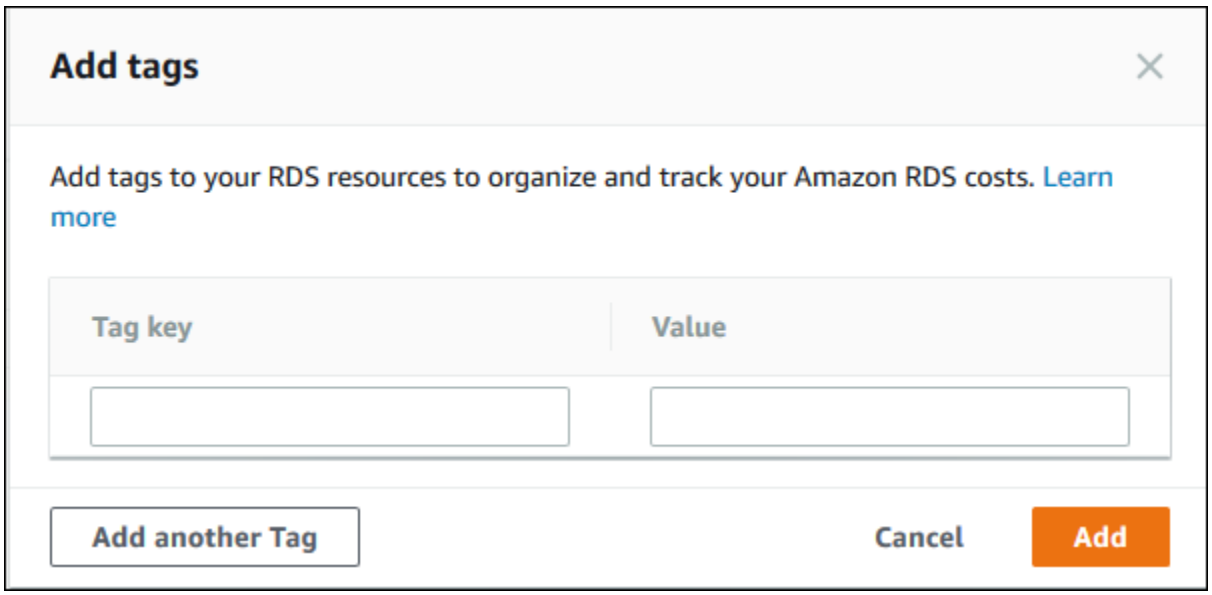
O processo para marcar um recurso do Amazon RDS é semelhante para todos os recursos. O procedimento a seguir mostra como marcar uma instância de banco de dados do Amazon RDS.

Para adicionar uma tag a uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).

   > **Note**
   >
   > Para filtrar a lista de instâncias de bancos de dados no painel Databases (Bancos de dados), digite uma string de texto de Filter databases (Filtrar bancos de dados). Somente instâncias de banco de dados que contiverem a string aparecerão.

3. Escolha no nome da instância de banco de dados que você deseja marcar para mostrar os detalhes.
4. Na seção de detalhes, role para baixo até a seção Tags.
5. Escolha Adicionar. A janela Add tags (Adicionar tags) é exibida.



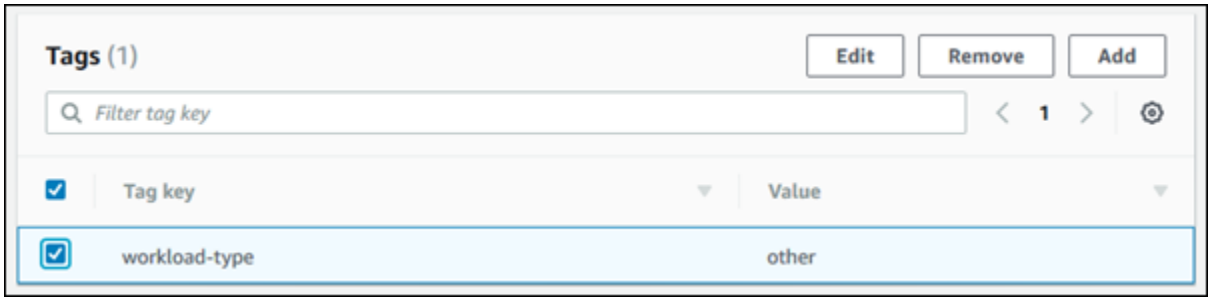
6. Digite um valor para Tag key (Chave de tag) e Value (Valor).
7. Para adicionar outra tag, escolha Add another Tag (Adicionar outra tag) e digite um valor para Tag key (Chave de tag) e Value (Valor).

   Repita esta etapa quantas vezes for necessário.
8. Escolha Adicionar.

Para excluir uma tag de uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).

   > **Note**
   >
   > Para filtrar a lista de instâncias de bancos de dados no painel Databases (Bancos de dados), digite uma string de texto na caixa Filter databases (Filtrar bancos de dados). Somente instâncias de banco de dados que contiverem a string aparecerão.
3. Escolha o nome da instância de banco de dados para mostrar os detalhes.
4. Na seção de detalhes, role para baixo até a seção Tags.
5. Escolha a tag que você deseja excluir.



6. Selecione Delete (Excluir) e escolha Delete (Excluir) na janela Delete tags (Excluir tags).

AWS CLI

É possível adicionar, listar ou remover tags de uma instância de banco de dados usando a AWS CLI.

- Para adicionar uma ou mais etiquetas a um recurso do Amazon RDS, use o comando da AWS CLI `add-tags-to-resource`.
- Para listar as etiquetas em um recurso do Amazon RDS, use o comando da AWS CLI `list-tags-for-resource`.
- Para remover uma ou mais etiquetas de um recurso do Amazon RDS, use o comando da AWS CLI `remove-tags-from-resource`.

Para saber mais sobre como criar o ARN necessário, consulte Criar um ARN para o Amazon RDS.

API do RDS

É possível adicionar, listar ou remover tags de uma instância de banco de dados usando a API do Amazon RDS.

- Para adicionar uma tag a um recurso do Amazon RDS, use a operação `AddTagsToResource`.
- Para listar tags atribuídas a um recurso do Amazon RDS, use `ListTagsForResource`.
- Para remover tags de um recurso do Amazon RDS, use a operação `RemoveTagsFromResource`.

Para saber mais sobre como criar o ARN necessário, consulte Criar um ARN para o Amazon RDS.

As tags usam o seguinte esquema ao trabalhar com o XML usando a API do Amazon RDS:

```
<Tagging>
      <TagSet>
          <Tag>
```

```
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

A tabela a seguir fornece uma lista das tags XML permitidas e suas características. Os valores de `Key` e `Value` diferenciam letras maiúsculas de minúsculas. Por exemplo, `project=Trinity` e `PROJECT=Trinity` são tags distintas.

Elemento de marcação por tag	Descrição
TagSet	Um conjunto de tags é um contêiner de todas as tags atribuídas a um recurso do Amazon RDS. Só pode haver um conjunto de tags por recurso. Você trabalha com um TagSet somente por meio da API do Amazon RDS.
Tag	Uma tag é um par de chave-valor definido pelo usuário. Pode haver de 1 a 50 tags em um conjunto de tags.
Chave	Uma chave é o nome obrigatório da tag. Consulte as restrições em Estrutura de tags no Amazon RDS. O valor da string pode ter de 1 a 128 caracteres Unicode e não pode ter os prefixos `aws:` ou `rds:`. A string pode conter apenas o conjunto de letras em Unicode, dígitos, espaços em branco, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\ \\p{Z}\\p{N}_.:/=+\\-]*)$"). As chaves devem ser exclusivas a um conjunto de tags. Por exemplo, não pode haver um par de chaves em um conjunto de tags com a mesma chave mas com valores diferentes, como projeto/Trinity e projeto/Xanadu.
Valor	Um valor é o valor opcional da tag. Consulte as restrições em Estrutura de tags no Amazon RDS.

Elemento de marcação por tag	Descrição
	O valor da string pode ter de 1 a 256 caracteres Unicode e não pode ter os prefixos `aws:` ou `rds:`. A string pode conter apenas o conjunto de letras em Unicode, dígitos, espaços em branco, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\ \p{Z}\\p{N}_.:/=+\\-]*)$"). Os valores não têm que ser exclusivos em um conjunto de tags e podem ser nulos. Por exemplo, você pode ter um par de chave-valor em um conjunto de tags definido como projeto/Trinity e centro-custos/Trinity.

Tutorial: Uso de tags para especificar quais clusters de bancos de dados Aurora interromper

Suponha que você esteja criando vários clusters de banco de dados Aurora em um ambiente de desenvolvimento ou teste. Você precisa manter todos esses clusters por vários dias. Alguns dos clusters realizam testes durante a noite. Outros clusters podem ser interrompidos durante a noite e recomeçar no dia seguinte. O exemplo a seguir mostra como atribuir uma tag aos clusters que são adequados para parar durante a noite. Em seguida, o exemplo mostra como um script pode detectar quais clusters têm essa tag e, em seguida, parar esses clusters. Neste exemplo, a parte de valor do par chave-valor não importa. A presença da tag `stoppable` significa que o cluster tem essa propriedade definida pelo usuário.

Para especificar quais clusters de bancos de dados Aurora devem ser interrompidas

1. Primeiro, determine o ARN de um cluster que você queira designar como passível de ser interrompida.

   Os comandos e as APIs para marcação funcionam com ARNs. Dessa forma, eles podem funcionar perfeitamente em regiões da AWS, contas da AWS e diferentes tipos de recursos que podem ter nomes curtos idênticos. Você pode especificar o ARN em vez do ID do cluster nos comandos CLI que se operam em clusters. Substitua o nome do seu próprio cluster por *`dev-test-cluster`*. Nos comandos subsequentes que usam parâmetros ARN, substitua o ARN do

seu próprio cluster. O ARN inclui seu próprio ID de conta da AWS e o nome da região da AWS onde seu cluster está localizado.

```
$ aws rds describe-db-clusters --db-cluster-identifier dev-test-cluster \
  --query "*[].{DBClusterArn:DBClusterArn}" --output text
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
```

2. Adicione a tag `stoppable` a esse cluster.

   Selecione o nome dessa tag. Essa abordagem significa que você pode evitar a criação de uma convenção de nomenclatura que codifique todas as informações relevantes nos nomes. Nessa convenção, você pode codificar informações no nome da instância de banco de dados ou nos nomes de outros recursos. Como esse exemplo trata a tag como um atributo presente ou ausente, ele omite a parte `Value=` do parâmetro `--tags`.

   ```
   $ aws rds add-tags-to-resource \
     --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster \
     --tags Key=stoppable
   ```

3. Confirme se a etiqueta está presente no cluster.

   Esses comandos recuperam as informações de tag para o cluster no formato JSON e em texto separado por tabulação simples.

   ```
   $ aws rds list-tags-for-resource \
     --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
   {
       "TagList": [
           {
               "Key": "stoppable",
               "Value": ""
   
           }
       ]
   }
   $ aws rds list-tags-for-resource \
     --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster --output
    text
   TAGLIST stoppable
   ```

4. Para interromper todos os clusters designados como `stoppable`, prepare uma lista de todos os clusters. Percorra a lista e verifique se cada cluster está marcado com o atributo relevante.

Este exemplo do Linux usa scripts de shell para salvar a lista de ARNs de cluster em um arquivo temporário e, em seguida, executar comandos CLI para cada cluster.

```
$ aws rds describe-db-clusters --query "*[].[DBClusterArn]" --output text >/tmp/
cluster_arns.lst
$ for arn in $(cat /tmp/cluster_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 'TAGLIST\tstoppable')"
  if [[ ! -z "$match" ]]
  then
      echo "Cluster $arn is tagged as stoppable. Stopping it now."
# Note that you can specify the full ARN value as the parameter instead of the
 short ID 'dev-test-cluster'.
      aws rds stop-db-cluster --db-cluster-identifier $arn
  fi
done

Cluster arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster is tagged as
 stoppable. Stopping it now.
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-1e",
            "us-east-1c",
            "us-east-1d"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "dev-test-cluster",
        ...
```

Você pode executar um script como este no final de cada dia para garantir que os clusters não essenciais sejam interrompidas. Você também pode agendar um trabalho com um utilitário, como `cron`, para realizar essa verificação todas as noites. Por exemplo, você pode fazer isso caso alguns clusters sejam deixados em execução por engano. Nesse caso, você pode ajustar o comando que prepara a lista de clusters para conferir.

O comando a seguir produz uma lista de seus clusters, mas apenas os que estão no estado `available`. O script pode ignorar clusters que já estão parados, porque eles terão valores de status diferentes, como `stopped` ou `stopping`.

```
$ aws rds describe-db-clusters \
  --query '*[].{DBClusterArn:DBClusterArn,Status:Status}|[?Status == `available`]|[].
{DBClusterArn:DBClusterArn}' \
  --output text
arn:aws:rds:us-east-1:123456789:cluster:cluster-2447
arn:aws:rds:us-east-1:123456789:cluster:cluster-3395
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
arn:aws:rds:us-east-1:123456789:cluster:pg2-cluster
```

Tip

Você pode usar a atribuição de tags e encontrar clusters com elas a fim de reduzir custos de outras maneiras. Por exemplo, considere o cenário com clusters de banco de dados do Aurora utilizados para desenvolvimento e testes. Aqui, você pode designar que alguns clusters sejam excluídos no final de cada dia ou que apenas as instâncias de banco de dados do leitor sejam excluídas. Ou você pode designar que alguns tenham as instâncias de banco de dados alteradas para classes de instância de banco de dados pequenas durante os períodos de baixa utilização esperada.

Nomes de recurso da Amazon (ARNs) no Amazon RDS

Os recursos criados na Amazon Web Services são identificados de forma exclusiva com um nome de recurso da Amazon (ARN). Para determinadas operações do Amazon RDS, você precisará identificar exclusivamente um recurso do Amazon RDS especificando seu ARN. Por exemplo, quando você cria uma réplica de leitura da instância de banco de dados do RDS, é necessário fornecer o ARN para a instância de banco de dados de origem.

Para ter informações sobre como criar um ARN e obter um ARN existente, consulte os tópicos a seguir.

Tópicos

- Criar um ARN para o Amazon RDS
- Obter um ARN existente para o Amazon RDS

Criar um ARN para o Amazon RDS

Os recursos criados na Amazon Web Services são identificados de forma exclusiva com um nome de recurso da Amazon (ARN). Você pode criar um ARN para um recurso do Amazon RDS usando a seguinte sintaxe.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Em relação a recursos globais de cluster, o ARN não inclui uma Região da AWS: `arn:aws:rds::<account number>:global-cluster:<name>`. Os ARNs para clusters globais não aparecem no Console de gerenciamento da AWS.

Nome da Região	Região	Endpoint	Protocolo	
Leste dos EUA (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	

Nome da Região	Região	Endpoint	Protocolo
Leste dos EUA (Norte da Virgínia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
		rds-fips.us-east-1.api.aws	HTTPS
		rds-fips.us-east-1.amazonaws.com	HTTPS
		rds.us-east-1.api.aws	HTTPS
Oeste dos EUA (N. da Califórnia)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
Oeste dos EUA (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
África (Cidade do Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
Ásia-Pacífico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
Ásia-Pacífico (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS

Nome da Região	Região	Endpoint	Protocolo
Ásia-Pacífico (Jacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com rds.ap-southeast-3.api.aws	HTTPS HTTPS
Ásia-Pacífico (Malásia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Ásia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
Ásia-Pacífico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Ásia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS
Ásia-Pacífico (Seul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS
Ásia-Pacífico (Singapura)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS

Nome da Região	Região	Endpoint	Protocolo
Ásia-Pacífico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
Ásia-Pacífico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
Ásia-Pacífico (Tailândia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
Ásia-Pacífico (Tóquio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
Canadá (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
Oeste do Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS
Europa (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
		rds.eu-west-1.api.aws	HTTPS

Nome da Região	Região	Endpoint	Protocolo
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS
Europa (Milão)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS
Europa (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS
Europa (Espanha)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS
Europa (Zurique)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS
México (Central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Oriente Médio (Barém)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS

Nome da Região	Região	Endpoint	Protocolo
Oriente Médio (Emirados Árabes Unidos)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
América do Sul (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Leste dos EUA)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

A tabela a seguir mostra o formato que deve ser usado para criar um ARN para um tipo de recurso específico do Amazon RDS.

Tipo de recurso	Formato ARN
Instância de banco de dados	arn:aws:rds:*<region>*:*<account>* :db:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
Cluster de banco de dados	arn:aws:rds:*<region>*:*<account>* :cluster:*<name>*

Tipo de recurso	Formato ARN
	Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
Assinatura de eventos	arn:aws:rds:*<region>*:*<account>* :es:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (Grupo de parâmetros de banco de dados)	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Parameter group do cluster de banco de dados	arn:aws:rds:*<region>*:*<account>* :cluster-pg:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Instância de banco de dados reservada	arn:aws:rds:*<region>*:*<account>* :ri:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :ri:my-reserved-postgresql`

Tipo de recurso	Formato ARN
Grupo de segurança de banco de dados	arn:aws:rds:*<region>*:*<account>* :secgrp:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :secgrp:my-public`
Snapshot de banco de dados automatizado	arn:aws:rds:*<region>*:*<account>* :snapshot:rds:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :snapshot:rds: my-mysql-db-2019-07-22-07-23`
Snapshot de cluster de banco de dados automatizado	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:rds:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds: my-aurora-cluster-2019-07-22-16-16`
Snapshot de banco de dados manual	arn:aws:rds:*<region>*:*<account>* :snapshot:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`
Snapshot de cluster de banco de dados manual	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`

Tipo de recurso	Formato ARN
Grupo de sub-rede de banco de dados	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* Por exemplo: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet -10`
Cluster global	arn:aws:rds::*<account>* :global-cluster:*<name>* Por exemplo: `arn:aws:rds:: 123456789012 :global-cluster: my-aurora-global-cluster-1`

Obter um ARN existente para o Amazon RDS

Você pode obter o ARN de um recurso do RDS usando o Console de gerenciamento da AWS, a AWS Command Line Interface (AWS CLI) ou a API do RDS.

Console

Para obter um ARN do Console de gerenciamento da AWS, navegue até o recurso para o qual deseja um ARN e veja os detalhes desse recurso.

Por exemplo, você pode obter o ARN de um cluster de banco de dados da guia Configuração dos detalhes do cluster de banco de dados.
ARN do cluster de banco de dados.

AWS CLI

Para obter um ARN a partir da AWS CLI para um recurso do RDS específico, use o comando `describe` para esse recurso. A tabela a seguir mostra cada comando da AWS CLI e a propriedade do ARN usada com o comando para obter um ARN.

AWS CLI command	Propriedade do ARN
describe-event-subscriptions	EventSubscriptionArn

AWS CLI command	Propriedade do ARN
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	ReservedDBInstanceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Por exemplo, o seguinte comando da AWS CLI obtém o ARN para uma instância de banco de dados.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Para Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
```

```
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

A saída desse comando é semelhante à seguinte:

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

API do RDS

Para obter um ARN para um recurso do RDS específico, é possível chamar as seguintes operações da API do RDS e usar as propriedades do ARN mostradas a seguir.

Operação da API do RDS	Propriedade do ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeDBClusters	DBClusterArn

Operação da API do RDS	Propriedade do ARN
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Atualizações do Amazon Aurora

O Amazon Aurora lança atualizações regularmente. Essas atualizações são aplicadas a clusters de banco de dados do Amazon Aurora durante janelas de manutenção do sistema. O cronograma de aplicação das atualizações depende da configuração da região e da janela de manutenção para o cluster de banco de dados, além do tipo de atualização. Como as atualizações exigem uma reinicialização de banco de dados, você normalmente enfrenta um tempo de inatividade de 20 a 30 segundos. Depois desse período de inatividade, você poderá retomar um ou mais clusters de banco de dados. É possível exibir ou alterar as configurações da janela de manutenção no Console de gerenciamento da AWS.

> **Note**
>
> O tempo necessário para reinicializar a instância de banco de dados depende do processo de recuperação de falhas, da atividade do banco de dados no momento da reinicialização e do comportamento do mecanismo de banco de dados específico. Para melhorar o tempo de reinicialização, recomendamos reduzir as atividades do banco de dados o máximo possível durante o processo de reinicialização. A redução das atividades do banco de dados reduz as atividades de reversão para transações em trânsito.

Para obter informações sobre atualizações do sistema operacional para o Amazon Aurora, consulte Atualizações do sistema operacional para clusters de banco de dados do Aurora.

Algumas atualizações são específicas de um mecanismo de banco de dados compatível com o Aurora. Para obter mais informações sobre atualizações do mecanismo de banco de dados, consulte a tabela a seguir.

Mecanismo do banco de dados	Atualizações
Amazon Aurora MySQL	Consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL

Identificar a versão do Amazon Aurora

O Amazon Aurora contém determinados recursos que são gerais para o Aurora e estão disponíveis para todos os clusters de banco de dados do Aurora. O Aurora contém outros recursos que são específicos de um mecanismo de banco de dados compatível com o Aurora. Esses recursos estão disponíveis somente para esses clusters de banco de dados do Aurora que usam o mecanismo de banco de dados, como o Aurora PostgreSQL.

Uma instância de banco de dados do Aurora fornece dois números de versão, o número de versão do Aurora e o número de versão do mecanismo de banco de dados do Aurora. Os números de versão do Aurora usam o formato a seguir.

```
<major version>.<minor version>.<patch version>
```

Para obter o número de versão do Aurora de uma instância de banco de dados do Aurora usando um mecanismo de banco de dados específico, use uma das consultas a seguir.

Mecanismo do banco de dados	Consultas
Amazon Aurora MySQL	`SELECT AURORA_VERSION();` `SHOW @@aurora_version;`
Amazon Aurora PostgreSQL	`SELECT AURORA_VERSION();`

Suporte estendido do Amazon RDS com Amazon Aurora

Após a data de término do suporte padrão do Aurora, o Suporte estendido do RDS permite que você continue executando um banco de dados em uma versão principal do mecanismo por um custo adicional.

Você só pode inscrever um banco de dados no Suporte estendido do RDS habilitando o Suporte estendido do RDS ao criar ou restaurar uma instância de banco de dados pela primeira vez. Não é possível atualizar o status de sua inscrição no Suporte estendido do RDS em instâncias de banco de dados existentes, a menos que as esteja restaurando.

Se você tiver habilitado o Suporte estendido do RDS durante a criação de uma instância de banco de dados, após a data de fim do suporte padrão do Aurora, o Amazon Aurora inscreverá automaticamente a instância de banco de dados no Suporte estendido do RDS. A inscrição automática no suporte estendido do RDS não altera o mecanismo do banco de dados e não afeta o tempo de atividade nem a performance da instância de banco de dados.

O Suporte estendido do RDS fornece as seguintes atualizações e suporte técnico:

- Atualizações de segurança para CVEs críticas e altas para a instância de banco de dados ou cluster de banco de dados, incluindo o mecanismo de banco de dados
- Correções de erros e patches para problemas críticos
- A capacidade de abrir casos de suporte e receber ajuda para solução de problemas dentro do acordo de serviço padrão do Amazon RDS

Essa oferta paga oferece a você mais tempo para atualizar para uma versão principal compatível do mecanismo. Por exemplo, a data de término do suporte padrão do Aurora, para o Aurora MySQL versão 2, é 31 de outubro de 2024. No entanto, não é possível atualizar manualmente para o Aurora MySQL versão 3 antes dessa data. Nesse caso, o Amazon Aurora inscreverá automaticamente o cluster no Suporte estendido do RDS em 31 de outubro de 2024. É possível continuar executando o Aurora MySQL versão 2. A partir de 1.º de dezembro de 2024, o Amazon Aurora cobrará você automaticamente pelo Suporte estendido do RDS.

O Suporte estendido do RDS ficará disponível por até três anos após a data de fim de vida útil da comunidade para uma versão principal (três anos e quatro meses para o Aurora MySQL versão 2). Após esse período, se você não tiver feito upgrade para a versão principal do mecanismo para uma versão compatível, o Amazon Aurora fará upgrade automático da versão do mecanismo principal.

Recomendamos que você atualize para uma versão principal compatível do mecanismo o mais rápido possível.

Para ter mais informações sobre as datas de fim do suporte padrão do Aurora e as datas de fim do suporte estendido do RDS, consulte Release calendar for Aurora MySQL major versions e Release calendar for Aurora PostgreSQL major versions.

Tópicos

- Visão geral do Suporte estendido do Amazon RDS
- Cobranças do Suporte estendido do Amazon RDS
- Versões com Suporte estendido do Amazon RDS
- Responsabilidades do Amazon Aurora e do cliente com o Suporte estendido do Amazon RDS
- Criação de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS
- Visualizar a inscrição de clusters de banco de dados do Aurora ou clusters globais no Suporte estendido do Amazon RDS
- Visualizar datas de suporte de versões de mecanismo no Suporte estendido do Amazon RDS
- Restauração de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS

Visão geral do Suporte estendido do Amazon RDS

Após a data de fim do suporte padrão do Aurora, se você não tiver desabilitado o suporte estendido do RDS durante a criação ou a restauração das instâncias de banco de dados, o Amazon Aurora as inscreverá automaticamente no suporte estendido do RDS. O Aurora atualizará a instância de banco de dados para a versão secundária mais recente lançada antes da data de término do suporte padrão do Aurora, caso você ainda não esteja executando essa versão. O Amazon Aurora só atualizará a versão secundária depois da data de término do suporte padrão do Aurora.

É possível criar bancos de dados com as principais versões do mecanismo que atingiram a data de término do suporte padrão do Aurora. O Aurora inscreve automaticamente esses novos bancos de dados no Suporte estendido do RDS e cobra por essa oferta.

Se você realizar a atualização para um mecanismo que ainda esteja sob o suporte padrão do Aurora antes da data de término do suporte padrão do Aurora, o Amazon Aurora não inscreverá o mecanismo no Suporte estendido do RDS.

Se você tentar restaurar um snapshot de um banco de dados compatível com um mecanismo que já passou da data de término do suporte padrão do Aurora, mas não está inscrito no Suporte estendido do RDS, o Amazon Aurora tentará atualizar o snapshot para ser compatível com a versão mais recente do mecanismo que ainda está sob o suporte padrão do Aurora. Se a restauração falhar, o Amazon Aurora inscreverá automaticamente o mecanismo no Suporte estendido do RDS com uma versão compatível com o snapshot.

É possível encerrar a inscrição no Suporte estendido do RDS a qualquer momento. Para encerrar a inscrição, atualize cada mecanismo inscrito para uma versão mais nova do mecanismo que ainda esteja sob o suporte padrão do Aurora. O término da inscrição no Suporte estendido do RDS entrará em vigor no dia em que você concluir a atualização para uma versão mais recente do mecanismo que ainda esteja sob o suporte padrão do Aurora.

Para ter mais informações sobre as datas de fim do suporte padrão do Aurora e as datas de fim do suporte estendido do RDS, consulte Release calendar for Aurora MySQL major versions e Release calendar for Aurora PostgreSQL major versions.

Cobranças do Suporte estendido do Amazon RDS

Você pagará por todos os mecanismos inscritos no Suporte estendido do RDS desde o dia seguinte à data de término do suporte padrão do Aurora. Para saber a data de término do suporte padrão do Aurora, consulte Versões principais do Amazon Aurora.

A cobrança adicional pelo Suporte estendido do RDS é interrompida automaticamente ao realizar uma das seguintes ações:

- Atualizar para uma versão do mecanismo coberta pelo suporte padrão.
- Excluir o banco de dados que está executando uma versão principal após a data de fim do suporte padrão do Aurora.

As cobranças serão reiniciadas se a versão do mecanismo de destino entrar no Suporte estendido do RDS no futuro.

Por exemplo, o Aurora PostgreSQL 11 entra no Suporte estendido em 1.º de março de 2024, mas as cobranças só começam a partir de 1.º de abril de 2024. Você atualiza o banco de dados do Aurora PostgreSQL 11 para o Aurora PostgreSQL 12 em 30 de abril de 2024. Você só receberá cobrança por trinta dias de Suporte estendido no Aurora PostgreSQL 11. Você continua executando

o Aurora PostgreSQL 12 nessa instância de banco de dados após a data de fim do suporte padrão do RDS, em 28 de fevereiro de 2025. O banco de dados vai gerar novamente cobranças do Suporte estendido do RDS desde 1.º de março de 2025.

Para obter mais informações, consulte Definição de preço do Amazon Aurora.

Evitar cobranças do Suporte estendido do Amazon RDS

É possível evitar a cobrança do Suporte estendido do RDS impedindo o Aurora de criar ou restaurar um cluster de banco de dados do Aurora ou um cluster global depois da data de término do suporte padrão do Aurora. Para fazer isso, use a AWS CLI ou a API do RDS.

Na AWS CLI, especifique `open-source-rds-extended-support-disabled` para a opção `--engine-lifecycle-support`. Na API do RDS, especifique `open-source-rds-extended-support-disabled` para o parâmetro `LifeCycleSupport`. Para ter mais informações, consulte Criação de um cluster de banco de dados do Aurora ou um cluster global ou Restauração de um cluster de banco de dados do Aurora ou um cluster global.

Versões com Suporte estendido do Amazon RDS

O Suporte estendido do RDS está disponível para o Aurora MySQL e o Aurora PostgreSQL. Para obter mais informações, consulte Versões principais do Amazon Aurora.

O Suporte estendido do RDS só está disponível em algumas versões secundárias. As versões secundárias só são indicadas como compatíveis com o Suporte estendido do RDS depois que as versões principais atingem as datas de fim da vida útil da comunidade. Para ter mais informações, consulte Release calendar for Aurora MySQL em Notas de lançamento do Aurora MySQL e Release calendar for Aurora PostgreSQL em Notas de lançamento do Aurora PostgreSQL.

O Suporte estendido do RDS só está disponível no Aurora Serverless v2. Não está disponível no Aurora Serverless v1.

Também é possível visualizar informações sobre datas de suporte para versões de mecanismo usando a AWS CLI ou a API do RDS. Para obter mais informações, consulte Visualizar datas de suporte de versões de mecanismo no Suporte estendido do Amazon RDS.

Responsabilidades do Amazon Aurora e do cliente com o Suporte estendido do Amazon RDS

O conteúdo a seguir descreve as responsabilidades do Amazon Aurora, bem como as suas, com o Suporte estendido do RDS.

Tópicos

- Responsabilidades do Amazon Aurora
- Suas responsabilidades

Responsabilidades do Amazon Aurora

Após a data de término do suporte padrão do Aurora, o Amazon Aurora fornecerá patches, correções de erros e atualizações para mecanismos inscritos no Suporte estendido do RDS. Isso ocorrerá por até três anos ou até que você pare de usar os mecanismos, o que ocorrer primeiro.

Os patches serão para CVEs graves e altos, conforme definido pelas classificações de gravidade CVSS do National Vulnerability Database (NVD). Para obter mais informações, consulte Vulnerability Metrics.

Suas responsabilidades

Você é responsável por aplicar os patches, correções de erros e atualizações fornecidos para , clusters de banco de dados do Aurora ou clusters globais inscritos no Suporte estendido do RDS. O Amazon Aurora reserva-se o direito de alterar, substituir ou retirar esses patches, correções de erros e atualizações a qualquer momento. Caso seja necessário um patch para resolver problemas críticos de segurança ou estabilidade, o Amazon Aurora reserva-se o direito de atualizar clusters de banco de dados do Aurora ou clusters globais com o patch ou exigir que você instale o patch.

Você também é responsável por atualizar o mecanismo para uma versão mais nova antes da data de término do Suporte estendido do RDS. A data de término do Suporte estendido do RDS normalmente é três anos após o fim da vida útil da versão da comunidade. . Em relação à data de término do Suporte estendido do RDS para a versão principal do mecanismo do banco de dados, consulte Versões principais do Amazon Aurora.

Se você não atualizar o mecanismo, após o fim do suporte estendido do RDS, o Amazon Aurora tentará atualizá-lo para a versão mais recente aceita pelo suporte padrão do Aurora. Caso a

atualização falhe, o Amazon Aurora reserva-se o direito de excluir o cluster de banco de dados do Aurora ou o cluster global que está executando o mecanismo após a data de término do suporte padrão do Aurora. No entanto, antes de fazer isso, o Amazon Aurora preservará os dados desse mecanismo.

Criação de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS

Ao criar um cluster de banco de dados do Aurora ou um cluster global, selecione Habilitar RDS Extended Support no console ou use a opção de suporte estendido na AWS CLI ou o parâmetro na API do RDS. Ao inscrever um cluster de banco de dados do Aurora ou um cluster global no Suporte estendido do Amazon RDS, ele é inscrito permanentemente no Suporte estendido do RDS durante a vida do cluster de banco de dados do Aurora ou do cluster global.

Se você usar o console, deverá selecionar Habilitar Suporte estendido do RDS. A configuração não está selecionada por padrão.

Se você usar a AWS CLI ou a API do RDS e não especificar a configuração do Suporte estendido do RDS, o Amazon RDS usará como padrão o Suporte estendido do RDS. Ao automatizar usando o CloudFormation ou outros serviços, esse comportamento padrão mantém a disponibilidade do banco de dados após a data de fim do suporte padrão do Aurora.

É possível impedir a inscrição no Suporte estendido do RDS usando a AWS CLI ou a API do RDS para criar um cluster de banco de dados do Aurora ou um cluster global.

Tópicos

- Comportamento do Suporte estendido do RDS
- Considerações para o Suporte estendido do RDS
- Criar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS

Comportamento do Suporte estendido do RDS

A tabela a seguir resume o que acontece quando uma versão principal do mecanismo atinge o fim do suporte padrão do .

Status do Suporte estendido do RDS*	Comportamento
Habilitado	O Amazon RDS cobra pelo Suporte estendido do RDS.
Desabilitado	O Amazon RDS atualiza um cluster de banco de dados do Aurora ou um cluster global para uma versão do mecanismo compatível. Essa atualização ocorre na data ou logo após a data de fim do suporte padrão do Aurora.

* No console do RDS, o status do Suporte estendido do RDS aparece como Sim ou Não. Na AWS CLI ou na API do RDS, o status do Suporte estendido do RDS é exibido como `open-source-rds-extended-support` ou `open-source-rds-extended-support-disabled`.

Considerações para o Suporte estendido do RDS

Antes de criar um cluster de banco de dados do Aurora ou um cluster global, considere os seguintes itens:

- Depois que a data de fim do suporte padrão do Aurora tiver passado, você poderá impedir a criação de um cluster de banco de dados do Aurora ou um cluster global e evitar cobranças do Suporte estendido do RDS. Para fazer isso, use a AWS CLI ou a API do RDS. Na AWS CLI, especifique `open-source-rds-extended-support-disabled` para a opção `--engine-lifecycle-support`. Na API do RDS, especifique `open-source-rds-extended-support-disabled` para o parâmetro `LifeCycleSupport`. Se você especificar `open-source-rds-extended-support-disabled` e a data de fim do suporte padrão do Aurora já tiver passado, a criação de um cluster de banco de dados do Aurora ou um cluster global sempre falhará.
- O Suporte estendido do RDS é definido no nível do cluster. Os membros de um cluster sempre terão a mesma configuração para o Suporte estendido do RDS no console do RDS, em `--engine-lifecycle-support` na AWS CLI e em `EngineLifecycleSupport` na API do RDS.

Para obter mais informações, consulte Versões do Amazon Aurora.

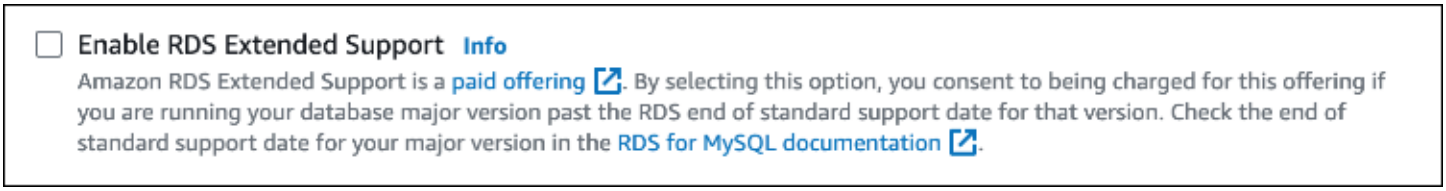
Criar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS

É possível criar um cluster de banco de dados do Aurora ou um cluster global com uma versão do Suporte estendido do RDS usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Ao criar um cluster de banco de dados do Aurora ou um cluster global, selecione Habilitar Suporte estendido do RDS na seção Opções do mecanismo. Por padrão, essa configuração não está selecionada.

A imagem a seguir mostra a configuração Habilitar Suporte estendido do RDS:



AWS CLI

Ao usar o comando create-db-cluster ou create-global-cluster AWS CLI, selecione o Suporte estendido do RDS especificando `open-source-rds-extended-support` para a opção `--engine-lifecycle-support`. Essa opção é definida como por padrão `open-source-rds-extended-support`.

Para evitar a criação de um cluster de banco de dados do Aurora ou um cluster global após a data de fim do suporte padrão do Aurora, especifique `open-source-rds-extended-support-disabled` para a opção `--engine-lifecycle-support`. Ao fazer isso, você evitará cobranças associadas ao Suporte estendido do RDS.

API do RDS

Ao usar a operação de API CreateDBCluster ou CreateGlobalCluster do Amazon RDS, selecione o Suporte estendido do RDS definindo o parâmetro `EngineLifecycleSupport` como `open-source-rds-extended-support`. Esse parâmetro é definido como por padrão `open-source-rds-extended-support`.

Para evitar a criação de um cluster de banco de dados do Aurora ou um cluster global após a data de fim do suporte padrão do Aurora, especifique `open-source-rds-extended-support-disabled`

para o parâmetro `EngineLifecycleSupport`. Ao fazer isso, você evitará cobranças associadas ao Suporte estendido do RDS.

Para saber mais, consulte os seguintes tópicos:

- Para criar um cluster de banco de dados do Aurora, siga as instruções do mecanismo de banco de dados em Criar um cluster de bancos de dados do Amazon Aurora.
- Para criar um cluster global, siga as instruções relacionadas ao mecanismo de banco de dados em Criar um banco de dados global do Amazon Aurora.

Visualizar a inscrição de clusters de banco de dados do Aurora ou clusters globais no Suporte estendido do Amazon RDS

É possível visualizar a inscrição de clusters de banco de dados do Aurora ou clusters globais no Suporte estendido do RDS usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

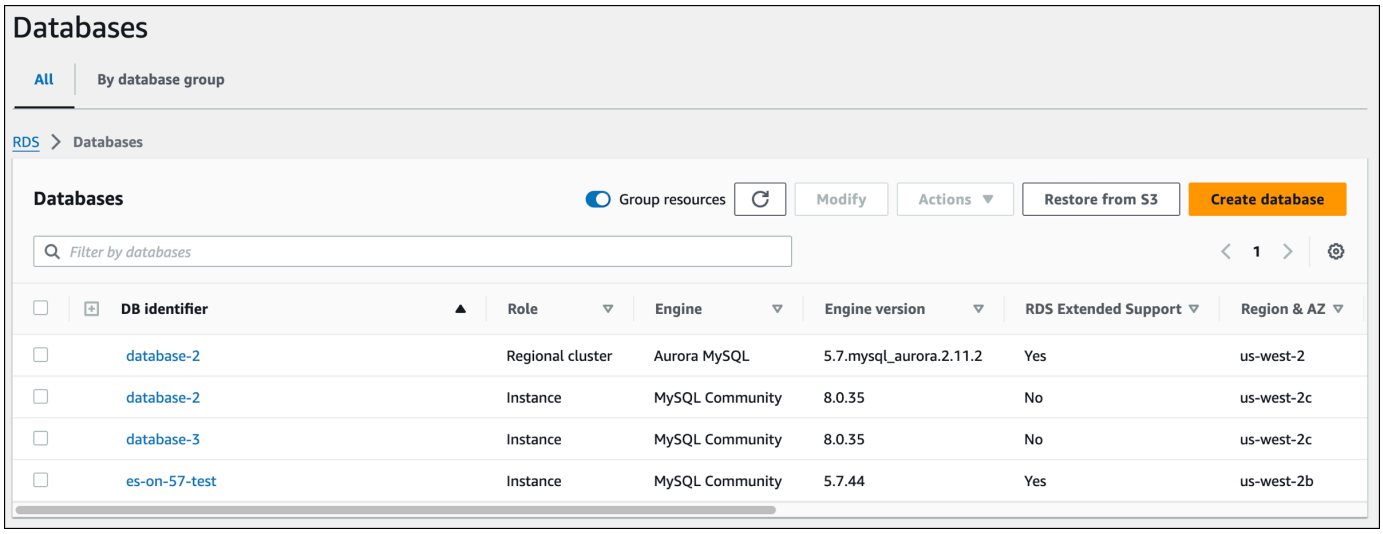
> **Note**
>
> A coluna Suporte estendido do RDS no console, a opção `--engine-lifecycle-support` na AWS CLI e o parâmetro `EngineLifecycleSupport` na API do RDS indicam somente a inscrição no Suporte estendido do RDS. As cobranças pelo Suporte estendido do RDS só começam quando sua versão do mecanismo de banco de dados atinge o final do suporte padrão do Aurora. Para obter mais informações, consulte Versões principais do Amazon Aurora.
>
> Por exemplo, você tem um banco de dados do Aurora PostgreSQL 11 que está inscrito no Suporte estendido do RDS. Em 1º de abril de 2024, o Amazon RDS começou a cobrar pelo Suporte estendido do RDS para esse banco de dados. Em 31 de julho de 2024, você atualizou esse banco de dados para o Aurora PostgreSQL 12. O status do Suporte estendido do RDS para esse banco de dados permanece habilitado. No entanto, as cobranças do Suporte estendido do RDS para esse banco de dados foram interrompidas porque o Aurora PostgreSQL 12 ainda não atingiu o fim do suporte padrão do RDS. O Amazon RDS não cobrará pelo Suporte estendido do RDS para esse banco de dados até 1º de março de 2025, quando o suporte padrão do Aurora terminará para o Aurora PostgreSQL 12.

Console

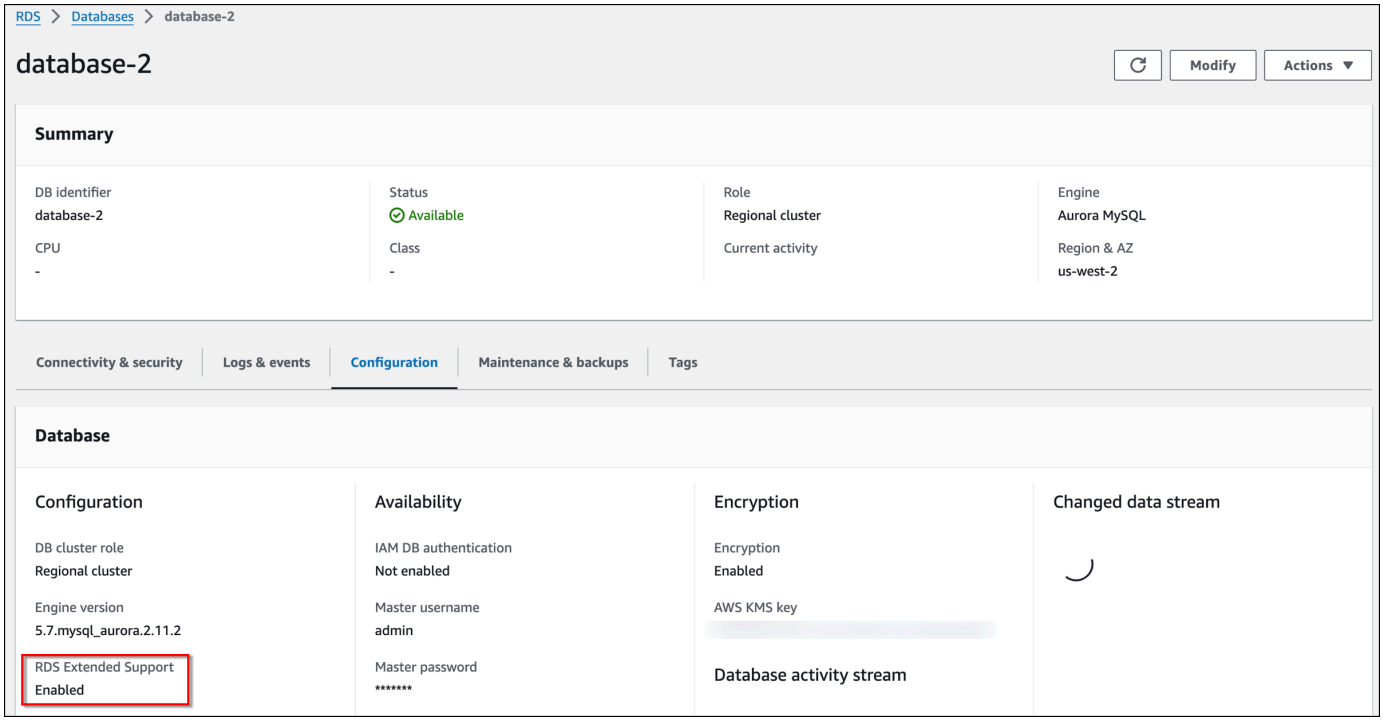
Para visualizar a inscrição de clusters de banco de dados do Aurora ou clusters globais no Suporte estendido do RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados. O valor em Suporte estendido do RDS indica se foi feita a inscrição de um cluster de banco de dados do Aurora ou um cluster global no Suporte estendido do RDS. Se nenhum valor for exibido, o Suporte estendido do RDS não estará disponível para o banco de dados.

> **Tip**
>
> Se a coluna Suporte estendido do RDS não aparecer, selecione o ícone Preferências e ative Suporte estendido do RDS.



3. Você também pode visualizar a inscrição na guia Configuração de cada banco de dados. Escolha um banco de dados em Identificador de banco de dados. Na guia Configuração, confira em Suporte estendido se o banco de dados está inscrito ou não.



AWS CLI

Para visualizar a inscrição de seus bancos de dados no Suporte estendido do RDS usando a AWS CLI, execute o comando describe-db-clusters ou describe-global-clusters.

Se o Suporte estendido do RDS estiver disponível para um banco de dados, a resposta incluirá o parâmetro `EngineLifecycleSupport`. O valor `open-source-rds-extended-support` indica que um cluster de banco de dados ou cluster global do Aurora está inscrito no Suporte estendido do RDS. O valor `open-source-rds-extended-support-disabled` indica que a inscrição do cluster de banco de dados ou cluster global do Aurora no Suporte estendido do RDS foi desabilitada.

Exemplo

O comando a seguir retorna informações de todos os clusters de banco de dados do Aurora:

```
aws rds describe-db-clusters
```

A resposta a seguir mostra que um mecanismo do Aurora PostgreSQL em execução no cluster de banco de dados do Aurora `database-1` está inscrito no Suporte estendido do RDS:

```
{
    "DBClusterIdentifier": "database-1",
```

```
    ...
    "Engine": "aurora-postgresql",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

API do RDS

Para visualizar a inscrição dos bancos de dados no Suporte estendido do RDS utilizando a API do Amazon RDS, use a operação DescribeDBClusters ou DescribeGlobalClusters.

Se o Suporte estendido do RDS estiver disponível para um banco de dados, a resposta incluirá o parâmetro `EngineLifecycleSupport`. O valor `open-source-rds-extended-support` indica que um cluster de banco de dados ou cluster global do Aurora está inscrito no Suporte estendido do RDS. O valor `open-source-rds-extended-support-disabled` indica que a inscrição do cluster de banco de dados ou cluster global do Aurora no Suporte estendido do RDS foi desabilitada.

Visualizar datas de suporte de versões de mecanismo no Suporte estendido do Amazon RDS

É possível visualizar informações sobre datas de suporte para versões de mecanismo de clusters de banco de dados do Aurora ou clusters globais no Suporte estendido do Amazon RDS usando a AWS CLI ou a API do RDS. Essas informações podem ajudar você a planejar as atualizações.

Os comandos da AWS CLI e as operações de API do RDS retornam as datas de início e término do suporte padrão do Aurora e do Suporte estendido do RDS. Essas datas também podem ser encontradas nas tabelas de versões principais do mecanismo. Para ter mais informações, consulte Versões principais do Amazon Aurora.

AWS CLI

Para visualizar as datas de início e término do suporte padrão do Aurora e do Suporte estendido do RDS para versões principais do mecanismo usando a AWS CLI, execute o comando describe-db-major-engine-versions.

Este comando retorna os seguintes parâmetros relevantes:

- `SupportedEngineLifecycles`: este parâmetro é uma matriz de objetos que inclui `LifecycleSupportName`, `LifecycleSupportStartDate` e `LifecycleSupportEndDate`.

- `LifecycleSupportName` – Esse parâmetro indica o tipo de suporte em que a versão do mecanismo está: suporte padrão do Aurora (`open-source-rds-standard-support`) ou RDS Extended Support (`open-source-rds-extended-support`).
- `LifecycleSupportStartDate`: este parâmetro lista a data de início do suporte padrão do Aurora ou do Suporte estendido do RDS para a versão principal do mecanismo, dependendo do valor de `LifecycleSupportName`.
- `LifecycleSupportEndDate`: este parâmetro lista a data de término do suporte padrão do Aurora ou do Suporte estendido do RDS para a versão principal do mecanismo, dependendo do valor de `LifecycleSupportName`.

Exemplo

O exemplo de resposta mostra as datas de início e término dos ciclos de vida `open-source-rds-standard-support` e `open-source-rds-extended-support` do mecanismo compatível para Aurora MySQL versão 2 (MySQL 5.7). O suporte estendido do RDS está disponível para o Aurora MySQL versão 2 (MySQL 5.7).

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "aurora-mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2018-02-06T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-10-31T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-11-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        ...
    ]
}
```

API do RDS

Para ver as datas de início e término do suporte padrão do Aurora e do Suporte estendido do RDS para as versões principais do mecanismo usando a API do RDS, use a operação DescribeDBMajorEngineVersions.

Se o Suporte estendido do RDS estiver disponível para uma versão do mecanismo, a resposta incluirá o parâmetro `SupportedEngineLifeCycles` como uma matriz com dois objetos. O primeiro objeto inclui as datas de início e término do suporte padrão do Aurora. O segundo inclui as datas de início e término do Suporte estendido do RDS.

Se o Suporte estendido do RDS não estiver disponível para uma versão do mecanismo, a resposta incluirá somente parâmetro `SupportedEngineLifeCycles` como uma matriz com único objeto. Esse objeto inclui as datas de início e término do suporte padrão do Aurora.

Restauração de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS

Ao restaurar um cluster de banco de dados do Aurora ou um cluster global, selecione Habilitar RDS Extended Support no console ou use a opção de suporte estendido na AWS CLI ou o parâmetro na API do RDS. Ao inscrever um cluster de banco de dados do Aurora ou um cluster global no Suporte estendido do RDS, ele é inscrito permanentemente no Suporte estendido do RDS durante a vida do cluster de banco de dados do Aurora ou do cluster global.

O padrão para a configuração do Suporte estendido do RDS depende de você usar o console, a AWS CLI ou API do RDS para restaurar o banco de dados. Se você usa o console, não seleciona Habilitar Suporte estendido do RDS, e a versão principal do mecanismo que você está restaurando já passou do fim do suporte padrão do Aurora. Então, o Amazon Aurora atualiza automaticamente a instância de banco de dados para uma versão mais recente do mecanismo. Se você usar a AWS CLI ou a API do RDS e não especificar a configuração do Suporte estendido do RDS, o Amazon RDS usará como padrão o Suporte estendido do RDS. Ao automatizar usando o CloudFormation ou outros serviços, esse comportamento padrão mantém a disponibilidade do banco de dados após a data de fim do suporte padrão do Aurora. É possível desabilitar o Suporte estendido do RDS usando a AWS CLI ou a API do RDS.

Tópicos

- Comportamento do Suporte estendido do RDS
- Considerações para o Suporte estendido do RDS

- Restaurar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS

Comportamento do Suporte estendido do RDS

A tabela a seguir resume o que acontece quando uma versão principal do mecanismo de um cluster de banco de dados do Aurora ou um cluster global que você está restaurando atinge o fim do suporte padrão do Aurora.

Status do Suporte estendido do RDS*	Comportamento
Habilitado	O Amazon RDS cobra pelo Suporte estendido do RDS.
Desabilitado	Após a conclusão da restauração, o Amazon RDS atualiza automatic amente o cluster de banco de dados do Aurora ou o cluster global para uma versão mais recente do mecanismo (em uma futura janela de manutenção).

* No console do RDS, o status do Suporte estendido do RDS aparece como Sim ou Não. Na AWS CLI ou na API do RDS, o status do Suporte estendido do RDS é exibido como `open-source-rds-extended-support` ou `open-source-rds-extended-support-disabled`.

Considerações para o Suporte estendido do RDS

Antes de restaurar um cluster de banco de dados do Aurora ou um cluster global, considere os seguintes itens:

- Depois que a data de término do suporte padrão do Aurora tiver passado, se quiser restaurar um cluster de banco de dados do Aurora ou um cluster global pelo Amazon S3, você só poderá fazer isso usando a AWS CLI ou a API do RDS. Use a opção `--engine-lifecycle-support` no comando restore-db-cluster-from-s3 da AWS CLI ou o parâmetro `EngineLifecycleSupport` na operação de API RestoreDBClusterFromS3 do RDS.
- Se você quiser impedir que o Aurora restaure os bancos de dados para as versões do Suporte estendido do RDS, especifique `open-source-rds-extended-support-disabled` na AWS CLI ou na API do RDS. Ao fazer isso, você evitará cobranças associadas ao Suporte estendido do RDS.

Se você especificar essa configuração, o Amazon Aurora atualizará automaticamente o banco de dados restaurado para uma versão principal compatível mais recente. Se o upgrade falhar nas verificações pré-upgrade, o Amazon Aurora voltará com segurança para a versão do mecanismo do Suporte estendido do RDS. Esse banco de dados permanecerá no modo de Suporte estendido do RDS e o Amazon Aurora cobrará pelo Suporte estendido do RDS até que você faça upgrade manual do banco de dados.

- O Suporte estendido do RDS é definido no nível do cluster. Os membros de um cluster sempre terão a mesma configuração para o Suporte estendido do RDS no console do RDS, em `--engine-lifecycle-support` na AWS CLI e em `EngineLifecycleSupport` na API do RDS.

Para obter mais informações, consulte Versões do Amazon Aurora.

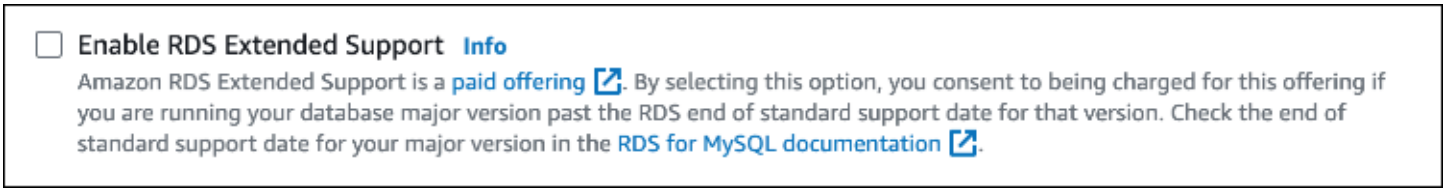
Restaurar um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do RDS

É possível restaurar um cluster de banco de dados do Aurora ou um cluster global com uma versão do Suporte estendido do RDS usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Ao restaurar um cluster de banco de dados do Aurora ou um cluster global, selecione Habilitar RDS Extended Support na seção Opções do mecanismo. Se você não selecionar essa configuração e a versão principal do mecanismo que você está restaurando tiver passado do fim do suporte padrão do Aurora, o Amazon Aurora atualizará automaticamente o cluster de banco de dados do Aurora ou o cluster global para uma versão sob o suporte padrão do Aurora.

A imagem a seguir mostra a configuração Habilitar Suporte estendido do RDS:



AWS CLI

Ao usar o comando restore-db-cluster-from-snapshot da AWS CLI, selecione o Suporte estendido do RDS especificando `open-source-rds-extended-support` para a opção `--engine-lifecycle-support`.

Se você quiser evitar cobranças associadas ao Suporte estendido do RDS, defina a opção `--engine-lifecycle-support` como `open-source-rds-extended-support-disabled`. Essa opção é definida como por padrão `open-source-rds-extended-support`.

Também é possível especificar esse valor usando os seguintes comandos da AWS CLI:

- restore-db-cluster-from-s3
- restore-db-cluster-to-point-in-time

API do RDS

Ao usar a operação de API RestoreDBClusterFromSnapshot do Amazon RDS, selecione o Suporte estendido do RDS definindo o parâmetro `EngineLifecycleSupport` como `open-source-rds-extended-support`.

Se você quiser evitar cobranças associadas ao Suporte estendido do RDS, defina o parâmetro `EngineLifecycleSupport` como `open-source-rds-extended-support-disabled`. Esse parâmetro é definido como por padrão `open-source-rds-extended-support`.

Também é possível especificar esse valor usando as seguintes operações de API do RDS:

- RestoreDBClusterFromS3
- RestoreDBClusterToPointInTime

Consulte mais informações sobre a restauração de um cluster de banco de dados do Aurora e siga as instruções do mecanismo de banco de dados em Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora.

Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados

Uma implantação azul/verde copia um ambiente de banco de dados de produção em um ambiente de teste separado e sincronizado. Usando as implantações azul/verde do Amazon Aurora, é possível fazer alterações no banco de dados no ambiente de teste sem prejudicar o ambiente de produção. Por exemplo, você pode atualizar a versão principal ou secundária do mecanismo de banco de dados ou alterar parâmetros do banco de dados no ambiente de preparação. Quando estiver tudo pronto, você poderá promover o ambiente de teste como o novo ambiente de banco de dados de produção, com tempo de inatividade normalmente inferior a um minuto.

O Amazon Aurora cria o ambiente de preparação clonando o volume de armazenamento subjacente do Aurora no ambiente de produção. O volume do cluster no ambiente de preparação armazena somente as alterações incrementais feitas nesse ambiente.

> **Note**
>
> No momento, as implantações azul/verde são compatíveis apenas com o Aurora MySQL, Aurora PostgreSQL e Aurora Global Database. Para saber a disponibilidade do mecanismo do Amazon RDS, consulte Using Amazon RDS Blue/Green Deployments for database updates no Guia do usuário do Amazon RDS.

Tópicos

- Visão geral das implantações azul/verde do Amazon Aurora
- Criar uma implantação azul/verde no Amazon Aurora
- Visualizar uma implantação azul/verde no Amazon Aurora
- Mudar uma implantação azul/verde no Amazon Aurora
- Excluir uma implantação azul/verde no Amazon Aurora

Visão geral das implantações azul/verde do Amazon Aurora

Ao usar implantações azul/verde do Amazon Aurora, é possível fazer e testar alterações no banco de dados antes de implementá-las em um ambiente de produção. Uma implantação azul/verde cria um ambiente de teste que copia o ambiente de produção. Em uma implantação azul/verde, o ambiente azul é o ambiente de produção atual. O ambiente verde é o ambiente de preparação e permanece sincronizado com o ambiente de produção atual.

Você pode fazer alterações no cluster de banco de dados do RDS no ambiente verde sem afetar as workloads de produção. Por exemplo, você pode atualizar a versão principal ou secundária do mecanismo de banco de dados ou alterar os parâmetros do banco de dados no ambiente de preparação. Você pode testar minuciosamente as alterações no ambiente verde. Quando estiver tudo pronto, você poderá fazer a transição do ambiente verde para o novo ambiente de produção. A transição normalmente leva menos de um minuto, sem perda de dados e sem necessidade de alterações na aplicação.

Como o ambiente verde é uma cópia da topologia do ambiente de produção, o cluster de banco de dados e todas as suas instâncias de banco de dados são copiados na implantação. O ambiente verde também inclui os recursos usados pelo cluster de banco de dados, como snapshots do cluster de banco de dados, Performance Insights, monitoramento aprimorado e o Aurora Serverless v2.

> **Note**
>
> Implantações azul/verde são aceitas pelo Aurora MySQL, o Aurora PostgreSQL e o Aurora Global Database. Para ter informações sobre a disponibilidade do RDS, consulte Visão geral das implantações azul/verde do Amazon RDS no Guia do usuário do Amazon RDS.

Tópicos

- Disponibilidade de região e versão
- Benefícios do uso de implantações azul/verde do Amazon RDS
- Fluxo de trabalho de uma implantação azul/verde
- Autorizar o acesso às operações de implantação azul/verde do Amazon Aurora
- Limitações e considerações relativas às implantações azul/verde do Amazon Aurora
- Práticas recomendadas para implantações azul/verde do Amazon Aurora

Disponibilidade de região e versão

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para obter mais informações, consulte the section called “Implantações azul/verde”.

Benefícios do uso de implantações azul/verde do Amazon RDS

Ao usar implantações azul/verde do Amazon RDS, você pode se manter atualizado sobre os patches de segurança, melhorar a performance do banco de dados e adotar novos recursos de banco de dados com um tempo de inatividade curto e previsível. As implantações azul/verde reduzem os riscos e o tempo de inatividade das atualizações do banco de dados, como atualizações principais ou secundárias de versões do mecanismo.

As implantações azul/verde oferecem os seguintes benefícios:

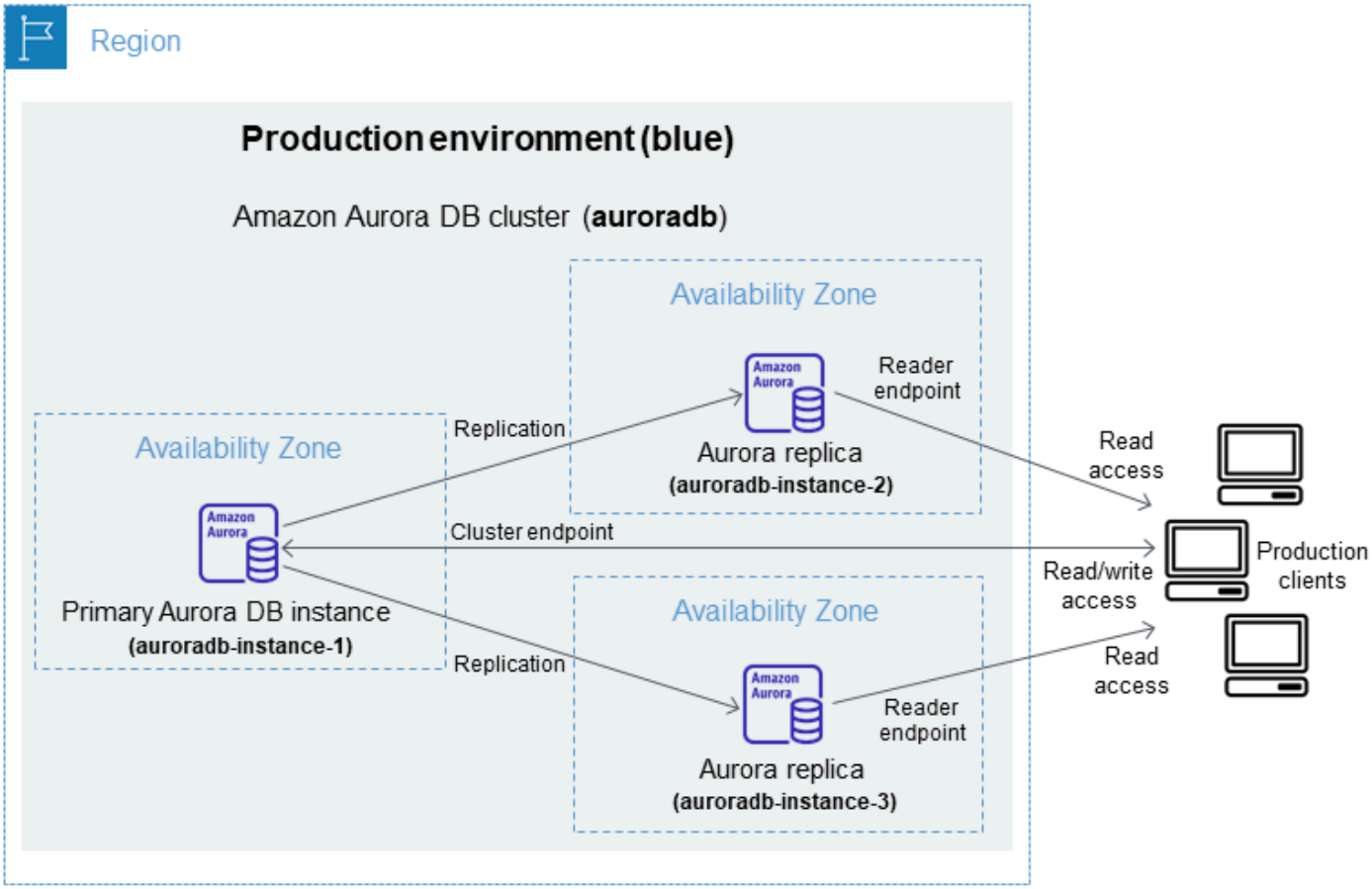
- Crie facilmente um ambiente de teste pronto para produção.
- Replique automaticamente as alterações do banco de dados do ambiente de produção para o ambiente de teste.
- Teste as alterações do banco de dados em um ambiente de teste seguro sem afetar o ambiente de produção.
- Mantenha-se atualizado com os patches do banco de dados e as atualizações do sistema.
- Implemente e teste novos recursos de banco de dados.
- Faça a transição de seu ambiente de teste para ser o novo ambiente de produção sem alterações em sua aplicação.
- Faça a transição com segurança por meio do uso de grades de proteção de transição integradas.
- Elimine a perda de dados durante a transição.
- Faça a transição rapidamente, normalmente em menos de um minuto, dependendo da sua workload.

Fluxo de trabalho de uma implantação azul/verde

Conclua as etapas principais a seguir ao usar uma implantação azul/verde para atualizações do cluster de banco de dados do Aurora.

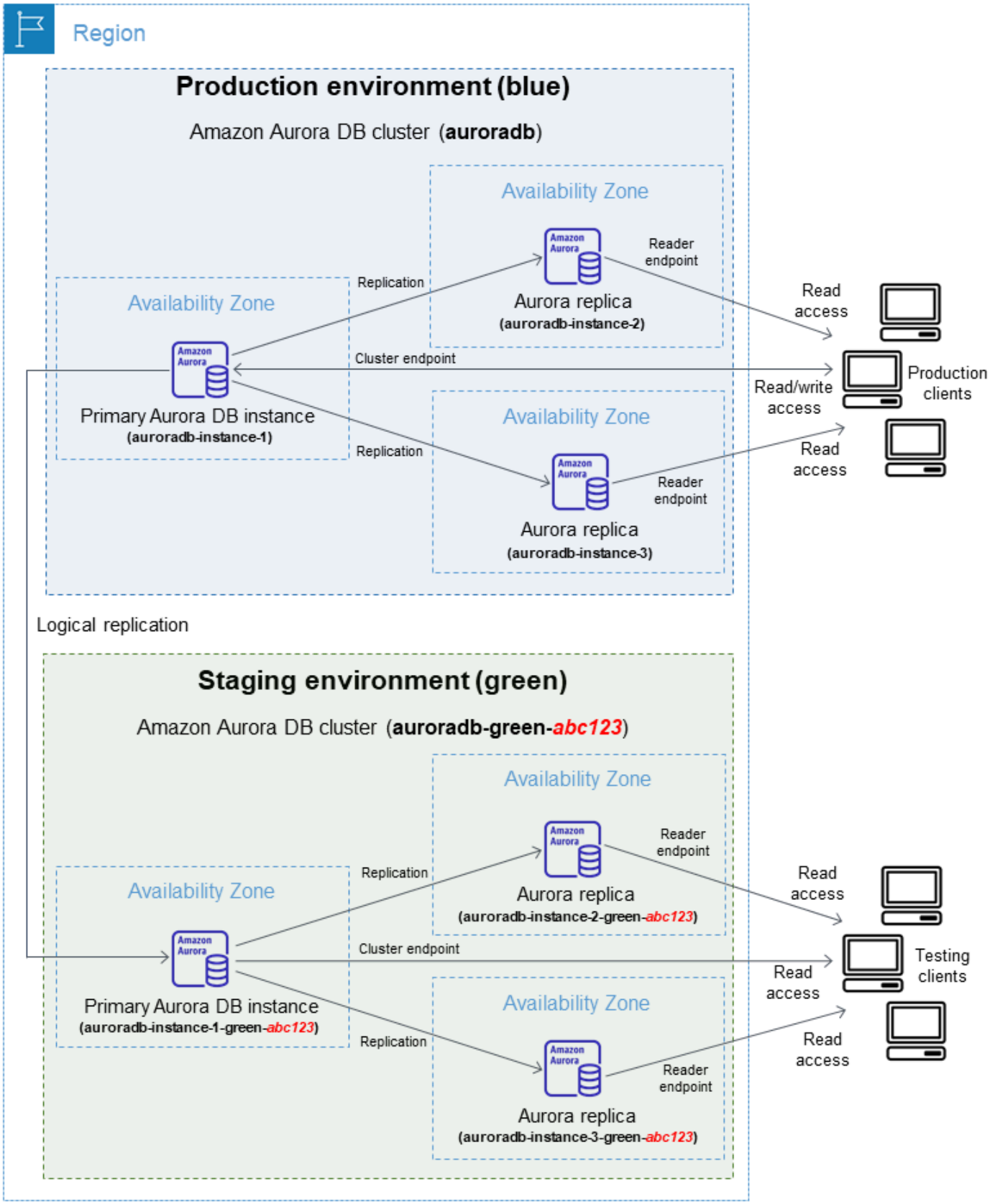
1. Identifique um cluster de banco de dados de produção que exija atualizações.

A imagem a seguir mostra um exemplo de cluster de banco de dados de produção.



2. Crie a implantação azul/verde Para instruções, consulte Criar uma implantação azul/verde no Amazon Aurora.

   A imagem a seguir mostra um exemplo de implantação azul/verde do ambiente de produção da etapa 1. Ao criar a implantação azul/verde, o RDS copia a topologia e a configuração completas do cluster de banco de dados do Aurora para criar o ambiente verde. Os nomes do cluster e das instâncias de banco de dados copiados são anexados com `-green-random-characters`. O ambiente de teste na imagem contém o cluster de banco de dados (auroradb-green-*abc123*). Ele também contém as três instâncias de banco de dados no cluster de banco de dados (auroradb-instance1-green-*abc123*, auroradb-instance2-green-*abc123* e auroradb-instance3-green-*abc123*).


Region
Production environment (blue)
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2)
Replication
Availability Zone
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Logical replication
Staging environment (green)
Amazon Aurora DB cluster (auroradb-green-abc123)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2-green-abc123)
Replication
Availability Zone
Read access
Cluster endpoint
Testing clients
Read access
Primary Aurora DB instance
(auroradb-instance-1-green-abc123)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3-green-abc123)

Ao criar a implantação azul/verde, você pode especificar uma versão do mecanismo de banco de dados posterior e um grupo de parâmetros de banco de dados diferente para o cluster de banco de dados no ambiente verde. Você também pode especificar um grupo de parâmetros de banco de dados diferente para as instâncias de banco de dados no cluster de banco de dados.

O RDS também configura a replicação da instância de banco de dados primária no ambiente azul para a instância de banco de dados primária no ambiente verde.

> **Important**
>
> Em relação ao Aurora MySQL versão 3, depois de criar a implantação azul/verde, o cluster de banco de dados no ambiente verde permite operações de gravação por padrão. No entanto, isso não se aplica aos usuários que têm o privilégio `CONNECTION_ADMIN`, incluindo o usuário principal do Aurora. Usuários com esse privilégio podem ignorar o comportamento `read_only`. Para obter mais informações, consulte Modelo de privilégios baseados em funções.

3. Faça as alterações no ambiente de teste.

   Por exemplo, você pode alterar a classe da instância de banco de dados usada por uma ou mais instâncias de banco de dados no ambiente verde.

   Para ter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

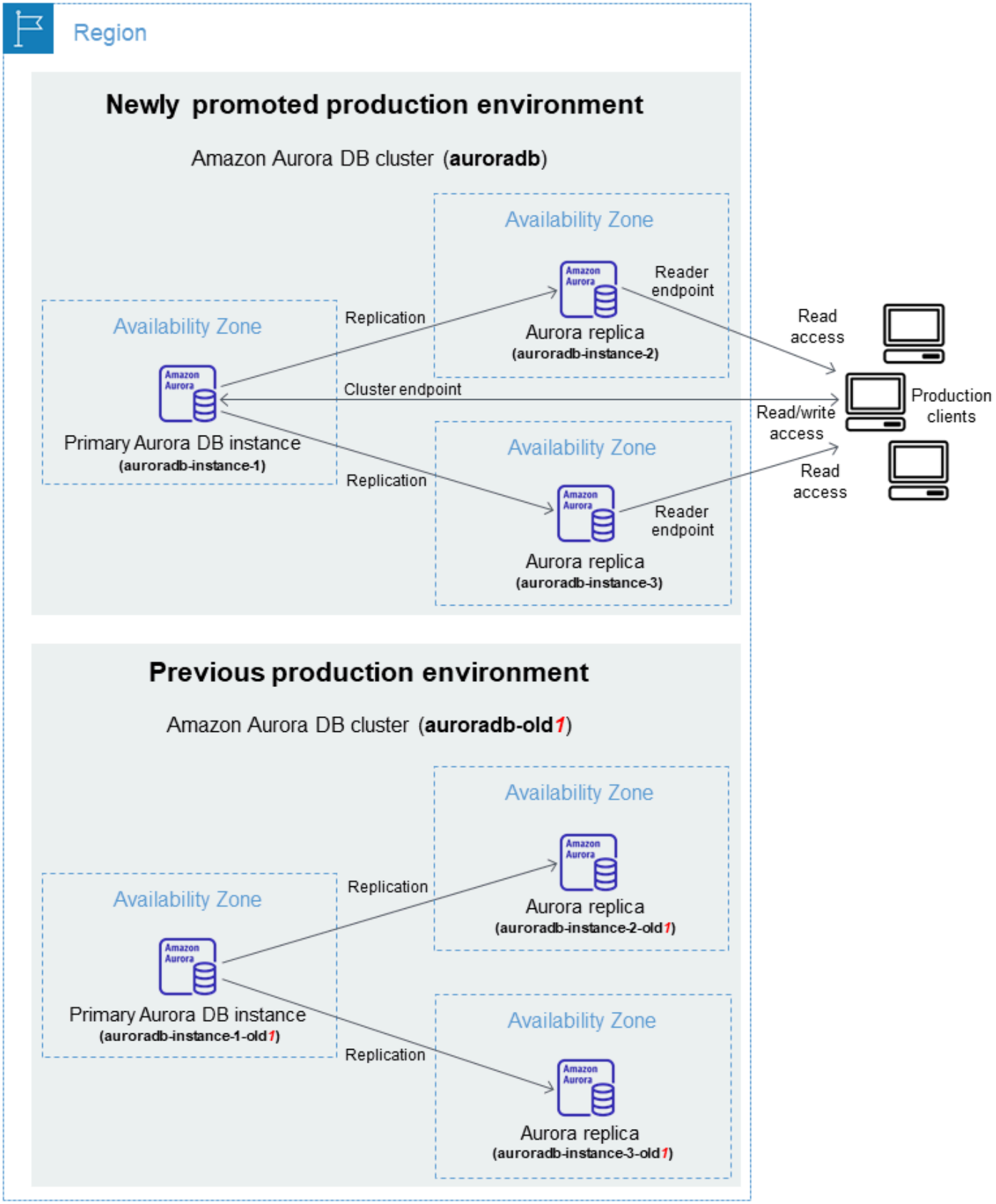
4. Teste seu ambiente de teste.

   Durante o teste, recomendamos que você mantenha seus bancos de dados no ambiente verde somente leitura. Habilite operações de gravação no ambiente verde com cuidado, pois elas podem causar conflitos de replicação. Elas também podem ocasionar dados não intencionais nos bancos de dados de produção após a transição. Para habilitar as operações de gravação para o Aurora MySQL, defina o parâmetro `read_only` como `0` e reinicialize a instância de banco de dados. Para o Aurora PostgreSQL, defina o parâmetro `default_transaction_read_only` como `off` no nível da sessão.

5. Quando estiver tudo pronto, faça a transição para promover o ambiente de teste para o novo ambiente de produção. Para instruções, consulte Mudar uma implantação azul/verde no Amazon Aurora.

A transição ocasiona tempo de inatividade. O tempo de inatividade geralmente é inferior a um minuto, mas pode ser maior dependendo de sua workload.

A imagem a seguir mostra os clusters de banco de dados após a transição.


Region
Newly promoted production environment
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Availability Zone
Replication
Aurora replica
(auroradb-instance-2)
Read access
Amazon Aurora
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-3)
Previous production environment
Amazon Aurora DB cluster (auroradb-old1)
Availability Zone
Availability Zone
Replication
Amazon Aurora
Aurora replica
(auroradb-instance-2-old1)
Primary Aurora DB instance
(auroradb-instance-1-old1)
Availability Zone
Replication
Amazon Aurora
Aurora replica
(auroradb-instance-3-old1)

Após a transição, o cluster de banco de dados do Aurora no ambiente verde se torna o novo cluster de banco de dados de produção. Os nomes e os endpoints no ambiente de produção atual são atribuídos ao ambiente de produção recém-transicionado, sem exigir alterações na aplicação. Como resultado, seu tráfego de produção agora flui para o novo ambiente de produção. O cluster e as instâncias de banco de dados no ambiente azul são renomeados anexando `-old`*n* ao nome atual, em que *n* é um número. Por exemplo, suponha que o nome da instância de banco de dados no ambiente azul seja `auroradb-instance-1`. Após a transição, o nome da instância de banco de dados pode ser `auroradb-instance-1-old1`.

No exemplo da imagem, as seguintes alterações ocorrem durante a alternância:

- O cluster de banco de dados do ambiente verde `auroradb-green-abc123` torna-se o cluster de banco de dados de produção chamado `auroradb`.
- A instância de banco de dados do ambiente verde denominada `auroradb-instance1-green-abc123` torna-se a instância de banco de dados de produção denominada `auroradb-instance1`.
- A instância de banco de dados do ambiente verde denominada `auroradb-instance2-green-abc123` torna-se a instância de banco de dados de produção denominada `auroradb-instance2`.
- A instância de banco de dados do ambiente verde denominada `auroradb-instance3-green-abc123` torna-se a instância de banco de dados de produção denominada `auroradb-instance3`.
- O cluster de banco de dados do ambiente azul chamado `auroradb` torna-se `auroradb-old1`.
- A instância de banco de dados do ambiente azul chamado `auroradb-instance1` torna-se `auroradb-instance1-old1`.
- A instância de banco de dados do ambiente azul chamado `auroradb-instance2` torna-se `auroradb-instance2-old1`.
- A instância de banco de dados do ambiente azul chamado `auroradb-instance3` torna-se `auroradb-instance3-old1`.

6. Caso não precise mais de uma implantação azul/verde, você pode excluí-la. Para instruções, consulte Excluir uma implantação azul/verde no Amazon Aurora.

   Após a transição, o ambiente de produção anterior não é excluído para que você possa usá-lo para testes de regressão, se necessário.

Autorizar o acesso às operações de implantação azul/verde do Amazon Aurora

Os usuários devem ter as permissões necessárias para realizar operações relacionadas às implantações azul/verde. É possível criar políticas do IAM que concedam aos usuários e perfis permissão para executar operações de API específicas nos recursos especificados de que precisam. Depois, você pode anexar essas políticas aos conjuntos de permissões do IAM ou às funções que exigem essas permissões. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

O usuário que cria uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterEndpoint`

O usuário que faz a transição de uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBCluster`
- `rds:PromoteReadReplicaDBCluster`

O usuário que exclui uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBCluster`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterEndpoint`

O Aurora provisiona e modifica recursos no ambiente de preparação em seu nome. Esses recursos incluem instâncias de banco de dados que usam uma convenção de nomenclatura definida internamente. Portanto, as políticas do IAM anexadas não podem conter padrões parciais de nomes de recursos, como `my-db-prefix-*`. Somente curingas (*) são compatíveis. Em geral, recomendamos o uso de tags de recursos e outros atributos compatíveis para controlar o acesso a esses recursos, em vez do uso de curingas. Consulte mais informações em Actions, resources, and condition keys for Amazon RDS.

Permissões adicionais para implantações azul/verde do Aurora Global Database

Ao criar implantações azul/verde para clusters do Aurora Global Database, além da permissão listada acima, os usuários precisam das permissões a seguir a fim de realizar operações para gerenciar a topologia global do cluster.

O usuário que cria uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:CreateGlobalCluster`

O usuário que faz a transição de uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:ModifyGlobalCluster`
- `rds:PromoteReadReplicaDBCluster`

O usuário que exclui uma implantação azul/verde deve ter permissões para realizar as seguintes operações do RDS:

- `rds:DeleteGlobalCluster`

Limitações e considerações relativas às implantações azul/verde do Amazon Aurora

Implantações azul/verde no Amazon RDS exigem uma análise cuidadosa de fatores, como slots de replicação, gerenciamento de recursos, dimensionamento de instâncias e possíveis impactos na performance do banco de dados. As seções a seguir fornecem orientação para ajudar a otimizar

a estratégia de implantação para garantir o mínimo de tempo de inatividade, transições perfeitas e gerenciamento eficaz do ambiente de banco de dados.

Tópicos

- Limitações para implantações azul/verde
- Limitações do Aurora Global Database para implantações azul/verde
- Considerações sobre implantações azul/verde

Limitações para implantações azul/verde

As seguintes limitações se aplicam às implantações azul/verde:

Tópicos

- Limitações para implantações azul/verde
- Limitações do Aurora MySQL em implantações azul/verde
- Limitações do Aurora PostgreSQL em implantações azuis/verdes

Limitações para implantações azul/verde

As seguintes limitações se aplicam às implantações azul/verde:

- Não é possível interromper e iniciar um cluster que faça parte de uma implantação azul/verde.
- As implantações azul/verde não são compatíveis com o gerenciamento de senhas de usuário principal com AWS Secrets Manager.
- Se você tentar forçar um retrocesso no cluster de banco de dados azul, a implantação azul/verde será interrompida e a transição será bloqueada.
- Durante a transição, os ambientes azul e verde não podem ter integrações ETL zero com o Amazon Redshift. Você deve excluir a integração primeiro, alternar e, depois, recriar a integração.
- O Agendador de Eventos (parâmetro `event_scheduler`) deve ser desativado no ambiente verde ao criar uma implantação azul/verde. Isso impede que eventos sejam gerados no ambiente verde e causem inconsistências.
- As políticas de ajuste de escala automático configuradas no cluster de banco de dados azul não serão copiadas para o ambiente verde. Você deve reconfigurá-las após a transição, independentemente de terem sido inicialmente configuradas no ambiente azul ou verde.

- Você não pode alterar um cluster de banco de dados não criptografado em um cluster de banco de dados não criptografado. Além disso, não é possível alterar um cluster de banco de dados não criptografado em um cluster de banco de dados não criptografado.
- Não é possível alterar um cluster de banco de dados azul para uma versão de mecanismo posterior ao cluster de banco de dados verde correspondente.
- Os recursos no ambiente azul e no ambiente verde devem estar na mesma Conta da AWS.
- As implantações azul/verde não são compatíveis com os seguintes recursos:
  - Proxy do Amazon RDS
  - Réplicas de leitura entre regiões
  - Aurora Serverless v1Clusters de banco de dados do
  - CloudFormation

Limitações do Aurora MySQL em implantações azul/verde

As seguintes limitações se aplicam às implantações azul/verde do Aurora MySQL:

- O cluster de banco de dados de origem não pode conter nenhum banco de dados denominado `tmp`. Bancos de dados com esse nome não serão copiados no ambiente verde.
- O cluster de banco de dados azul não pode ser uma réplica externa de log binário.
- Se o cluster de banco de dados de origem tiver o retrocesso habilitado, o cluster de banco de dados verde será criado sem compatibilidade com retrocesso. Isso ocorre porque o retrocesso não funciona com a replicação de log binário (binlog), que é necessária para implantações azul/verde. Para obter mais informações, consulte the section called “Retroceder um cluster de banco de dados”.
- As implantações azul/verde não são compatíveis com o driver JDBC da AWS para MySQL. Consulte mais informações em Known Limitations no GitHub.

Limitações do Aurora PostgreSQL em implantações azuis/verdes

As seguintes limitações do se aplicam às implantações azuis/verdes do Aurora PostgreSQL .

- As tabelas não registradas não são replicadas no ambiente verde, a menos que o parâmetro `rds.logically_replicate_unlogged_tables` esteja definido como 1 no cluster de banco de dados azul. Não modifique esse valor de parâmetro depois de criar uma implantação azul/verde para evitar possíveis erros de replicação em tabelas não registradas em log.

- O cluster de banco de dados azul não pode ser uma origem lógica (publicador) nem uma réplica (assinante).
- Se o cluster de banco de dados azul estiver configurado como o servidor externo de uma extensão de invólucro de dados externo (FDW), você deverá usar o nome do endpoint do cluster em vez dos endereços IP. Isso permite que a configuração permaneça funcional após a transição.
- Em uma implantação azul/verde, cada banco de dados exige um slot de replicação lógica. À medida que o número de bancos de dados aumenta, a sobrecarga de recursos aumenta e pode gerar um atraso na replicação, principalmente se o cluster de banco de dados não estiver suficientemente escalado. O impacto depende de fatores, como a workload de bancos de dados e o número de conexões. Para atenuar isso, pense em escalar sua classe de instância de banco de dados ou reduzir o número de bancos de dados no cluster de origem.
- As implantações azul/verde são compatíveis com o Babelfish para Aurora PostgreSQL somente na versão 15.7, ou versões 15 posteriores, e na versão 16.3, ou versões 16 posteriores.
- Se quiser capturar planos de execução nas réplicas do Aurora, você deve fornecer o endpoint azul do cluster de banco de dados ao chamar a função. `apg_plan_mgmt.create_replica_plan_capture` Isso garante que as capturas do plano continuem funcionando após a transição. Para obter mais informações, consulte the section called “Capturar planos de execução nas réplicas do Aurora PostgreSQL”.
- O processo de aplicação da replicação lógica no ambiente verde é de thread único. Se o ambiente azul gerar um alto volume de tráfego de gravação, o ambiente verde talvez não consiga acompanhar o ritmo. Isso pode causar atraso ou falha na replicação, especialmente para workloads que produzem um throughput de gravação alto e contínuo. Teste minuciosamente suas workloads. Em cenários que exigem atualizações de versão principal e processamento de workloads com alto volume de gravação, considere abordagens alternativas, como usar o AWS Database Migration Service (AWS DMS) ou a replicação lógica autogerenciada.
- Não é possível criar outras partições em tabelas particionadas durante implantações azul/verde do Aurora. A criação de outras partições requer operações de linguagem de definição de dados (DDL), como `CREATE TABLE`, que não são replicadas do ambiente azul para o ambiente verde. No entanto, as tabelas particionadas existentes e os respectivos dados serão replicados para o ambiente verde.
- As limitações a seguir se aplicam às extensões do PostgreSQL:
  - A extensão `pg_partman` deve ser desabilitada no ambiente azul quando uma implantação azul/verde é criada. A extensão realiza operações de DDL como `CREATE TABLE`, que interrompem a replicação lógica do ambiente azul no ambiente verde.

- A extensão `pg_cron` deve permanecer desativada em todos os bancos de dados verdes após a criação da implantação azul/verde. A extensão tem trabalhadores em segundo plano que são executados como superusuários e ignoram a configuração somente leitura do ambiente verde, o que pode causar conflitos de replicação.
- A extensão `apg_plan_mgmt` deve ter o parâmetro `apg_plan_mgmt.capture_plan_baselines` definido como `off` em todos os bancos de dados verdes para evitar conflitos de chave primária se um plano idêntico for capturado no ambiente azul. Para obter mais informações, consulte the section called “Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL”.
- As extensões `pglogical` e `pgactive` devem ser desativadas no ambiente azul ao criar uma implantação azul/verde. Depois de fazer a transição do ambiente verde para o novo ambiente de produção, será possível habilitar as extensões novamente. Além disso, o banco de dados azul não pode ser um assinante lógico de uma instância externa.
- Se você estiver usando a extensão `pgAudit`, ela deverá permanecer nas bibliotecas compartilhadas (`shared_preload_libraries`) nos grupos de parâmetros de banco de dados personalizados para as instâncias de banco de dados azul e verde. Para obter mais informações, consulte the section called “Configurar a extensão pgAudit”.

Limitações específicas de replicação lógica para implantações azuis/verdes

O PostgreSQL tem certas restrições relacionadas à replicação lógica, que se traduzem em limitações ao criar implantações azul/verdes para clusters de banco de dados Aurora PostgreSQL RDS para .

Para obter mais informações sobre a replicação lógica do PostgreSQL, consulte a documentação do PostgreSQL.

Limitação	Explicação
Declarações de linguagem de definição de dados (DDL), como `CREATE TABLE` e`CREATE SCHEMA`, não são replicadas do	Se o Aurora detectar uma alteração de DDL no ambiente azul, seus bancos de dados verdes entrarão em um estado de replicação degradada. Você deve excluir a implantação azul/verde e todos os bancos de dados verdes e, em seguida, recriá-la.

Limitação	Explicação
ambiente azul para o ambiente verde.	
As instruções de linguagem de controle de dados (DCL), como `GRANT` e `REVOKE`, não são replicadas do ambiente azul para o ambiente verde.	Se o Aurora detectar uma tentativa de execução de uma instrução de DCL no ambiente azul, você verá uma mensagem de aviso. Não há configuração ou API disponível para alterar esse comportamento, pois é uma limitação do processo de implantação azul/verde.
As operações `NEXTVAL` em objetos de sequência não são sincronizadas entre o ambiente azul e o ambiente verde.	Durante a transição, o Aurora incrementa os valores da sequência no ambiente verde para corresponder aos do ambiente azul. Se você tiver milhares de sequências, isso pode atrasar a transição.

Limitação	Explicação
Os objetos grandes no ambiente azul não são replicados no ambiente verde. Isso inclui objetos grandes existentes e quaisquer objetos grandes recém-criados ou modificados durante o processo de implantação azul/verde.	Se o Aurora detectar a criação ou modificação de objetos grandes no ambiente azul que estão armazenados na tabela do **`pg_largeobject`** sistema, seus bancos de dados verdes entrarão em um estado de replicação degradada. Você deve excluir a implantação azul/verde e todos os bancos de dados verdes e, em seguida, recriá-la.
Atualizar visões materializadas interrompe a replicação.	Atualizar visões materializadas no ambiente azul interrompe a replicação no ambiente verde. Evite atualizar visões materializadas no ambiente azul. Após uma transição, é possível atualizá-las manualmente usando o comando REFRESH MATERIALIZED VIEW ou programar uma atualização.
As operações UPDATE e DELETE não são permitidas em tabelas que não têm uma chave primária.	Antes de criar uma implantação azul/verde, verifique se todas as tabelas têm uma chave primária ou use `REPLICA IDENTITY FULL`. No entanto, use somente `REPLICA IDENTITY FULL` se não houver uma chave primária ou única, pois isso afeta a performance da replicação. Para obter mais informações, consulte a Documentação do PostgreSQL.

Limitações do Aurora Global Database para implantações azul/verde

Além das limitações gerais e específicas do mecanismo mencionadas acima, as seguintes limitações se aplicam às implantações azul/verde do Aurora Global Database:

- Todas as operações devem ser iniciadas na mesma região que o cluster de gravador do banco de dados global.
- Realizar uma transição global ou um failover global fará com que a implantação ativa azul/verde se torne inválida. A implantação azul-verde precisa ser excluída e recriada da nova região primária.
- Para o Aurora PostgreSQL, se você tiver o encaminhamento de gravação global habilitado em seu ambiente de produção e criar uma implantação azul/verde, o encaminhamento de gravação será desabilitado no cluster verde. Ele é habilitado no ambiente verde somente após a transição azul/verde, quando o ambiente verde se torna o novo ambiente de produção. Após a transição, o encaminhamento de gravação é desabilitado no cluster `-old1`.
- A modificação da topologia do banco de dados global após a criação da implantação azul/verde fará com que a implantação azul/verde ativa se torne inválida. Seria necessário excluir e recriar a implantação azul-verde da nova região primária.
- Os snapshots automatizados ficam retidos de acordo com os dias de retenção de backup que foram originalmente configurados no antigo ambiente azul. Os snapshots automatizados do antigo cluster azul não são copiados no verde.
- O failover global é aceito durante uma transição azul/verde, mas a transição global não é aceita durante uma transição azul/verde.
- Garanta que o cluster de banco de dados e os grupos de parâmetros de banco de dados para o ambiente verde existam em todas as regiões secundárias com nomes idênticos. Se o grupo de parâmetros em qualquer região não estiver disponível, o grupo de parâmetros padrão nas regiões será usado.
- Evite usar o RDS Proxy em qualquer membro do banco de dados global durante a transição de implantação azul/verde.

Considerações sobre implantações azul/verde

O Amazon RDS rastreia recursos em implantações azul/verde com o `DbiResourceId` e o `DbClusterResourceId` de cada recurso. Esse ID de recurso é um identificador imutável e exclusivo da Região da AWS do recurso.

O ID do recurso é diferente do ID do cluster de banco de dados. Cada um está listado na configuração do banco de dados no console do RDS.

O nome (ID do cluster) de um recurso muda quando você faz a transição de uma implantação azul/verde, mas cada recurso mantém o mesmo ID de recurso. Por exemplo, um identificador de cluster

de banco de dados pode ser `mycluster` no ambiente azul. Após a transição, o mesmo cluster de banco de dados pode ser renomeado para `mycluster-old1`. No entanto, o ID do recurso do cluster de banco de dados não muda durante a transição. Portanto, na transição dos recursos verdes para os novos recursos de produção, os respectivos IDs de recurso não correspondem aos IDs de recurso azuis que estavam anteriormente em produção.

Depois que você realizar a transição de uma implantação azul/verde, pense em atualizar os IDs de recurso para aqueles dos recursos de produção cuja transição foi feita recentemente para recursos e serviços integrados que você utilizou com os recursos de produção. Especificamente, considere as seguintes atualizações:

- Se você realizar a filtragem usando a API e os IDs de recursos do RDS, ajuste os IDs de recursos usados na filtragem após a transição.
- Se você usa o CloudTrail para recursos de auditoria, ajuste os consumidores do CloudTrail para rastrear os novos IDs de recursos após a transição. Para ter mais informações, consulte Monitorar chamadas de API do Amazon Aurorano AWS CloudTrail.
- Se você usar o Database Activity Streams para recursos no ambiente azul, ajuste sua aplicação para monitorar os eventos do banco de dados para o novo fluxo após a transição. Para ter mais informações, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com fluxos de atividades de banco de dados.
- Se você usar a API do Performance Insights, ajuste os IDs dos recursos nas chamadas para a API após a transição. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

  Você pode monitorar um banco de dados com o mesmo nome após a transição, mas ele não contém os dados de antes da transição.
- Se você usar IDs de recurso nas políticas do IAM, adicione os IDs dos recursos dos quais foi feita a transição recentemente quando necessário. Para obter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.
- Se você tiver perfis do IAM associados ao cluster de banco de dados, associe-os novamente depois da transição. Os perfis anexados não são copiados automaticamente no ambiente verde.
- Se você se autenticar no cluster de banco de dados usando a autenticação do banco de dados do IAM, garanta que a política do IAM usada para acesso ao banco de dados tenha os bancos de dados azul e verde listados sob o elemento `Resource` da política. Isso é necessário para se conectar ao banco de dados verde após a transição. Para obter mais informações, consulte the section called “Criar e usar uma política do IAM para acesso do banco de dados do IAM”.

- Se você quiser restaurar um snapshot de cluster de banco de dados manual ou automatizado para um cluster de banco de dados que fazia parte de uma implantação azul/verde, restaure o snapshot de cluster de banco de dados correto examinando a hora em que o snapshot foi obtido. Para ter mais informações, consulte Restauração de um snapshot de um cluster de banco de dados.
- Depois de mudar, as tarefas de replicação do AWS Database Migration Service (AWS DMS) não podem ser retomadas porque o ponto de verificação do ambiente azul é inválido no ambiente verde. Você deve recriar a tarefa do DMS com um novo ponto de verificação para continuar a replicação.
- O Amazon Aurora cria o ambiente verde clonando o volume de armazenamento subjacente do Aurora no ambiente azul. O volume do cluster verde armazena somente as alterações incrementais feitas no ambiente verde. Se você excluir o cluster de banco de dados no ambiente azul, o tamanho do volume de armazenamento subjacente do Aurora no ambiente verde será aumentado para o tamanho total. Para ter mais informações, consulte the section called “Clonar um volume para um cluster de banco de dados do Aurora”.
- Quando você adiciona uma instância de banco de dados ao cluster de banco de dados no ambiente verde de uma implantação azul/verde, a nova instância de banco de dados não substituirá uma instância de banco de dados no ambiente azul quando você fizer a transição. No entanto, a nova instância de banco de dados é mantida no cluster de banco de dados e torna-se uma instância de banco de dados no novo ambiente de produção.
- Quando você exclui uma instância de banco de dados no cluster de banco de dados no ambiente verde de uma implantação azul/verde, não é possível criar uma instância de banco de dados para substituí-la na implantação azul/verde.

  Se você criar uma instância de banco de dados com o mesmo nome e ARN da instância de banco de dados excluída, ela terá um `DbiResourceId` diferente, portanto, não fará parte do ambiente verde.

  Ocorrerá o comportamento a seguir se você excluir uma instância de banco de dados no cluster de banco de dados no ambiente verde:

  - Se existir uma instância de banco de dados no ambiente azul com o mesmo nome, não será feita a transição dela para a instância de banco de dados no ambiente verde. Essa instância de banco de dados não será renomeada anexando `-old`*n* ao nome da instância de banco de dados.
  - Qualquer aplicação que aponte para a instância de banco de dados no ambiente azul continua usando a mesma instância de banco de dados após a transição.

- Se você usa tags de recursos para controle de acesso ou gerenciamento operacional, precisa entender que as alterações das tags não são sincronizadas entre os ambientes azul e verde até a transição. Ao criar uma implantação azul/verde, as tags do ambiente azul são copiadas para o ambiente verde. Após a criação, nenhuma modificação de tag que você fizer em qualquer um dos ambientes será sincronizada automaticamente. Durante a transição, as tags de ambiente azul substituem todas as tags no ambiente verde. Aplique todas as tags necessárias ao ambiente azul antes de criar a implantação azul/verde ou reaplique as tags necessárias ao novo ambiente de produção após a transição. Para ter mais informações sobre tags, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.

Práticas recomendadas para implantações azul/verde do Amazon Aurora

Veja as práticas recomendadas para implantações azul/verde.

Tópicos

- Práticas recomendadas gerais para implantações azuis/verdes
- Práticas recomendadas do Aurora MySQL para implantações azuis/verdes
- Práticas recomendadas do Aurora PostgreSQL para implantações azuis/verdes
- Práticas recomendadas do Aurora Global Database para implantações azul/verde.

Práticas recomendadas gerais para implantações azuis/verdes

Considere as seguintes práticas recomendadas gerais ao criar uma implantação azul/verde.

- Teste minuciosamente o cluster de banco de dados do Aurora no ambiente verde antes da transição.
- Mantenha seus bancos de dados no ambiente verde somente leitura. Recomendamos que você habilite as operações de gravação no ambiente verde com cuidado, pois elas podem causar conflitos de replicação. Elas também podem ocasionar dados não intencionais nos bancos de dados de produção após a transição.
- Se você usar uma implantação azul/verde para implementar alterações de esquema, faça somente alterações compatíveis com a replicação.

  Por exemplo, é possível adicionar novas colunas ao final de uma tabela sem interromper a replicação da implantação azul para a implantação verde. No entanto, alterações de esquema, como renomear colunas ou renomear tabelas, transformam a replicação na implantação verde.

Para ter mais informações sobre alterações compatíveis com replicação, consulte Replicação com diferentes definições de tabela na origem e na réplica na documentação do MySQL e Restrições na documentação de replicação lógica do PostgreSQL.

- Use o endpoint do cluster, o endpoint do leitor ou o endpoint personalizado para todas as conexões nos dois ambientes. Não use endpoints de instância nem endpoints personalizados com listas estáticas ou de exclusão.
- Ao realizar a transição de uma implantação azul/verde, siga as práticas recomendadas de transição. Para obter mais informações, consulte the section called “Práticas recomendadas de transição”.

Práticas recomendadas do Aurora MySQL para implantações azuis/verdes

Considere as seguintes práticas recomendadas ao criar uma implantação azul/verde em um cluster de banco de dados do Aurora MySQL.

- Se o ambiente verde apresentar atraso na réplica, pense no seguinte:
  - Desabilite a retenção de logs binários, se não for necessária, ou desabilite-a temporariamente até que a replicação seja atualizada. Para fazer isso, redefina o parâmetro `binlog_format` de cluster de banco de dados como `0` e reinicialize a instância verde de banco de dados do gravador.
  - Defina o parâmetro `innodb_flush_log_at_trx_commit` temporariamente como `0` no grupo de parâmetros verde do banco de dados. Depois de atualizar a replicação, faça a reversão para o valor padrão de 1 antes da transição. Se ocorrer um desligamento inesperado ou uma falha com os valores dos parâmetros temporários, compile o ambiente verde novamente para evitar que nenhuma corrupção de dados passe despercebida. Para obter mais informações, consulte the section called “Configurar a frequência com que o buffer de log é liberado”.

Práticas recomendadas do Aurora PostgreSQL para implantações azuis/verdes

Considere as seguintes práticas recomendadas ao criar uma implantação azul/verde em um cluster de banco de dados do Aurora PostgreSQL.

- Monitore o cache de gravação de replicação lógica do Aurora PostgreSQL e realize ajustes no buffer de cache, se necessário. Para obter mais informações, consulte the section called “Monitorar o cache de gravação de replicação lógica”.

- Aumente o valor do parâmetro de banco de dados `logical_decoding_work_mem` no ambiente azul. Isso permite menos decodificação no disco e uso da memória. Para obter mais informações, consulte the section called “Ajustar a memória de trabalho para decodificação lógica”.
  - É possível monitorar o estouro de transações que está sendo gravado em disco usando a métrica `ReplicationSlotDiskUsage` do CloudWatch. Essa métrica oferece insights sobre o uso do espaço em disco por slots de replicação, ajudando a identificar quando os dados da transação excedem a capacidade de memória e são armazenados em disco. Você pode monitorar a memória livre com a métrica do `FreeableMemory` do CloudWatch. Para obter mais informações, consulte the section called “Métricas no nível da instância do Amazon Aurora”.
  - No Aurora PostgreSQL versão 14 e posterior, é possível monitorar o tamanho dos arquivos de estouro lógico usando a visualização `pg_stat_replication_slots` do sistema.
- Atualize todas as extensões do PostgreSQL para a versão mais recente antes de criar uma implantação azul/verde. Para obter mais informações, consulte the section called “Atualizar extensões do PostgreSQL”.
- Se você estiver usando a extensão `aws_s3`, conceda ao cluster de banco de dados verde acesso ao Amazon S3 por meio de um perfil do IAM após a criação do ambiente verde. Isso permite que os comandos de importação e exportação continuem funcionando após a transição. Para instruções, consulte the section called “Configurar o acesso a um bucket do Amazon S3”.
- Se você especificar uma versão posterior do mecanismo para o ambiente verde, execute a operação `ANALYZE` em todos os bancos de dados para atualizar a tabela `pg_statistic`. As estatísticas do otimizador não são transferidas durante uma atualização de versão principal, portanto, é necessário gerar novamente todas as estatísticas para evitar problemas de performance. Para conhecer práticas recomendadas adicionais durante as principais atualizações de versões, consulte the section called “Realizar uma atualização da versão principal”.
- Evite configurar gatilhos como `ENABLE REPLICA` ou `ENABLE ALWAYS` se o gatilho for usado na origem para manipular dados. Caso contrário, o sistema de replicação propagará as alterações e executará o gatilho, o que ocasiona duplicação.
- Transações de longa duração podem causar um atraso significativo na réplica. Para reduzir o atraso na réplica, pense no seguinte:
  - Reduza as transações de longa duração e as subtransações que podem ser adiadas até que o ambiente verde alcance o ambiente azul.
  - Reduza as operações em massa no ambiente azul até que o ambiente verde alcance o ambiente azul.

- Inicie uma operação manual de congelamento de vacuum em tabelas ocupadas antes de criar a implantação azul/verde. Para conferir instruções, consulte Realização de um congelamento manual de vacuum.
- No PostgreSQL versão 12 e posterior, desabilite o parâmetro `index_cleanup` em tabelas grandes ou muito utilizadas para melhorar a eficiência da manutenção normal em bancos de dados azuis. Para ter mais informações, consulte Aspirar uma tabela o mais rápido possível.

  > **Note**
  >
  > Ignorar regularmente a limpeza do índice durante a aspiração pode causar inchaço no índice, o que pode prejudicar o desempenho da limpeza. Como prática recomendada, utilize essa abordagem somente ao usar uma implantação azul/verde. Depois que a implantação estiver concluída, recomendamos retomar a manutenção e a limpeza regulares do índice.

- Poderá ocorrer atraso na réplica se as instâncias de banco de dados azul e verde estiverem subdimensionadas para a workload. Garanta que as instâncias de banco de dados não estejam atingindo os limites de recurso para o tipo de instância. Para obter mais informações, consulte the section called “Analisar o uso de recursos com métricas do CloudWatch”.

- A replicação lenta pode fazer com que remetentes e destinatários sejam reiniciados com frequência, o que atrasa a sincronização. Para garantir que eles permaneçam ativos, desabilite os tempos limite definindo o parâmetro `wal_sender_timeout` como `0` no ambiente azul e o parâmetro `wal_receiver_timeout` como `0` no ambiente verde.
- Analise o desempenho das instruções UPDATE e DELETE e avalie se a criação de um índice na coluna utilizada na cláusula WHERE pode otimizar essas consultas. Isso pode melhorar o desempenho quando as operações são repetidas no ambiente verde. Para obter mais informações, consulte the section called “Verificar filtros de predicados em busca de consultas que geram esperas”.
- Se você estiver usando gatilhos, garanta que eles não interfiram na criação, atualização e eliminação de objetos `pg_catalog.pg_publication`, `pg_catalog.pg_subscription` e `pg_catalog.pg_replication_slots` cujos nomes comecem com “rds”.
- Se o Babelfish estiver ativado no cluster de banco de dados de origem, os seguintes parâmetros deverão ter as mesmas configurações no grupo de parâmetros do cluster de banco de dados de destino para o ambiente verde e no grupo de parâmetros do cluster de banco de dados de origem:
  - rds.babelfish_status

- babelfishpg_tds.tds_default_numeric_precision
- babelfishpg_tds.tds_default_numeric_scale
- babelfishpg_tsql.default_locale
- babelfishpg_tsql.migration_mode
- babelfishpg_tsql.server_collation_name

Para mais informações sobre esses parâmetros, consulte the section called “Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish ”.

Práticas recomendadas do Aurora Global Database para implantações azul/verde.

Além das práticas recomendadas gerais e específicas do mecanismo listadas acima, pense nas práticas recomendadas a seguir para o Aurora Global Database..

- Monitore as seguintes métricas do CloudWatch para identificar períodos de baixa atividade em seu ambiente de produção:
  - `DatabaseConnections`
  - `ActiveTransactions`

  Programe a transição entre azul e verde durante a janela de manutenção planejada ou durante um período de baixa atividade.
- A duração da transição entre azul e verde varia de acordo com sua workload e o número de regiões secundárias. Quando você inicia uma transição entre azul e verde, o serviço espera que o atraso da réplica chegue a zero antes de continuar. Recomendamos conferir o atraso da réplica antes de iniciar a transição.
- Se você pretende usar um parâmetro de banco de dados ou um grupo de parâmetros de cluster de banco de dados diferente do padrão para seu ambiente verde, crie o grupo de parâmetros desejado com o mesmo nome em todas as regiões secundárias antes de iniciar a implantação azul/verde.

Criar uma implantação azul/verde no Amazon Aurora

O RDS copia a topologia e os recursos do ambiente azul para uma área de preparação. Se uma instância de banco de dados azul tiver réplicas de leitura, elas serão copiadas como réplicas da

instância verde. O armazenamento alocado de todas as réplicas verdes corresponde à instância primária verde, enquanto outros parâmetros de armazenamento são herdados das réplicas azuis.

Ao criar uma implantação azul/verde, você especifica o cluster de banco de dados a ser copiado na implantação. O cluster de banco de dados selecionado é o cluster de banco de dados de produção e torna-se o cluster de banco de dados no ambiente azul. O RDS copia a topologia do ambiente azul para uma área de teste, junto com seus recursos configurados. O cluster de banco de dados é copiado no ambiente verde, e o RDS configura a replicação do cluster de banco de dados no ambiente azul para o cluster de banco de dados no ambiente verde. O RDS também copia todas as instâncias de banco de dados no cluster de banco de dados.

Tópicos

- Preparação para uma implantação azul/verde
- Especificar as alterações ao criar uma implantação azul/verde
- Criar uma implantação azul/verde
- Configurações para criar implantações azuis/verdes

Preparação para uma implantação azul/verde

Há algumas etapas que você deve seguir antes de criar uma implantação azul/verde, dependendo do mecanismo que o cluster de banco de dados do Aurora está executando.

Tópicos

- Preparar um cluster de banco de dados do Aurora MySQL para uma implantação azul/verde
- Preparar um cluster de banco de dados do Aurora PostgreSQL para uma implantação azul/verde
- Preparar um cluster de banco de dados do Aurora Global Databases para uma implantação azul/verde

Preparar um cluster de banco de dados do Aurora MySQL para uma implantação azul/verde

Antes de criar uma implantação azul/verde para um cluster de banco de dados do Aurora MySQL, o cluster de banco de dados deve estar associado a um grupo de parâmetros de cluster de banco de dados personalizado com o registro em log binário (`binlog_format`) ativado. O registro em log binário é necessário para a replicação do ambiente azul para o ambiente verde. Embora qualquer

formato de log binário funcione, recomendamos `ROW` para reduzir o risco de inconsistências de replicação. Para obter informações sobre como criar um grupo de parâmetros de cluster de banco de dados personalizado e definir parâmetros, consulte the section called “Grupos de parâmetros de cluster de banco de dados”.

> **Note**
>
> Habilitar o registro em log binário aumenta o número de operações de E/S de disco de gravação no cluster de banco de dados. Você pode monitorar o uso de IOPS com a métrica `VolumeWriteIOPs` do CloudWatch.

Depois de habilitar o registro em log binário, reinicialize o cluster de banco de dados para que as alterações tenham efeito. As implantações azul/verde exigem que a instância do gravador esteja sincronizada com o grupo de parâmetros do cluster de banco de dados, caso contrário a criação falhará. Para obter mais informações, consulte Reinicializar uma instância de banco de dados em um cluster do Aurora.

Além disso, recomendamos alterar o período de retenção de logs binários para um valor diferente de `NULL` a fim de evitar que os arquivos de log binários sejam eliminados. Para obter mais informações, consulte the section called “Definir e mostrar a configuração de logs binários”.

Preparar um cluster de banco de dados do Aurora PostgreSQL para uma implantação azul/verde

Antes de criar uma implantação azul/verde para um cluster de banco de dados do Aurora PostgreSQL, faça o seguinte.

- Associe o cluster a um grupo de parâmetros de cluster de banco de dados personalizado com a replicação lógica (`rds.logical_replication`) ativada. A replicação lógica é necessária para a replicação do ambiente azul no ambiente verde.

  Ao habilitar a replicação lógica, é necessário também ajustar determinados parâmetros do cluster, como `max_replication_slots`, `max_logical_replication_workers` e `max_worker_processes`. Para ter instruções sobre como habilitar a replicação lógica e ajustar esses parâmetros, consulte the section called “Configurar a replicação lógica”.

  Além disso, verifique se o parâmetro `synchronous_commit` está definido como on.

Depois de configurar os parâmetros necessários, reinicialize o cluster de banco de dados para que as alterações tenham efeito. As implantações azul/verde exigem que a instância do gravador esteja sincronizada com o grupo de parâmetros do cluster de banco de dados, caso contrário a criação falhará. Para obter mais informações, consulte Reinicializar uma instância de banco de dados em um cluster do Aurora.

- Confirme se o cluster de banco de dados está executando uma versão do Aurora PostgreSQL compatível com implantações azuis/verdes. Para conferir uma lista de versões compatíveis, consulte the section called “Implantações azul/verde com o Aurora PostgreSQL”.
- Certifique-se de que todas as tabelas no cluster de banco de dados tenham uma chave primária. A replicação lógica do PostgreSQL não permite operações UPDATE ou DELETE em tabelas que não têm uma chave primária.

Preparar um cluster de banco de dados do Aurora Global Databases para uma implantação azul/verde

Antes de criar uma implantação azul/verde para seu cluster de banco de dados do Aurora Global Database, observe os seguintes pontos:

- Todas as operações devem ser iniciadas na mesma região que o cluster de gravador do banco de dados global.
- Configuração do grupo de parâmetros:
  - O ambiente verde usa um novo grupo de parâmetros que você especifica ou o mesmo grupo de parâmetros que do cluster azul (padrão).
  - Os grupos de parâmetros personalizados são copiados no ambiente verde.
  - Se um grupo de parâmetros especificado não existir na região secundária, o grupo de parâmetros padrão na região secundária será usado para o ambiente verde.

Especificar as alterações ao criar uma implantação azul/verde

Você pode fazer as seguintes alterações no cluster de banco de dados no ambiente verde ao criar a implantação azul/verde:

Você pode fazer outras modificações de banco de dados no cluster e em suas instâncias de banco de dados no ambiente verde após sua implantação. Por exemplo, você pode especificar uma versão posterior do mecanismo ou um grupo de parâmetros diferente.

Para ter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Tópicos

- Especifique uma versão de mecanismo superior
- Especificar outro grupo de parâmetros de banco de dados

Especifique uma versão de mecanismo superior

Você poderá especificar uma versão superior do mecanismo se quiser testar uma atualização do mecanismo de banco de dados. Após a transição, o banco de dados é atualizado para a versão principal ou secundária do mecanismo de banco de dados que você especificar.

Especificar outro grupo de parâmetros de banco de dados

Você pode especificar um grupo de parâmetros de cluster de banco de dados diferente do usado pelo cluster de banco de dados. É possível testar como as alterações de parâmetros afetam o cluster de banco de dados no ambiente verde ou especificar um grupo de parâmetros para uma nova versão principal do mecanismo de banco de dados no caso de uma atualização.

Se você especificar um grupo de parâmetros de cluster de banco de dados diferente, o grupo de parâmetros especificado será associado ao cluster de banco de dados no ambiente verde. Se você não especificar um grupo de parâmetros de cluster de banco de dados diferente, o cluster de banco de dados no ambiente verde será associado ao mesmo grupo de parâmetros que o cluster de banco de dados azul.

Criar uma implantação azul/verde

Você pode criar a implantação azul/verde usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

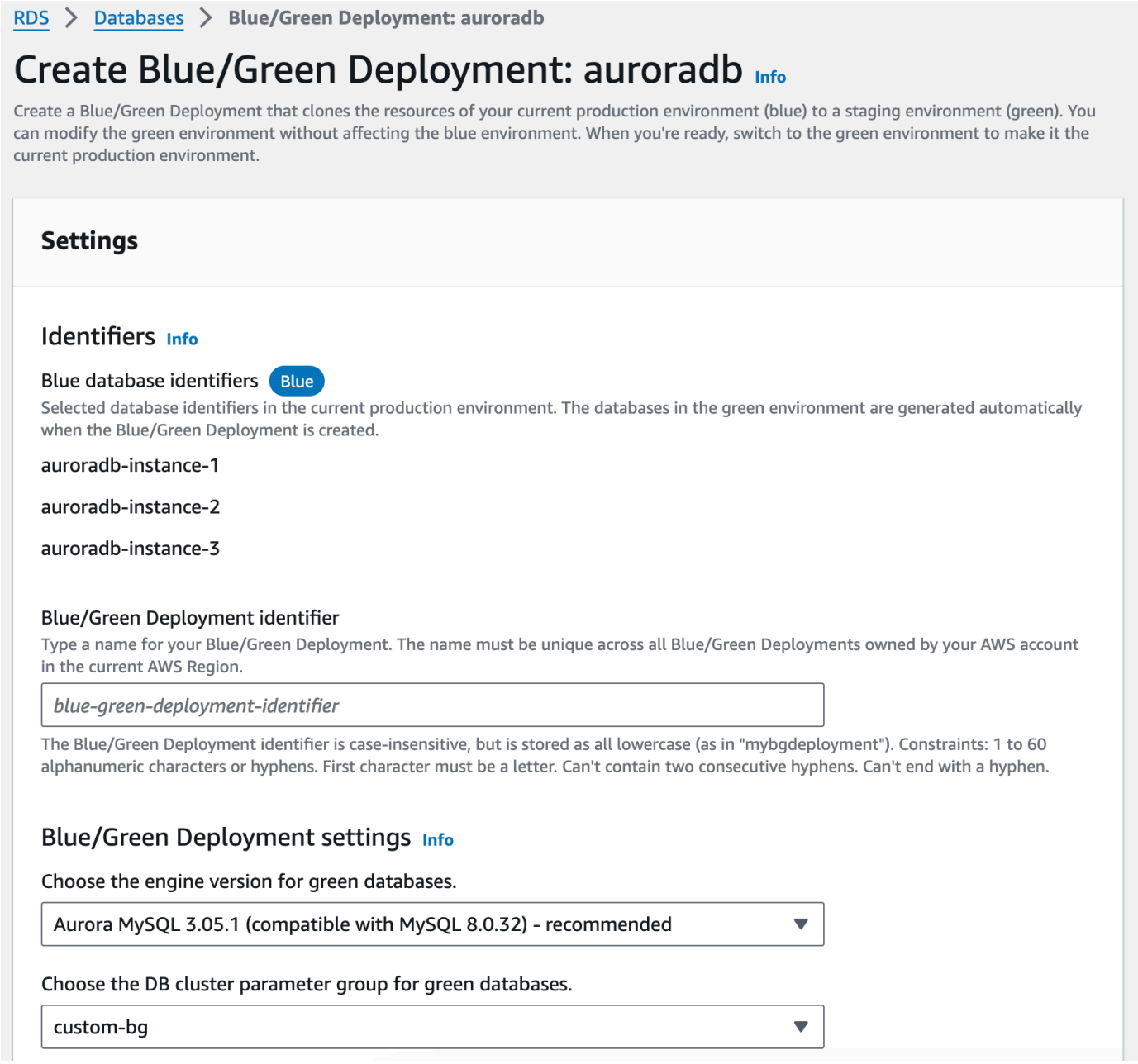
Console

Como criar uma implantação azul/verde

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e, depois, o cluster de banco de dados que você deseja copiar em um ambiente verde.

3. Selecione Ações, Criar implantação azul/verde.

   A página Criar implantação azul/verde é exibida.



4. Analise os identificadores azuis do banco de dados. Eles devem corresponder às instâncias de banco de dados que você espera no ambiente azul. Caso contrário, selecione Cancel (Cancelar).
5. Em Nome da implantação azul/verde, insira um nome para sua implantação azul/verde.
6. Nas seções restantes, especifique as configurações do ambiente verde. Para obter informações sobre cada configuração, consulte the section called “Configurações disponíveis”.

Você pode fazer outras modificações nos bancos de dados no ambiente verde após sua implantação.

7. Escolha Criar.

AWS CLI

Para criar uma implantação azul/verde usando a AWS CLI, utilize o comando create-blue-green-deployment. Para obter informações sobre todas as opções disponíveis, consulte the section called "Configurações disponíveis".

Example

Para Linux, macOS ou Unix:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name aurora-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb \
    --target-engine-version 8.0 \
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

Para Windows:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name aurora-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb ^
    --target-engine-version 8.0 ^
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

API do RDS

Para criar uma implantação azul/verde usando a API do Amazon RDS, use a operação `CreateBlueGreenDeployment`. Para ter mais informações sobre cada opção, consulte the section called "Configurações disponíveis".

Configurações para criar implantações azuis/verdes

A tabela a seguir explica as configurações que você pode escolher ao criar uma implantação azul/verde. Consulte mais informações sobre as opções da AWS CLI em create-blue-green-deployment. Consulte mais informações sobre os parâmetros da API do RDS em CreateBlueGreenDeployment.

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS

Identificador de implantação azul/verde	Um nome para a implantação azul/verde.	Opção da CLI:
		`--blue-green-deployment-name`
		Parâmetro da API:
		`BlueGreenDeploymentName`

Identificador de banco de dados azul	O identificador do cluster que você deseja copiar no ambiente verde. Ao usar a CLI ou a API, especifique o nome do recurso da Amazon (ARN) do cluster.	Opção da CLI:
		`--source`
		Parâmetro da API:
		`Source`

Grupo de parâmetros do cluster de banco de dados para bancos de dados verdes	Um grupo de parâmetros para associar aos bancos de dados no ambiente verde.	Opção da CLI:
		`--target-db-cluster-parameter-group-name`
		Parâmetro da API:
		`TargetDBClusterParameterGroupName`

Versão do mecanismo para bancos de dados verdes	Faça upgrade do cluster no ambiente verde para a versão especificada do mecanismo de banco de dados.
	Se você escolher um cluster de banco de dados Aurora PostgreSQL, analise e reconheça as limitações da replicação lógica. Para obter
		Opção da CLI:
		`--target-engine-version`
		Parâmetro da API do RDS:
		`TargetEngineVersion`

Configuração do console	Descrição da configuração	Opção da CLI e parâmetro da API do RDS
	mais informações, consulte the section called “Limitações específicas de replicação lógica para implantações azuis/verdes”.	

Visualizar uma implantação azul/verde no Amazon Aurora

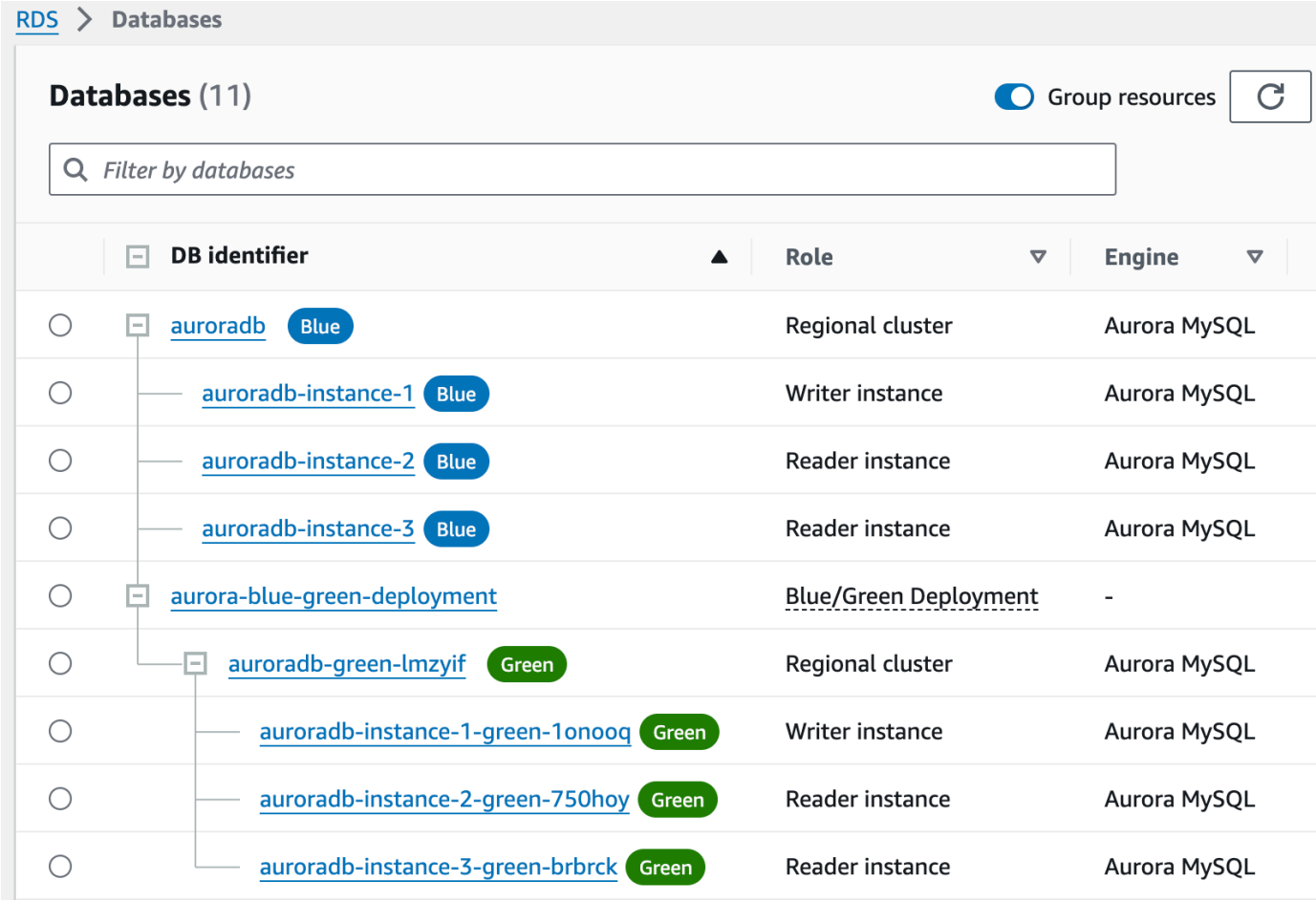
Você pode visualizar os detalhes de uma implantação azul/verde usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Você também pode visualizar e assinar eventos para obter informações sobre uma implantação azul/verde. Para ter mais informações, consulte Eventos de implantação azul/verde.

Console

Como visualizar os detalhes sobre uma implantação azul/verde

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e, depois, localize a implantação azul/verde na lista.

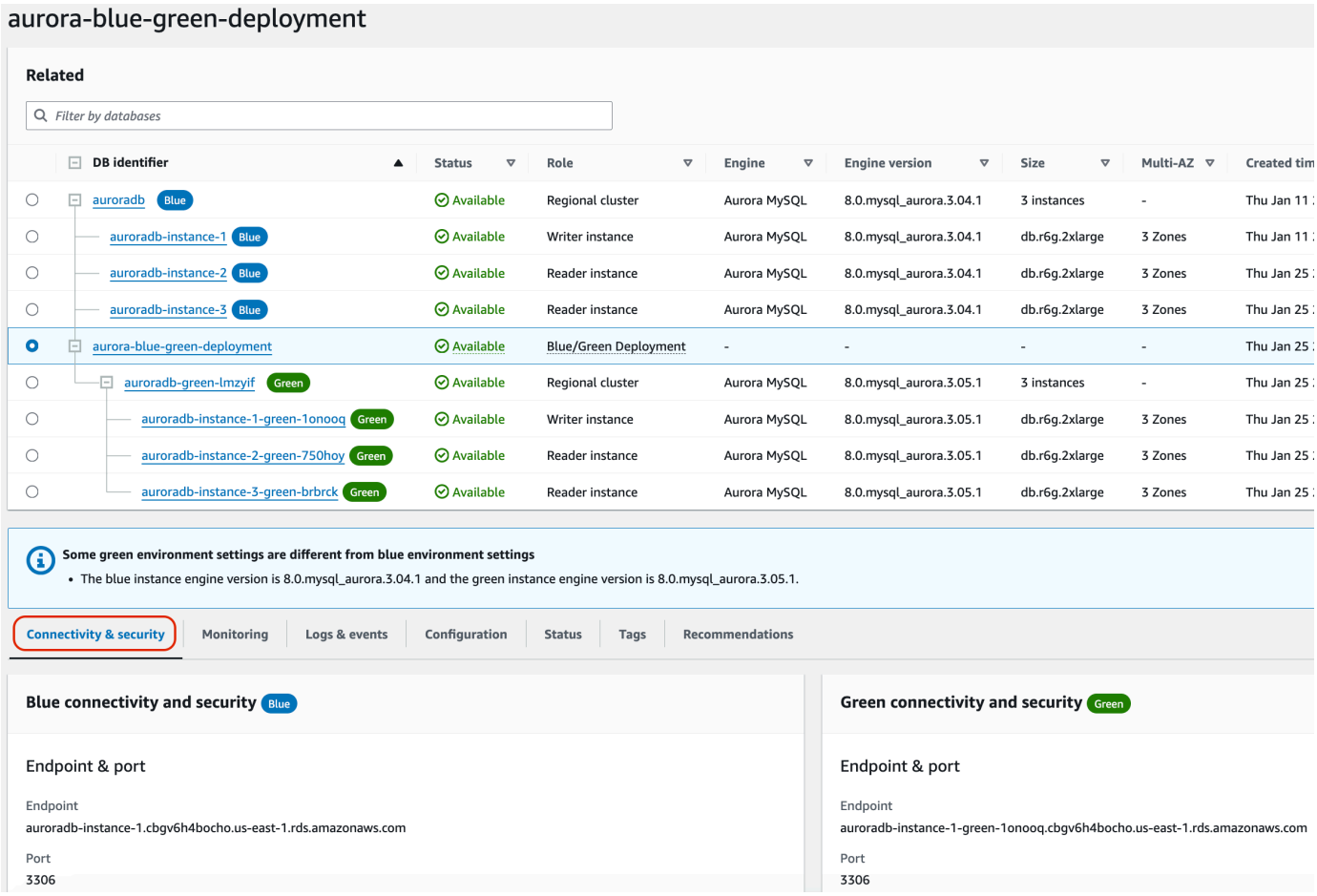


O valor de Role (Função) para a implantação azul/verde é Blue/Green Deployment (Implantação azul/verde).

3. Selecione o nome da implantação azul/verde que você deseja visualizar para exibir seus detalhes.

   Cada guia tem uma seção para a implantação azul e uma seção para a implantação verde. Por exemplo, na guia Configuração, a versão do mecanismo de banco de dados pode ser diferente no ambiente azul e no ambiente verde se você estiver atualizando a versão do mecanismo de banco de dados no ambiente verde.

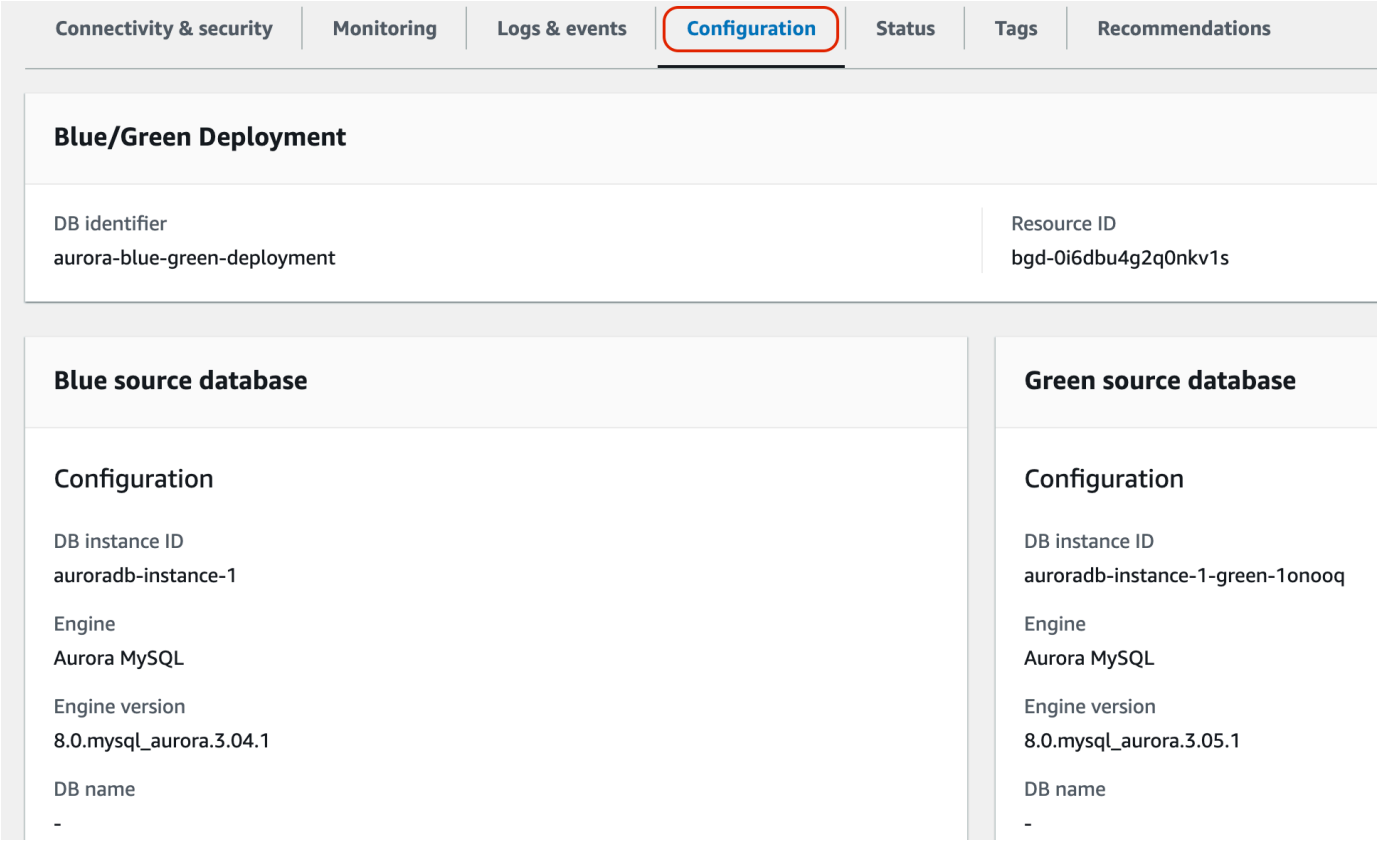
   A imagem a seguir mostra um exemplo da guia Conectividade e segurança.



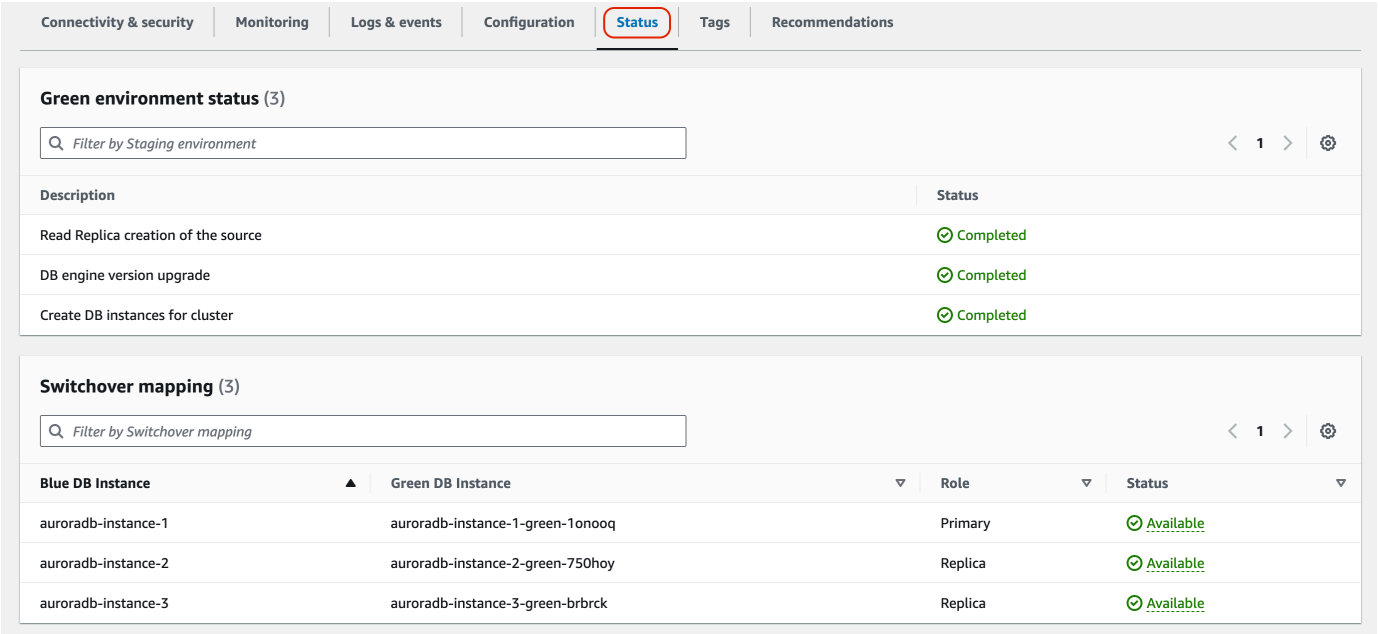
A guia Conectividade e segurança também inclui uma seção chamada Replicação, que mostra o estado atual da replicação e o atraso da réplica entre os ambientes azul e verde. Se o estado de replicação for `Replicating`, a implantação azul/verde está sendo replicada com êxito.

Para implantações azuis/verdes do Aurora PostgreSQL que usam replicação lógica, o estado da replicação pode mudar para `Replication degraded` se você fizer alterações de DDL incompatíveis ou de objetos grandes no ambiente azul. Para ter mais informações, consulte the section called “Limitações específicas de replicação lógica para implantações azuis/verdes”.

A imagem a seguir mostra um exemplo da guia Configuração.



A seguinte imagem mostra um exemplo da guia Status:



AWS CLI

Para ver os detalhes sobre uma implantação azul/verde usando a AWS CLI, use o comando describe-blue-green-deployments.

Example Veja os detalhes sobre uma implantação azul/verde filtrando por seu nome

Ao usar o comando describe-blue-green-deployments, você pode filtrar no `--blue-green-deployment-name`.

O exemplo a seguir mostra os detalhes de uma implantação azul/verde chamada *`my-blue-green-deployment`*.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example Visualizar os detalhes sobre uma implantação azul/verde especificando seu identificador

Ao usar o comando describe-blue-green-deployments, é possível especificar a opção `--blue-green-deployment-identifier`.

O exemplo a seguir mostra os detalhes de uma implantação azul/verde com o identificador *`bgd-1234567890abcdef`*.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

API do RDS

Para ver os detalhes sobre uma implantação azul/verde usando a API do Amazon RDS, use a operação `DescribeBlueGreenDeployments` e especifique o `BlueGreenDeploymentIdentifier`.

Mudar uma implantação azul/verde no Amazon Aurora

Uma transição transforma o cluster de banco de dados, inclusive as instâncias de banco de dados, do ambiente verde no cluster de banco de dados de produção. Antes de fazer a transição, o tráfego de produção é roteado para o cluster no ambiente azul. Depois de fazer a transição, o tráfego de produção é roteado para o cluster de banco de dados no ambiente verde.

Mudar para uma implantação azul/verde não é o mesmo que promover o cluster de banco de dados verde na implantação azul/verde. Se você promover manualmente o cluster de banco de dados escolhendo Promover no menu Ações, a replicação entre os ambientes azul e verde será interrompida e a implantação azul/verde entrará no estado de Configuração inválida.

Tópicos

- Tempo limite de transição
- Barreiras de proteção de transição
- Ações de transição
- Práticas recomendadas de transição
- Verificar as métricas do CloudWatch antes da transição
- Monitorar o atraso da réplica antes da transição
- Realizar a transição de uma implantação azul/verde
- Após a transição

Tempo limite de transição

Você pode especificar um tempo limite de transição entre 30 segundos e 3.600 segundos (uma hora). Se a transição demorar mais do que o especificado, todas as alterações serão revertidas e nenhuma alteração será feita em nenhum dos ambientes. O limite de tempo padrão é 300 segundos (cinco minutos).

Barreiras de proteção de transição

Quando você inicia uma transição, o Amazon RDS executa algumas verificações básicas para testar a prontidão dos ambientes azul e verde para a transição. Essas verificações são conhecidas como barreiras de proteção de transição. Essas barreiras evitarão uma transição se os ambientes não estiverem prontos para isso. Portanto, elas evitam tempo de inatividade mais longo do que o esperado e evitam a perda de dados entre os ambientes azul e verde que pode ocorrer se a transição for iniciada.

O Amazon RDS executa as seguintes verificações de barreira de proteção no ambiente verde:

- Integridade da replicação: confira se o status de replicação do cluster de banco de dados verde é íntegro. O cluster de banco de dados verde é uma réplica do cluster de banco de dados azul.

- Atraso na replicação: confira se o atraso da réplica do cluster de banco de dados está nos limites permitidos para a transição. Os limites permitidos são baseados no tempo limite especificado. O atraso da réplica indica até que ponto o cluster de banco de dados verde está atrás de seu cluster de banco de dados azul. Para obter mais informações, consulte the section called “Monitorar o atraso da réplica antes da transição”.
- Gravações ativas: certifique-se de que não haja gravações ativas no cluster de banco de dados.

O Amazon RDS executa as seguintes verificações de barreira de proteção no ambiente azul:

- Replicação externa: para o Aurora PostgreSQL, garante que o ambiente azul não seja uma fonte lógica autogerenciada (publicador) nem uma réplica (assinante). Se ele for, recomendamos que você elimine os slots de replicação autogerenciados e as assinaturas em todos os bancos de dados no ambiente azul, continue com a transição e, depois, recrie-os para retomar a replicação. Para o Aurora MySQL, verifica se o banco de dados azul não é uma réplica externa de log binário. Se for, verifique se ele não está se replicando ativamente.
- Gravações ativas de longa duração: verifica se não há gravações ativas de longa duração no cluster de banco de dados azul, pois elas podem aumentar o atraso da réplica.
- Instruções DDL de longa duração: verifica se não há instruções DDL de longa duração no cluster de banco de dados azul, pois elas podem aumentar o atraso da réplica.
- Alterações não compatíveis do PostgreSQL: para implantações azuis/verdes do Aurora PostgreSQL, verifica se não há nenhuma alteração de DDL e se nenhuma adição ou modificação de objetos grandes foi realizada no ambiente azul. Para obter mais informações, consulte the section called “Limitações específicas de replicação lógica para implantações azuis/verdes”.

  Se o Amazon RDS detectar alterações não compatíveis do PostgreSQL, ele alterará o estado de replicação para `Replication degraded` e notificará você de que a transição não está disponível para a implantação azul/verde. Para continuar com a transição, recomendamos que você exclua e recrie a implantação azul/verde e todos os bancos de dados verdes. Para fazer isso, escolha Ações, Excluir com bancos de dados verdes.

Ações de transição

Quando você alterna uma implantação azul/verde, o RDS realiza as seguintes ações:

1. Executa verificações de barreira de proteção para verificar se os ambientes azul e verde estão prontos para a transição.

2. Interrompe novas operações de gravação na no cluster de banco de dados nos dois ambientes.
3. Descarta conexões com as instâncias de banco de dados em ambos os ambientes e não permite novas conexões.
4. Espera que a replicação alcance o ambiente verde para que este esteja em sincronia com o ambiente azul.
5. Renomeia de banco de dados o cluster e instâncias de banco de dados nos dois ambientes.

   O RDS renomeia o cluster e as instâncias de banco de dados no ambiente verde para coincidir com o cluster e as instâncias de banco de dados no ambiente azul. Por exemplo, suponha que o nome de uma instância de banco de dados no ambiente azul seja `mydb`. Suponha também que o nome da instância de banco de dados correspondente no ambiente verde seja `mydb-green-abc123`. Durante a transição, o nome da instância de banco de dados no ambiente verde é alterado para `mydb`.

   O RDS renomeia o cluster e as instâncias de banco de dados no ambiente azul anexando `-old`*n* ao nome atual, em que *n* é um número. Por exemplo, suponha que o nome de uma instância de banco de dados no ambiente azul seja `mydb`. Após a transição, o nome da instância de banco de dados pode ser `mydb-old1`.

   O RDS também renomeia os endpoints no ambiente verde para corresponder aos endpoints correspondentes no ambiente azul, para que as alterações na aplicação não sejam necessárias.
6. Permite conexões com bancos de dados nos dois ambientes.
7. Permite operações de gravação no cluster de banco de dados no novo ambiente de produção.

   Após a transição, o cluster de banco de dados de produção anterior só permite operações de leitura. Mesmo se você ativar gravações no cluster de banco de dados, ele permanecerá somente leitura até que você exclua a implantação azul/verde.

Você pode monitorar o status de uma transição usando o Amazon EventBridge. Para obter mais informações, consulte the section called “Eventos de implantação azul/verde”.

Durante a transição, as tags do ambiente azul substituem todas as tags dos recursos no ambiente verde. Todas as tags adicionadas diretamente aos recursos do ambiente verde são substituídas durante esse processo. Para ter mais informações sobre tags, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.

Se a transição começar e parar antes de terminar por qualquer motivo, todas as alterações serão revertidas e nenhuma alteração será feita em nenhum dos ambientes.

Práticas recomendadas de transição

Antes de fazer a transição, é altamente recomendável que você siga as práticas recomendadas concluindo as seguintes tarefas:

- Teste minuciosamente os recursos no ambiente verde. Eles devem funcionar de forma adequada e eficiente.
- Monitore as métricas relevantes do Amazon CloudWatch. Para obter mais informações, consulte the section called “Verificar as métricas do CloudWatch antes da transição”.
- Identifique o melhor momento para a transição.

  Durante a transição, as gravações são cortadas dos bancos de dados nos dois ambientes. Identifique um momento em que o tráfego é o menor em seu ambiente de produção. Transações de longa duração, como DDLs ativas, podem aumentar seu tempo de transição, ocasionando maior tempo de inatividade para suas workloads de produção.

  Se houver um grande número de conexões em seu cluster de banco de dados e instâncias de banco de dados, considere reduzi-las manualmente até a quantidade mínima necessária para sua aplicação antes de realizar a transição da implantação azul/verde. Uma maneira de fazer isso é criar um script que monitore o status da implantação azul/verde e comece a limpar as conexões quando detectar que o status mudou para `SWITCHOVER_IN_PROGRESS`.
- O cluster e as instâncias de banco de dados nos dois ambientes devem estar no estado `Available`.
- O cluster de banco de dados no ambiente verde devem estar funcionando e sendo replicado.
- Garanta que suas configurações de rede e cliente não aumentem o tempo de vida útil (TTL) do cache DNS além de cinco segundos, que é o padrão para zonas DNS do Aurora . Do contrário, as aplicações continuarão a enviar tráfego de gravação ao ambiente azul após a transição.
- Para implantações azuis/verdes do Aurora PostgreSQL, faça o seguinte:
  - Analise as limitações de replicação lógica e realize todas as ações necessárias antes da transição. Para obter mais informações, consulte the section called “Limitações específicas de replicação lógica para implantações azuis/verdes”.

- Execute a operação `ANALYZE` para atualizar a tabela `pg_statistics`. Isso reduz o risco de problemas de desempenho após a transição.
- Antes de iniciar uma transição de implantação azul/verde, verifique se a aplicação não substitui o parâmetro `default_transaction_read_only` em nível de sessão. Durante a transição, esse parâmetro é definido como on no gravador do ambiente verde para evitar gravações enquanto a promoção não for concluída. Se a aplicação ou as transações substituírem essa configuração por `off`, a aplicação poderá gravar dados no ambiente verde durante o processo de transição. Caso seja necessário reverter a transição, essas gravações não estarão disponíveis no ambiente azul, o que exige que as inconsistências nos dados sejam resolvidas manualmente. Antes de prosseguir com a transição, é altamente recomendável auditar as consultas à aplicação para garantir que elas respeitem a configuração `default_transaction_read_only`.

> **Note**
>
> Durante uma transição, você não pode modificar nenhum cluster de banco de dados incluído na transição.

Verificar as métricas do CloudWatch antes da transição

Antes de realizar a transição de uma implantação azul/verde, recomendamos que verifique os valores das métricas a seguir no Amazon CloudWatch.

- `DatabaseConnections`: use esta métrica para estimar o nível de atividade na implantação azul/verde; certifique-se de que o valor esteja em um nível aceitável para sua implantação antes da transição. Se o recurso Insights de Performance estiver ativado, `DBLoad` será uma métrica mais precisa.
- `ActiveTransactions`: se `innodb_monitor_enable` estiver definido como `all` no grupo de parâmetros de banco de dados para qualquer uma de suas instâncias de banco de dados, use esta métrica para ver se há um grande número de transações ativas que podem bloquear a transição.

Para obter mais informações, consulte the section called “Métricas do CloudWatch para Aurora”.

Monitorar o atraso da réplica antes da transição

Antes de realizar a transição de uma implantação azul/verde, verifique se o atraso de réplica é próximo de zero para reduzir o tempo de inatividade.

Aurora MySQL

Para implantações azul/verde do MySQL, confira a métrica `AuroraBinlogReplicaLag` do CloudWatch no ambiente verde para identificar o atraso atual da réplica. Para obter mais informações, consulte the section called “Diagnosticar e resolver atrasos entre réplicas de leitura”.

Aurora PostgreSQL

Para implantações azuis/verdes do PostgreSQL, verifique a métrica `OldestReplicationSlotLag` do CloudWatch no ambiente azul para identificar o atraso atual da réplica. Para obter mais informações, consulte the section called “Métricas no nível da instância do Amazon Aurora”.

Além disso, você pode executar a consulta SQL a seguir no ambiente azul:

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

O `confirmed_flush_lsn` representa o número de sequência de log (LSN) mais recente enviado à réplica. O `pg_current_wal_lsn` representa onde o banco de dados está agora. Um `lsn_distance` de 0 significa que a réplica foi capturada.

Realizar a transição de uma implantação azul/verde

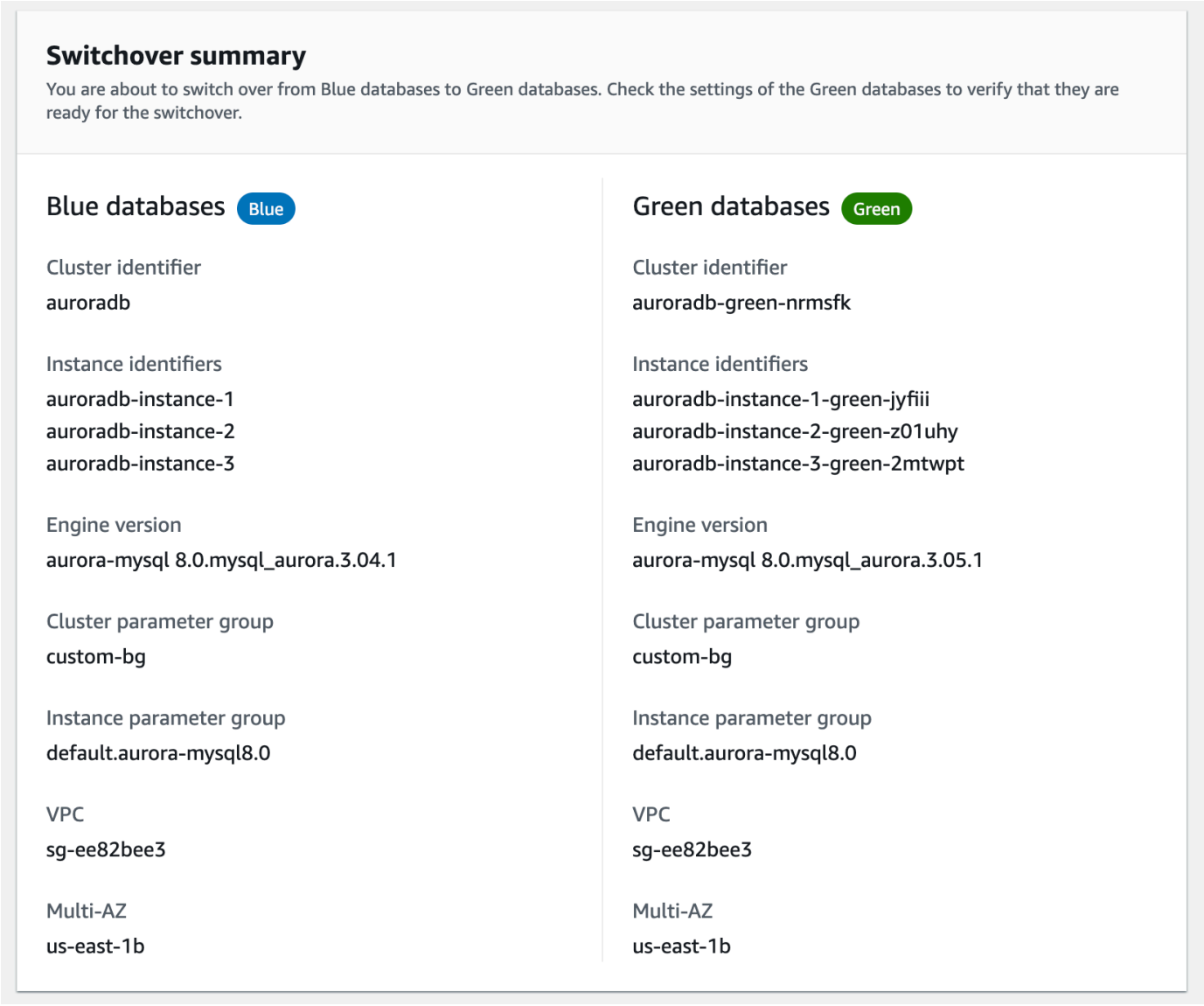
Você pode fazer a transição de uma implantação azul/verde usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como realizar a transição de uma implantação azul/verde

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e, depois, selecione a implantação azul/verde da qual você deseja realizar a transição.
3. Para Actions (Ações), selecione Switch over (Realizar transição).

   A página Switch over (Realizar transição) é exibida.



4. Na página Switch over (Realizar transição), revise o resumo da transição. Os recursos nos dois ambientes devem corresponder ao que você espera. Caso contrário, selecione Cancel (Cancelar).
5. Em Timeout (Tempo limite), insira o limite de tempo para a transição.
6. Se seu cluster estiver executando o Aurora PostgreSQL, a instância , analise e confirme as recomendações de pré-transição. Para obter mais informações, consulte the section called "Limitações específicas de replicação lógica para implantações azuis/verdes".
7. Selecione Switch Role (Realizar transição).

AWS CLI

Para realizar a transição de uma implantação azul/verde usando a AWS CLI, utilize o comando switchover-blue-green-deployment com as seguintes opções:

- `--blue-green-deployment-identifier`: especifique o ID do recurso da implantação azul/verde.
- `--switchover-timeout`: especifique o limite de tempo para a transição, em segundos. O padrão é 300.

Example Fazer a transição de uma implantação azul/verde

Para Linux, macOS ou Unix:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Para Windows:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

API do RDS

Para realizar a transição de uma implantação azul/verde usando a API do Amazon RDS, use a operação `SwitchoverBlueGreenDeployment` com os seguintes parâmetros:

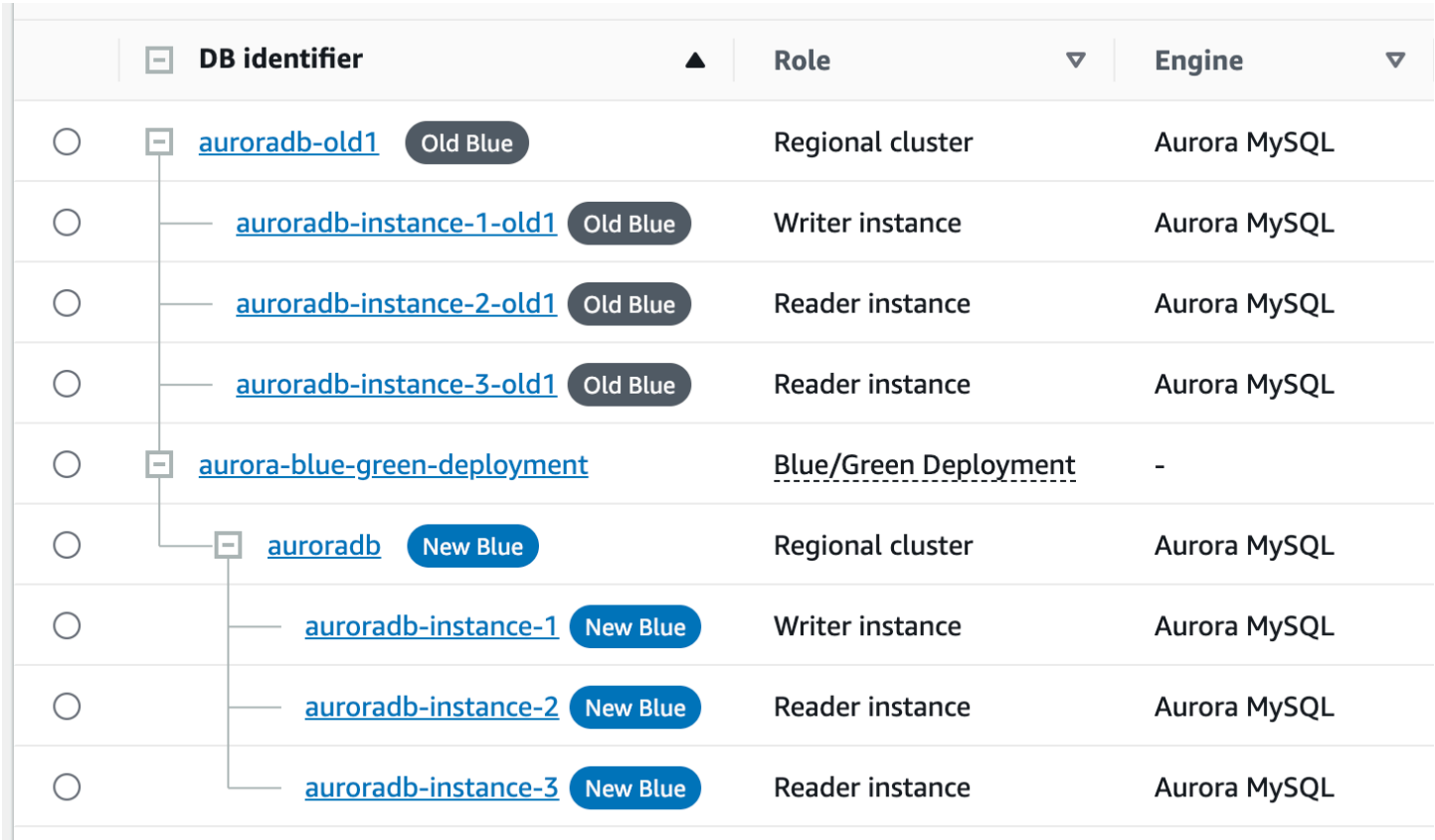
- `BlueGreenDeploymentIdentifier`: especifique o ID do recurso da implantação azul/verde.
- `SwitchoverTimeout`: especifique o limite de tempo para a transição, em segundos. O padrão é 300.

Após a transição

Depois de uma transição, o cluster e as instâncias de banco de dados no ambiente azul anterior são retidos. Os custos padrão se aplicam a esses recursos. A replicação entre os ambientes azul e verde é interrompida, bem como o registro em log binário.

O RDS renomeia o cluster e as instâncias de banco de dados no ambiente azul anexando `-old`*n* ao nome de recurso atual, em que *n* é um número. O cluster de banco de dados azul é forçado a entrar em um estado somente leitura. Mesmo se você habilitar as operações de gravação, ele permanecerá somente leitura até que você exclua a implantação azul/verde. O RDS nomeia o cluster e as instâncias de banco de dados no ambiente verde `-new`*n*.

Se você excluir o recurso de implantação azul/verde, o RDS reterá os recursos `-old`*n* e `-new`*n*.



Atualizar o nó principal para consumidores

O RDS oferece réplicas de leitura totalmente gerenciadas. No entanto, ele também oferece a opção de configurar réplicas autogerenciadas, também conhecidas como réplicas externas. As réplicas externas possibilitam o uso de recursos de terceiros como destinos de replicação.

Depois de fazer a transição de uma implantação azul/verde do Aurora MySQL, se o cluster de banco de dados tiver alguma réplica externa ou consumidores de log binário antes da transição, será necessário atualizar o nó principal após a transição a fim de manter a continuidade da replicação.

Como atualizar o nó pai

1. Após a transição, a instância de banco de dados de gravador que estava anteriormente no ambiente verde emite um evento que contém o nome do arquivo de log principal e a posição do log principal. Para localizar o evento, acesse o console do RDS e escolha Eventos no painel de navegação esquerdo.
2. Filtre por eventos em que a origem é o nome da antiga instância de banco de dados verde do gravador, antes da transição.
3. Localize o evento que contém as coordenadas de logs binários. A mensagem do evento é semelhante a: `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.000003 and position 40134574`.
4. Garanta que o consumidor ou a réplica tenha aplicado todos os logs binários do antigo ambiente azul. Depois, use as coordenadas de logs binários fornecidas para retomar a replicação nos consumidores. Por exemplo, se você estiver executando uma réplica do MySQL no EC2, poderá usar o seguinte comando:

   MySQL 8.0.22 e versões principais e secundárias anteriores

   ```
   CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
   ```

   MySQL 8.0.23 e versões principais e secundárias posteriores

   ```
   CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
    SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
   ```

Excluir uma implantação azul/verde no Amazon Aurora

Você pode excluir uma implantação azul/verde antes ou depois de realizar a transição.

Quando você exclui uma implantação azul/verde antes de realizar a transição, o Amazon RDS exclui opcionalmente o cluster de banco de dados no ambiente verde:

- Se você optar por excluir o cluster de banco de dados no ambiente verde (`--delete-target`), ele deverá ter proteção contra exclusão desativada.
- Se você não excluir o cluster de banco de dados no ambiente verde (`--no-delete-target`), ele será retido, mas ele não fará mais parte de uma implantação azul/verde. Para o Aurora MySQL, a replicação continua entre os ambientes. Para o Aurora PostgreSQL, o ambiente verde é promovido a um ambiente autônomo, então a replicação é interrompida.

A opção de excluir bancos de dados verdes não estará disponível no console depois da transição. Ao excluir uma implantação azul/verde usando a AWS CLI, você não poderá especificar a opção `--delete-target` se o status da implantação for `SWITCHOVER_COMPLETED`.

> **Important**
>
> Após a exclusão de uma implantação azul/verde, o RDS remove as proteções somente leitura do cluster de banco de dados de produção anterior. Se o parâmetro `read_only` estiver desabilitado para o cluster de banco de dados, ele começará a permitir operações de gravação novamente.

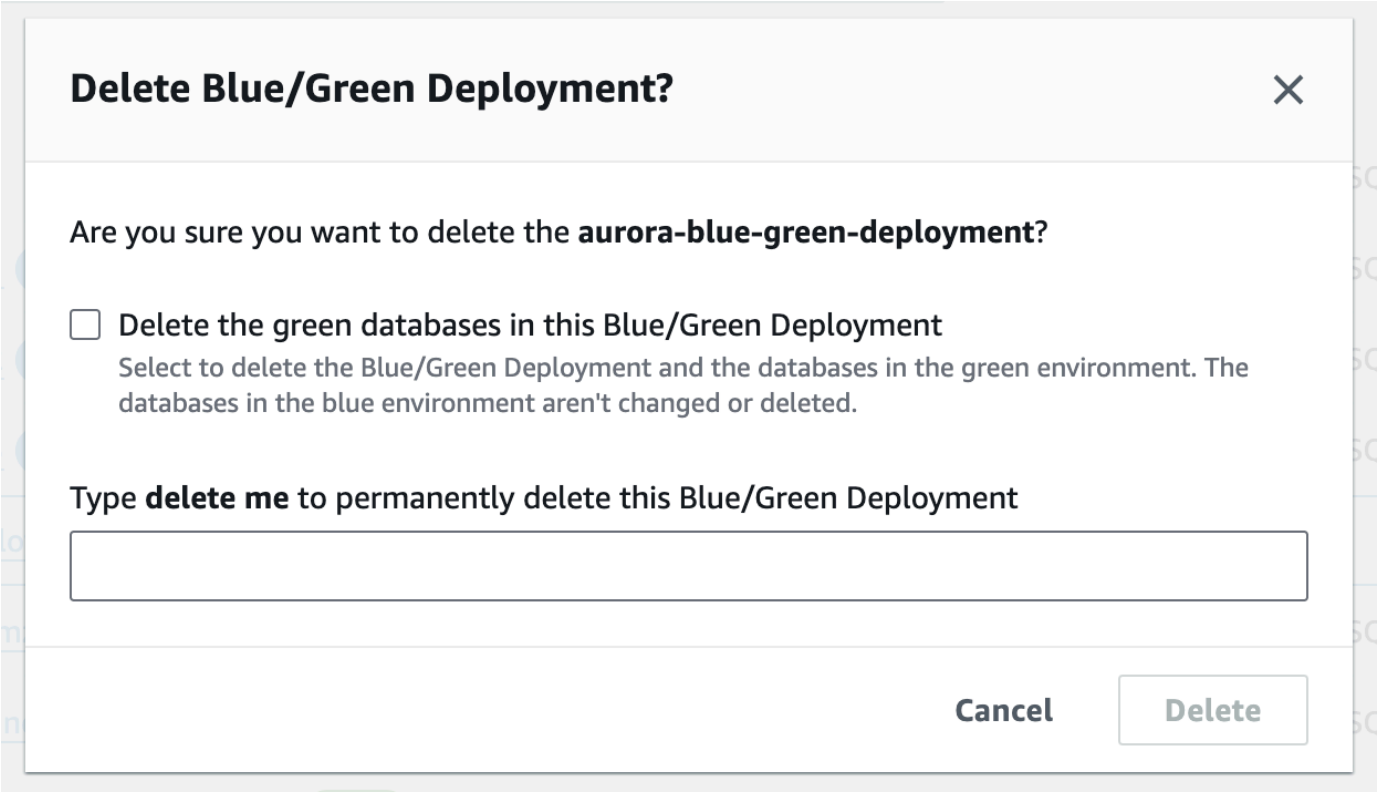
Você pode excluir a implantação azul/verde usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como excluir uma implantação azul/verde

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Databases (Bancos de dados) e, depois, selecione a implantação azul/verde que você deseja excluir.
3. Em Ações, escolha Excluir.

A janela Excluir implantação azul/verde? é exibida.



Para excluir os bancos de dados verdes, selecione Excluir os bancos de dados verdes nesta implantação azul/verde.

4. Digite **`delete me`** na caixa.
5. Escolha Delete (Excluir).

AWS CLI

Para excluir uma implantação azul/verde usando a AWS CLI, use o comando delete-blue-green-deployment com as seguintes opções:

- `--blue-green-deployment-identifier`: o ID do recurso da implantação azul/verde a ser excluída.
- `--delete-target`: especifica que o cluster de banco de dados no ambiente verde seja excluído. Você não poderá especificar essa opção se a implantação azul/verde tiver um status de `SWITCHOVER_COMPLETED`.
- `--no-delete-target`: especifica que o cluster de banco de dados no ambiente verde seja retido.

Example: excluir uma implantação azul/verde e o cluster de banco de dados no ambiente verde

Para Linux, macOS ou Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Para Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example: excluir uma implantação azul/verde, mas manter o cluster de banco de dados no ambiente verde

Para Linux, macOS ou Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Para Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

API do RDS

Para excluir uma implantação azul/verde usando a API do Amazon RDS, use a operação `DeleteBlueGreenDeployment` com os seguintes parâmetros:

- `BlueGreenDeploymentIdentifier`: o ID do recurso da implantação azul/verde a ser excluída.
- `DeleteTarget`: especifique `TRUE` se deseja excluir o cluster de banco de dados no ambiente verde ou `FALSE` para mantê-lo. Não poderá ser `TRUE` se a implantação azul/verde tiver um status de `SWITCHOVER_COMPLETED`.

Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora

Esses tópicos fornecem informações sobre como fazer backup e restauração de clusters do Amazon Aurora.

> **Tip**
>
> Os recursos de backup automático e os recursos de alta disponibilidade do Aurora ajudam a manter seus dados seguros sem exigir uma configuração extensa de sua parte. Antes de implementar uma estratégia de backup, saiba como o Aurora mantém várias cópias de seus dados e ajuda a acessá-las em várias instâncias de banco de dados e regiões da AWS. Para obter mais detalhes, consulte Alta disponibilidade do Amazon Aurora.

Tópicos

- Visão geral do backup e da restauração de um cluster de banco de dados do Aurora
- Reter backups automatizados
- Noções básicas do uso do armazenamento de backup do Amazon Aurora
- Criar um snapshot de cluster de banco de dados
- Restauração de um snapshot de um cluster de banco de dados
- Cópia de snapshot de cluster de banco de dados
- Compartilhar um snapshot do cluster de banco de dados
- Exportar dados do cluster de banco de dados para o Amazon S3
- Exportar dados de snapshot de cluster de banco de dados para o Amazon S3
- Restaurar um cluster de banco de dados para um horário especificado
- Excluir um snapshot de cluster de banco de dados
- Tutorial: Restaurar um cluster de banco de dados do Amazon Aurora de um snapshot de cluster de banco de dados

Visão geral do backup e da restauração de um cluster de banco de dados do Aurora

Os tópicos a seguir descrevem os backups do Aurora e como restaurar o cluster de banco de dados do Aurora.

Sumário

- Backups
  - Usar o AWS Backup
- Janela de backup
- Restaurar dados
- Clonagem de banco de dados para Aurora
- Retrocesso

Backups

O Aurora faz backup do volume de cluster automaticamente e mantém dados de restauração pela duração do período de retenção de backup. Os backups automáticos do Aurora são contínuos e incrementais para que você possa restaurar rapidamente em qualquer ponto do período de retenção de backup. Quando os dados do backup estão sendo gravados, não há nenhum impacto sobre a performance ou interrupção de serviço do banco de dados. É possível especificar um período de retenção de backup de 1 a 35 dias ao criar, modificar ou restaurar um cluster de banco de dados. Os backups automáticos do Aurora são armazenados no Amazon S3. Para ter mais informações sobre retenção de backups automatizados, consulte Reter backups automatizados.

Se você quiser reter dados além do período de retenção do backup, será possível fazer um snapshot dos dados no seu volume de cluster. Os snapshots do cluster de banco de dados do Aurora não expiram. Crie um novo cluster de banco de dados a partir do snapshot. Para obter mais informações, consulte Criar um snapshot de cluster de banco de dados.

Durante operações de restauração, existe a opção de especificar um período de retenção de backup para o cluster de banco de dados do Amazon Aurora. Quando esse valor não é definido explicitamente, o cluster restaurado herda o período de retenção de backup do snapshot ou do cluster de origem. Observe que esse comportamento de herança é exclusivo para operações de restauração. Ao criar um cluster, o sistema aplica períodos de retenção padrão.

> **Note**
>
> - Para os clusters de banco de dados Amazon Aurora, o período de retenção de backup padrão é de um dia, independentemente de como o cluster de banco de dados é criado.
> - Não é possível desativar backups automatizados no Aurora. O período de retenção de backup para o Aurora é controlado pelo cluster de Banco de Dados.

Os custos de armazenamento de backup dependem da quantidade de dados de backup e snapshot do Aurora mantidos e por quanto tempo você os mantêm. Para obter informações sobre o armazenamento associado a backups e snapshots do Aurora, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora. Para obter informações da definição de preço sobre o armazenamento de backup do Aurora, consulte Preços do Amazon RDS for Aurora. Depois que o cluster do Aurora associado a um snapshot for excluído, armazenar esse snapshot incorrerá em cobranças de armazenamento de backup padrão do Aurora.

Usar o AWS Backup

É possível usar o AWS Backup para gerenciar backups de clusters de banco de dados do Amazon Aurora.

Os snapshots gerenciados pelo AWS Backup são considerados snapshots manuais de cluster de banco de dados, mas não são contabilizados para a cota de snapshots de cluster de banco de dados do Aurora. Os snapshots que foram criados com o AWS Backup têm nomes com `awsbackup:job-`*`AWS-Backup-job-number`*. Para obter mais informações sobre AWS Backup, consulte o Guia do desenvolvedor do backup da AWS.

Também é possível usar o AWS Backup para gerenciar backups automatizados de clusters de banco de dados do Amazon Aurora. Se o cluster de banco de dados estiver associado a um plano de backup no AWS Backup, você pode usar esse plano de backup para recuperação para um ponto no tempo. Os backups automatizados (contínuos) gerenciados pelo AWS Backup têm nomes com `continuous:cluster-`*`AWS-Backup-job-number`*. Para obter mais informações, consulte Restaurar um cluster de banco de dados para um horário especificado usando o AWS Backup.

Janela de backup

Os backups automáticos são feitos diariamente durante a janela de backup escolhida. Se o backup exigir mais tempo do que o da janela de backup, ele continuará após a janela encerrar, até que

esteja concluído. A janela de backup não pode se sobrepor à janela de manutenção semanal do cluster do banco de dados.

Os backups automatizados do Aurora são contínuos e incrementais, mas a janela de backup é usada para criar um backup diário do sistema que é preservado dentro do período de retenção de backup. Você pode copiar o backup para preservá-lo fora do período de retenção.

> **Note**
>
> Ao criar um cluster de banco de dados usando o Console de gerenciamento da AWS, não é possível especificar uma janela de backup. No entanto, você pode especificar uma janela de backup ao criar um cluster de banco de dados usando AWS CLI ou a API do RDS.

Se você não especificar uma janela de backup preferencial ao criar o cluster de banco de dados, o Aurora atribuirá uma janela de backup padrão de 30 minutos. Essa janela é selecionada aleatoriamente em um bloco de tempo de 8 horas para cada Região da AWS. A tabela a seguir lista os blocos de tempo de cada Região da AWS a partir dos quais as janelas de backups padrão são atribuídas.

Nome da Região	Região	Bloco de hora
Leste dos EUA (Norte da Virgínia)	us-east-1	Das 3h às 11h (UTC)
Leste dos EUA (Ohio)	us-east-2	De 03:00 a 11:00 UTC
US West (N. Californi a)	us-west-1	De 06:00 a 14:00 UTC
US West (Oregon)	us-west-2	De 06:00 a 14:00 UTC
Africa (Cape Town)	af-south-1	De 03:00 a 11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	6h às 14h (UTC)
Ásia-Pacífico (Hyderabad)	ap-south-2	06h30 a 14h30 UTC

Nome da Região	Região	Bloco de hora
Ásia-Pacífico (Jacarta)	ap-southeast-3	Das 08h às 16h UTC
Ásia-Pacífico (Malásia)	ap-southeast-5	Das 9h às 17h (UTC)
Ásia-Pacífico (Melbourne)	ap-southeast-4	Das 11h às 19h UTC
Ásia-Pacífico (Mumbai)	ap-south-1	De 16:30 a 00:30 UTC
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	Das 13h às 21h (UTC)
Ásia-Pacífico (Osaka)	ap-northeast-3	De 00:00 a 08:00 UTC
Asia Pacific (Seoul)	ap-northeast-2	De 13:00 a 21:00 UTC
Ásia-Pacífico (Singapura)	ap-southeast-1	De 14:00 a 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	De 12:00 a 20:00 UTC
Ásia-Pacífico (Taipei)	ap-east-2	Das 9h às 17h (UTC)
Ásia-Pacífico (Tailândia)	ap-southeast-7	Das 8h às 16h (UTC)
Ásia-Pacífico (Tóquio)	ap-northeast-1	De 13:00 a 21:00 UTC
Canada (Central)	ca-central-1	Das 3h às 11h (UTC)
Oeste do Canadá (Calgary)	ca-west-1	Das 18h às 2h (UTC)
China (Pequim)	cn-north-1	De 06:00 a 14:00 UTC

Nome da Região	Região	Bloco de hora
China (Ningxia)	cn-northwest-1	De 06:00 a 14:00 UTC
Europe (Frankfurt)	eu-central-1	De 20:00 a 04:00 UTC
Europe (Ireland)	eu-west-1	De 22:00 a 06:00 UTC
Europe (London)	eu-west-2	Das 22:00 às 06:00 (UTC)
Europa (Milão)	eu-south-1	De 02:00 a 10:00 UTC
Europa (Paris)	eu-west-3	De 07:29 a 14:29 UTC
Europa (Espanha)	eu-south-2	De 02:00 a 10:00 UTC
Europe (Stockholm)	eu-north-1	De 23:00 a 07:00 UTC
Europa (Zurique)	eu-central-2	De 02:00 a 10:00 UTC
Israel (Tel Aviv)	il-central-1	Das 3h às 11h (UTC)
México (Central)	mx-central-1	Das 19h às 3h (UTC)
Oriente Médio (Barém)	me-south-1	6h às 14h (UTC)
Oriente Médio (Emirados Árabes Unidos)	me-central-1	Das 5h às 13h UTC
América do Sul (São Paulo)	sa-east-1	De 23:00 a 07:00 UTC
AWS GovCloud (Leste dos EUA)	us-gov-east-1	De 17:00 a 01:00 UTC
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	De 06:00 a 14:00 UTC

Restaurar dados

Você pode recuperar os dados criando um cluster de bancos de dados do Aurora por meio dos dados de backup retidos pelo Aurora, de um snapshot de cluster de banco de dados que você salvou ou de um backup automatizado retido. É possível restaurar rapidamente uma nova cópia de um cluster de banco de dados criado com os dados de backup a qualquer momento do período de retenção do backup. Devido à natureza contínua e incremental dos backups do Aurora durante o período de retenção do backup, você não precisa gerar snapshots frequentes dos dados para melhorar os tempos de restauração.

O tempo de restauração mais recente de um cluster de banco de dados é o ponto mais recente para o qual é possível restaurar o cluster de banco de dados. Normalmente, isso ocorre em até cinco minutos do horário atual para um cluster de banco de dados ativo ou cinco minutos do tempo de exclusão do cluster para um backup automatizado retido.

O tempo de restauração mais antigo especifica o ponto do período de retenção de backup até o qual é possível restaurar o volume do cluster.

Para determinar o tempo de restauração mais recente ou mais antigo para um cluster de banco de dados, procure os valores `Latest restorable time` ou `Earliest restorable time` no console do RDS. Para obter informações sobre como visualizar esses valores, consulte Visualização de backups retidos.

Você pode descobrir quando a restauração de um cluster de banco de dados foi encerrada verificando os valores `Latest restorable time` e `Earliest restorable time`. Esses valores retornam NULL até que a operação de restauração seja concluída. Não será possível solicitar uma operação de backup ou restauração se `Latest restorable time` ou `Earliest restorable time` retornar NULL.

Para obter informações sobre como restaurar um cluster de banco de dados para um período específico, consulte Restaurar um cluster de banco de dados para um horário especificado.

Clonagem de banco de dados para Aurora

Você também pode usar a clonagem de banco de dados para cloná-los a partir do seu cluster de banco de dados do Aurora, em vez de restaurar um snapshot do cluster de banco de dados. Os bancos de dados clonados usam apenas o espaço adicional mínimo quando são criados pela primeira vez. Os dados são copiados somente como alterações de dados, seja nos bancos de dados de origem ou nos clonados. Você pode fazer vários clones do mesmo cluster de banco de dados, ou

criar clones adicionais até mesmo de outros clones. Para obter mais informações, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora.

Retrocesso

O Aurora MySQL agora oferece suporte ao retrocesso de um cluster de banco de dados para um período específico, sem restaurar os dados de um backup. Para obter mais informações, consulte Retroceder um cluster de banco de dados Aurora.

Reter backups automatizados

Ao excluir um cluster de banco de dados provisionado ou do Aurora Serverless v2, é possível reter backups automatizados. Isso permite restaurar um cluster de banco de dados para um ponto no tempo específico, dentro do período de retenção de backup, mesmo após a exclusão do cluster.

Os backups automatizados retidos contêm snapshots de sistema e logs de transação de um cluster de banco de dados. Eles também incluem propriedades de cluster de banco de dados, como classe de instância de banco de dados, que são necessárias à restauração para um cluster ativo.

Restaure ou remova backups automatizados retidos usando o Console de gerenciamento da AWS, a API do RDS e a AWS CLI.

> **Note**
>
> Não é possível reter backups automatizados para clusters de banco de dados do Aurora Serverless v1.

Tópicos

- Período de retenção
- Visualização de backups retidos
- Custos de retenção
- Limitações
- Excluir backups automatizados retidos

Período de retenção

Os snapshots do sistema e os logs de transação em um backup automatizado retido expiram da mesma maneira que para o cluster de banco de dados de origem. As configurações do período de retenção do cluster de origem também se aplicam aos backups automatizados. Como não há novos snapshots ou logs criados para esse cluster, os backups automatizados retidos acabam expirando por completo. Após o término do período de retenção, você continua retendo snapshots manuais do cluster de banco de dados, mas todos os backups automatizados expiram.

Remova backups automatizados retidos usando o console, a AWS CLI ou a API do RDS. Para ter mais informações, consulte Excluir backups automatizados retidos.

Ao contrário de um backup automatizado retido, um snapshot final não expira. É altamente recomendável fazer um snapshot final, mesmo que você retenha backups automatizados, porque eles acabarão expirando.

Visualização de backups retidos

Para visualizar os backups automatizados retidos no console do RDS, selecione Backups automatizados no painel de navegação e, depois, Retido. Para visualizar instantâneos individuais associadas a um backup automatizado retido, escolha Snapshots no painel de navegação.
Ou descreva snapshots individuais associados a um backup automatizado retido. Lá, restaure diretamente uma instância de banco de dados de um desses snapshots.

Para descrever seus backups automatizados retidos com a AWS CLI, utilize o seguinte comando:

```
aws rds describe-db-cluster-automated-backups --db-cluster-resource-
id DB_cluster_resource_ID
```

Para descrever seus backups automatizados retidos utilizando a API do RDS, chame a ação `DescribeDBClusterAutomatedBackups` ação com o parâmetro `DbClusterResourceId`.

Custos de retenção

Não há cobrança adicional pelo armazenamento de backup de até 100% do armazenamento total do banco de dados Aurora para cada cluster de banco de dados Aurora. Também não há cobrança adicional de até um dia quando você retém backups automatizados após a exclusão de um cluster de banco de dados. Os backups retidos por mais de um dia são cobrados.

Não há cobrança adicional de logs de transação ou metadados de instância. Todas as outras regras de preço se aplicam a clusters restauráveis. Para ter mais informações, consulte a página Definição de preço do Amazon Aurora.

Limitações

As seguintes limitações se aplicam a backups automatizados retidos:

- O número máximo de backups automáticos retidos em uma região da AWS é 40. Ele não está incluído na cota para clusters de banco de dados. Você pode ter, ao mesmo tempo, até 40 clusters de banco de dados em execução, 40 instâncias de banco de dados em execução e 40 backups automatizados retidos para clusters de banco de dados.

Para ter mais informações, consulte Cotas no Amazon Aurora.

- Os backups automatizados retidos não contêm informações sobre grupos de parâmetros ou opções.
- Restaure um cluster excluído para um ponto no tempo que esteja dentro do período de retenção no momento da exclusão.
- Não é possível modificar um backup automatizado retido porque ele consiste em backups de sistema, em logs de transação e nas propriedades do cluster de banco de dados em vigor no momento em que você excluiu o cluster de origem.

Excluir backups automatizados retidos

Você pode excluir backups automatizados retidos quando eles não são mais necessários.

Console

Como excluir um backup automatizado retido

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Automated backups (Backups automatizados).
3. Selecione a guia Retido.

4. Escolha o backup automatizado retido que você deseja excluir.
5. Em Actions, selecione Delete.
6. Na página de confirmação, insira **`delete me`** e escolha Delete (Excluir).

AWS CLI

Você pode excluir um backup automatizado retido usando o comando delete-db-cluster-automated-backup da AWS CLI com a seguinte opção:

- `--db-cluster-resource-id`: o identificador do recurso para o cluster de banco de dados de origem.

  Você pode encontrar o identificador de recurso do cluster de banco de dados de origem de um backup automatizado retido executando o comando describe-db-cluster-automated-backups da AWS CLI.

Example

Este exemplo exclui o backup automatizado retido para o cluster de banco de dados de origem que tem o ID do recurso `cluster-123ABCEXAMPLE`.

Para Linux, macOS ou Unix:

```
aws rds delete-db-cluster-automated-backup \
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

Para Windows:

```
aws rds delete-db-cluster-automated-backup ^
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

API do RDS

É possível excluir um backup automatizado retido usando a operação de API DeleteDBClusterAutomatedBackup do Amazon RDS com o seguinte parâmetro:

- `DbClusterResourceId`: o identificador do recurso para o cluster de banco de dados de origem.

  Você pode encontrar o identificador de recurso da instância de banco de dados de origem de um backup automatizado retido usando a operação de API DescribeDBClusterAutomatedBackups do Amazon RDS.

Noções básicas do uso do armazenamento de backup do Amazon Aurora

O Amazon Aurora mantém dois tipos de backup: backups automatizados (contínuos) e snapshots.

Armazenamento de backups automatizados

O backup automatizado (contínuo) de um cluster armazena incrementalmente todas as alterações do banco de dados dentro de um período de retenção especificado para poder restaurar qualquer ponto temporal dentro desse período de retenção. Os períodos de retenção podem variar de 1 a 35 dias. Os backups automatizados são incrementais e cobrados com base na quantidade de armazenamento necessária para restaurar qualquer ponto temporal dentro do período de retenção.

O Aurora também fornece uma quantidade gratuita de uso de backup. Essa quantidade gratuita de uso é igual ao tamanho do volume do cluster mais recente (conforme representado pela métrica `VolumeBytesUsed` do Amazon CloudWatch). Essa quantidade é subtraída do uso calculado de backups automatizados. Também não há cobrança por um backup automatizado cujo período de retenção seja de apenas um dia.

Por exemplo, seu backup automatizado tem um período de retenção de sete dias e você deseja restaurar seu cluster para o estado de quatro dias atrás. O Aurora usa os dados incrementais armazenados no backup automatizado para recriar o estado do cluster naquele exato momento, há quatro dias.

O backup automatizado armazena todas as informações necessárias para poder restaurar qualquer ponto temporal do cluster na janela de retenção. Isso significa que ele armazena todas as alterações durante a janela de retenção, incluindo gravações de novas informações ou exclusão de informações existentes. Para bancos de dados em que ocorrem muitas mudanças, o tamanho do backup automatizado aumenta com o tempo. Depois que um banco de dados parar de sofrer alterações, você pode esperar que o tamanho do backup automatizado diminua, à medida que as alterações armazenadas anteriormente saem da janela de retenção.

O uso total cobrado pelo backup automatizado nunca excede o tamanho do volume cumulativo do cluster durante o período de retenção. Por exemplo, se o período de retenção for de sete dias e o volume do cluster for de 100 GB por dia, o uso cobrado do backup automatizado nunca excederá 700 GB (100 GB * 7).

Armazenamento de snapshots

Os snapshots de cluster de banco de dados são sempre backups completos que capturam o tamanho do volume do cluster quando você os cria. Quer esteja gerando snapshots manualmente ou por meio de um plano do AWS Backup, o Aurora os considera snapshots manuais. O Aurora fornece armazenamento gratuito e ilimitado para snapshots dentro do período de retenção de backups automatizados. Depois que um snapshot manual ultrapassa esse período, ele incorre em cobranças por GB/mês. Os snapshots automatizados do sistema permanecerão gratuitos se você os copiar. Como os backups automatizados não abrangem cópias de snapshots, a AWS sempre cobra por elas.

Para obter informações gerais sobre backups do Aurora, consulte Backups. Para ter informações sobre os preços de armazenamento de backup do Aurora, consulte a página Definição de preço do Amazon Aurora.

Métricas do Amazon CloudWatch para armazenamento de backup do Aurora

Monitore os clusters do Aurora e crie relatórios usando métricas do Amazon CloudWatch por meio do console do CloudWatch. Use as métricas a seguir do CloudWatch para examinar e monitorar a quantidade de armazenamento usado pelos backups do Aurora. Essas métricas são calculadas de maneira independente para cada cluster de banco de dados do Aurora.

- `BackupRetentionPeriodStorageUsed`: representa a quantidade de armazenamento de backup usado, em bytes, para armazenar backups automáticos no momento atual.
  - O valor depende do tamanho do volume do cluster e do número de alterações (gravações e atualizações) feitas no cluster de banco de dados durante o período de retenção. Isso ocorre porque o backup automatizado deve armazenar todas as alterações incrementais feitas no cluster para poder restaurar qualquer ponto temporal.
  - Essa métrica não subtrai o nível gratuito de uso de backups que o Aurora fornece.
  - Essa métrica emite um único ponto de dados diário para o uso de backup automatizado registrado naquele dia.
- `SnapshotStorageUsed`: representa a quantidade de armazenamento de backup usado, em bytes, para armazenar snapshots manuais além do período de retenção do backup automatizado.
  - O valor depende do número de snapshots que você mantém depois do período de retenção do backup automatizado e do tamanho de cada snapshot.

- O tamanho de cada snapshot é o tamanho do volume do cluster no momento em que você faz o snapshot.
- Os snapshots são backups completos, não incrementais.
- Essa métrica emite um ponto de dados diário para cada snapshot que será cobrado. Para recuperar seu uso diário total de snapshots, calcule a soma dessa métrica em um período de um dia.

- `TotalBackupStorageBilled`: representa as métricas de todo o uso de backups cobrado, em bytes, para o cluster:

  `BackupRetentionPeriodStorageUsed + SnapshotStorageUsed - free tier`

  - Essa métrica emite um ponto de dados diário para o valor de `BackupRetentionPeriodStorageUsed` menos o nível gratuito de uso de backups que o Aurora fornece. Esse nível gratuito é igual ao tamanho mais recente registrado do volume do cluster de banco de dados. Esse ponto de dados representa o uso real cobrado pelo backup automatizado.
  - Essa métrica emite pontos de dados diários individuais para todos os valores de `SnapshotStorageUsed`.
  - Para recuperar seu uso diário total cobrado pelos backups, calcule a soma dessa métrica em um período de um dia. Isso soma todo o uso cobrado por snapshots com o uso cobrado pelo backup automatizado para obter o uso total cobrado pelos backups.

Para obter mais informações sobre como usar as métricas do CloudWatch, consulte Disponibilidade de métricas Aurora no console Amazon RDS.

Calcular o uso do armazenamento dos backups

O uso de um backup automatizado é calculado pela análise de todos os registros incrementais que devem ser armazenados para que seja possível restaurar qualquer ponto temporal dentro do período de retenção do backup. Essas alterações incluem não apenas o número de operações de gravação, mas também o tamanho e o escopo das modificações de dados. Cada tipo de operação (INSERT, UPDATE e DELETE) cria registros de alterações que devem ser retidos para a recuperação para um ponto no tempo. Portanto, embora dois bancos de dados possam ter o mesmo número de operações de gravação (IOPS), os respectivos requisitos de armazenamento de backup podem diferir significativamente com base no volume de dados que estão sendo modificados em cada transação.

Por exemplo, você tem um backup automatizado com período de retenção de sete dias. O tamanho do volume do seu cluster pouco antes do período de retenção era de 100 GB, então essa é a menor quantidade que o Aurora precisa armazenar. Depois, você tem a seguinte atividade para os 7 dias seguintes, em que o tamanho incremental do registro é a quantidade de armazenamento necessária para armazenar os registros de alterações provenientes das gravações e atualizações do seu banco de dados.

Dia	Tamanho incremental do registro (GB)
1	10
2	15
3	25
4	20
5	10
6	25
7	30
Total	135

Esses dados significam que o uso calculado do seu backup automatizado é o seguinte:

```
100 GB (volume size before retention period) + 135 GB (size of incremental records) =
 235 GB total backup usage
```

Depois, o uso cobrado subtrai o nível gratuito de uso. Suponha que o tamanho mais recente do seu volume seja 200 GB:

```
235 GB total backup usage - 200 GB (latest volume size) = 35 GB billed backup usage
```

Perguntas frequentes

Quando acontece a cobrança pelos snapshots?

São cobrados os snapshots manuais que estão fora do período de retenção do backup automatizado (mais antigos).

O que é um snapshot manual?

Um snapshot manual é um snapshot ao qual uma das seguintes condições se aplica:

- Solicitado manualmente por você.
- Gerado por um serviço de backup automatizado, como o AWS Backup.
- Copiado de um snapshot automatizado do sistema para mantê-lo após o período de retenção.

O que acontecerá com meus snapshots manuais se eu excluir meu cluster de banco de dados?

Snapshots manuais só expiram quando você os exclui.

Quando você exclui seu cluster de banco de dados, os snapshots manuais que você gerou continuam existindo. Se antes esses snapshots não estavam sendo cobrados porque estavam dentro do período de retenção de backup automatizado, agora eles não estão mais cobertos e todos começam a ser cobrados em tamanho real pelo uso.

Como posso reduzir meus custos de armazenamento de backups?

Há algumas maneiras de reduzir os custos relacionados ao uso de backups:

- Exclua os snapshots manuais que estão fora do período de retenção do backup automatizado. Isso inclui os snapshots que você gerou e os snapshots que seu plano do AWS Backup pode ter gerado. Verifique seu plano do AWS Backup para ter certeza de que ele não está mantendo snapshots indesejados depois do período de retenção.
- Avalie suas gravações e atualizações no banco de dados para ver se é possível reduzir o número de alterações que está fazendo. Como nosso backup automatizado armazena todas as alterações incrementais dentro do período de retenção, reduzir o número de atualizações feitas também reduz as cobranças pelo backup automatizado.
- Avalie se a redução do período de retenção do backup automatizado faria sentido. Reduzir o período de retenção significa que o backup vai armazenar menos dias de dados incrementais, o que pode reduzir o custo geral do backup. No entanto, reduzir esse período de retenção também poderá fazer com que alguns snapshots comecem a ser cobrados, pois vão sair do

período de retenção. Verifique todos os custos extras de snapshots que possam incorrer antes de decidir se esse é o caminho certo para você.

Como o armazenamento de backups é cobrado?

O armazenamento de backup é cobrado por GB mensal.

Isso significa que o uso do armazenamento de backup é cobrado como a média ponderada do uso durante determinado mês. Aqui estão alguns exemplos para um mês de 30 dias:

- O uso cobrado pelo backup é de 100 GB em todos os 30 dias do mês. Sua cobrança é a seguinte:

```
(100 GB * 30) / 30 = 100 GB-month
```

- O uso cobrado pelo backup é de 100 GB nos primeiros 15 dias do mês, depois 0 GB nos últimos 15. Sua cobrança é a seguinte:

```
(100 GB * 15 + 0 GB * 15) / 30 = 50 GB-month
```

- O uso cobrado pelo backup é de 50 GB nos primeiros 10 dias do mês, 100 GB nos próximos 10 dias, depois 150 GB nos últimos 10. Sua cobrança é a seguinte:

```
(50 GB * 10 + 100 GB * 10 + 150 GB * 10) / 30 = 100 GB-month
```

Como a configuração de retrocesso do cluster de banco de dados afeta o uso do armazenamento de backup?

A configuração de retrocesso para um cluster de banco de dados do Aurora não afeta o volume dos dados de backup para esse cluster. A Amazon cobra o armazenamento de dados do retrocesso separadamente. Para obter informações sobre retrocesso no Aurora, consulte a página Definição de preço do Amazon Aurora.

Como os custos de armazenamento se aplicam aos snapshots compartilhados?

Se você compartilhar um snapshot com outro usuário, a propriedade desse snapshot continuará sendo sua. Os custos de armazenamento se aplicam ao proprietário do snapshot. Se excluir um snapshot compartilhado pertencente a você, ninguém poderá acessá-lo.

Para manter o acesso a um snapshot compartilhado pertencente a outra pessoa, você pode copiar esse snapshot. Isso faz com que você seja o proprietário do novo snapshot. Qualquer custo de armazenamento do snapshot copiado será aplicado à sua conta.

Para obter mais informações sobre o compartilhamento de snapshots, consulte Compartilhar um snapshot do cluster de banco de dados. Para obter mais informações sobre a cópia de snapshots, consulte Cópia de snapshot de cluster de banco de dados.

Criar um snapshot de cluster de banco de dados

O Amazon RDS cria um snapshot do volume de armazenamento do cluster de banco de dados, fazendo backup de todo o cluster de banco de dados, e não apenas dos bancos de dados individuais. Ao criar um snapshot do cluster de banco de dados, você precisa identificar de qual cluster de banco de dados deseja fazer backup e dar um nome ao snapshot do cluster de banco de dados para que você possa restaurá-lo depois. O tempo necessário para criar um snapshot do cluster de banco de dados varia de acordo com o tamanho dos bancos de dados. Como o snapshot inclui todo o volume de armazenamento, o tamanho de arquivos, como arquivos temporários, também afeta o tempo necessário para criar o snapshot.

> **Note**
>
> Seu cluster de banco de dados deve estar no estado `available` para tirar um snapshot do cluster de banco de dados.

Diferentemente dos backups automatizados, os snapshots manuais não estão sujeitos ao período de retenção de backup. Os snapshots não expiram.

Para backups de muito longo prazo, recomendamos exportar dados de snapshot para o Amazon S3. Se a versão principal do mecanismo de banco de dados não for mais compatível, você não poderá restaurar para essa versão a partir de um snapshot. Para obter mais informações, consulte Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.

Você pode criar um snapshot do cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para criar um snapshot de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.

   A lista de snapshots manuais aparece.
3. Selecione Take Snapshot (Fazer snapshot).

A janela Take snapshot de banco de dados (Fazer snapshot de banco de dados) é exibida.

4. Para Tipo de snapshot, selecione Cluster de banco de dados.
   Crie um snapshot de banco de dados.
5. Escolha o cluster de banco de dados para o qual você deseja criar um snapshot.
6. Insira o Nome do snapshot.
7. Selecione Take Snapshot (Fazer snapshot).

   A lista Snapshots manuais é exibida com o status do novo snapshot do cluster de banco de dados mostrado como `Creating`. Com o status é `Available`, você pode ver a hora de criação.

AWS CLI

Ao criar um snapshot do cluster de banco de dados usando a AWS CLI, você precisa identificar de qual cluster de banco de dados deseja fazer backup e dar um nome ao snapshot do cluster de banco de dados para que você possa restaurá-lo depois. Você pode fazer isso usando o comando AWS CLI da `create-db-cluster-snapshot` com os seguintes parâmetros:

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

Neste exemplo, você cria um snapshot do cluster de banco de dados chamado *`mydbclustersnapshot`* para um cluster de banco de dados chamado *`mydbcluster`*.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mydbcluster \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Para Windows:

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mydbcluster ^
```

```
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API do RDS

Ao criar um snapshot do cluster de banco de dados usando a API do Amazon RDS, você precisa identificar de qual cluster de banco de dados deseja fazer backup e dar um nome ao snapshot do cluster de banco de dados para que você possa restaurá-lo depois. Você pode fazer isso usando o comando `CreateDBClusterSnapshot` da API do Amazon RDS com os seguintes parâmetros:

- DBClusterIdentifier
- DBClusterSnapshotIdentifier

Como determinar se o snapshot do cluster de banco de dados está disponível

É possível conferir se o snapshot do cluster de banco de dados está disponível procurando em Snapshots na guia Manutenção e backups, na página de detalhes do cluster no Console de gerenciamento da AWS usando o comando da CLI `describe-db-cluster-snapshots` ou usando a ação da API `DescribeDBClusterSnapshots`.

Também é possível usar o comando `wait db-cluster-snapshot-available` da CLI para sondar a API a cada 30 segundos até que o snapshot esteja disponível.

Restauração de um snapshot de um cluster de banco de dados

O Amazon RDS cria um snapshot do volume de armazenamento do cluster de banco de dados, fazendo backup de todo o cluster de banco de dados, e não apenas dos bancos de dados individuais. É possível criar um cluster de banco de dados com a restauração de um snapshot do banco de dados. Você fornece o nome do snapshot do cluster de banco de dados do qual restaurar e o nome do novo cluster de banco de dados criado na operação de restauração. Não é possível restaurar de um snapshot de cluster de banco de dados para um cluster de banco de dados existente. Um novo cluster de banco de dados é criado quando você restaura.

Important

Você não pode restaurar um snapshot em uma versão do mecanismo de banco de dados que tenha passado da data de término do suporte padrão do Aurora. Você só poderá acessar um banco de dados depois que ele for atualizado com êxito para uma versão aceita. Para acessar mais informações sobre as versões de mecanismo do Aurora aceitos, consulte Suporte estendido do Amazon RDS com Amazon Aurora.
Se uma atualização para uma versão aceita do seu cluster falhar, o status do cluster mudará para `upgrade_failed` e o Aurora criará um snapshot final com o prefixo `rds-final`. Para acessar seu banco de dados restaurado na versão obsoleta após uma falha na atualização, entre em contato com o AWS Support.

É possível usar o cluster de banco de dados restaurado quando o status for `available`.

É possível usar o CloudFormation para restaurar um cluster de banco de dados de um snapshot de instância de banco de dados. Para obter mais informações, consulte AWS::RDS::DBCluster, no Guia do usuário do AWS CloudFormation.

Note

Compartilhar um snapshot de cluster de banco de dados manual, criptografado ou não, permite às contas da AWS autorizadas restaurarem diretamente um cluster de banco de dados a partir do snapshot em vez de fazer uma cópia dele e restaurar a partir daí. Para obter mais informações, consulte Compartilhar um snapshot do cluster de banco de dados.

Consulte informações sobre como restaurar um cluster de banco de dados do Aurora ou um cluster global com uma versão do Suporte estendido do RDS, consulte Restauração de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS.

Considerações de grupos de parâmetros

Recomendamos manter o grupo de parâmetros de banco de dados e o grupo de parâmetros de cluster de banco de dados para todos os snapshots do cluster de banco de dados criados, para que você possa associar um cluster de banco de dados restaurado ao grupo de parâmetros correto.

O grupo de parâmetros de banco de dados padrão e o grupo de parâmetros de cluster de banco de dados estão associados ao cluster restaurado, a menos que você escolha outros diferentes. Nenhuma configuração de parâmetro personalizada está disponível nos grupos de parâmetros padrão.

Você pode especificar o grupo de parâmetros ao restaurar o cluster de banco de dados.

Para obter mais informações sobre grupos de parâmetros de banco de dados e grupos de parâmetros de cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Considerações sobre os grupos de segurança

Ao restaurar um cluster de banco de dados, a nuvem privada virtual (VPC) padrão, o grupo de sub-redes de banco de dados e o grupo de segurança da VPC são associados à instância restaurada, a menos que você escolha outros diferentes.

- Se estiver usando o console do Amazon RDS, você poderá especificar um grupo de segurança da VPC personalizado para associar ao cluster ou criar um novo grupo de segurança da VPC.
- Se estiver usando a AWS CLI, você poderá especificar um grupo de segurança da VPC personalizado para associar ao cluster incluindo a opção `--vpc-security-group-ids` no comando `restore-db-cluster-from-snapshot`.
- Se você estiver usando a API do Amazon RDS, será possível incluir o parâmetro `VpcSecurityGroupIds.VpcSecurityGroupId.N` na ação `RestoreDBClusterFromSnapshot`.

Assim que a restauração for concluída e o novo cluster de banco de dados estiver disponível, também será possível alterar as configurações da VPC modificando o cluster de banco de dados. Para obter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Considerações sobre o Amazon Aurora

Com o Aurora, você restaura um snapshot de cluster de banco de dados em um cluster de banco de dados.

Com o Aurora MySQL e o Aurora PostgreSQL, você também pode recuperar um snapshot do cluster de banco de dados em um cluster de banco de dados Aurora Serverless. Para obter mais informações, consulte Restaurar um cluster de banco de dados do Aurora Serverless v1.

Com o Aurora MySQL, você pode recuperar um snapshot de cluster de banco de dados de um cluster sem consulta paralela para um cluster com consulta paralela. Como a consulta paralela normalmente é usada com tabelas grandes, o mecanismo de snapshot é a maneira mais rápida de ingerir grandes volumes de dados para um cluster do Aurora MySQL habilitado para consulta paralela. Para obter mais informações, consulte Consulta paralela do Amazon Aurora MySQL.

Restauração a partir de um snapshot

Restaure um cluster de banco de dados de um snapshot de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para restaurar um cluster de banco de dados de um snapshot do cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Snapshots.
3. Escolha o snapshot do cluster de banco de dados a partir do qual você deseja restaurar.
4. Em Actions (Ações), escolha Restore snapshot (Restaurar snapshot).

   A página Restaurar snapshot é exibida.
5. Escolha a versão do mecanismo de banco de dados para a qual você deseja restaurar o cluster de banco de dados.

   Por padrão, o snapshot é restaurado na mesma versão do mecanismo de banco de dados do cluster de banco de dados de origem, se essa versão estiver disponível.
6. Em DB instance identifier (Identificador da instância de banco de dados), insira o nome da instância de banco de dados restaurada. Observe que o Amazon RDS deriva o identificador do cluster de banco de dados do identificador de instância de banco de dados que você especificar.

7. Especifique outras configurações, como a configuração de armazenamento de cluster de banco de dados.

   Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

8. Escolha Restore DB Cluster (Restaurar cluster de banco de dados).

AWS CLI

Para restaurar um cluster de banco de dados a AWS CLIpartir de um snapshot de cluster de banco de dados, use o comando restore-db-cluster-from-snapshot da .

Neste exemplo, restaure a partir de um snapshot de cluster de banco de dados chamado `mydbclustersnapshot`. Restaure em um novo cluster de banco de dados chamado `mynewdbcluster`.

É possível especificar outras configurações, como a versão do mecanismo de banco de dados. Se você não especificar uma versão do mecanismo, o cluster de banco de dados será restaurado para a versão padrão do mecanismo.

Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

Example

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine aurora-mysql|aurora-postgresql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewdbcluster ^
    --snapshot-identifier mydbclustersnapshot ^
```

```
    --engine aurora-mysql|aurora-postgresql
```

Depois que o cluster de banco de dados tiver sido restaurado, é necessário adicioná-lo ao grupo de segurança usado pelo cluster de banco de dados usado para criar o snapshot de cluster de banco de dados se quiser a mesma funcionalidade que a do cluster de banco de dados anterior.

> **Important**
>
> Se você usar o console para restaurar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância de banco de dados primária (leitura) para o cluster de banco de dados. Se você usar a AWS CLI para restaurar um cluster de banco de dados, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados. Se você não criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `creating`.
> Chame o comando create-db-instance da AWS CLI para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados como o valor da opção `--db-cluster-identifier`.

API do RDS

Para restaurar um cluster de banco de dados de um snapshot de cluster de banco de dados, chame a operação de API RestoreDBClusterFromSnapshot do RDS com os seguintes parâmetros:

- `DBClusterIdentifier`
- `SnapshotIdentifier`

> **Important**
>
> Se você usar o console para restaurar um cluster de banco de dados, o Amazon RDS criará automaticamente a instância de banco de dados primária (leitura) para o cluster de banco de dados. Se você usar a API do RDS para restaurar um cluster de banco de dados, será necessário criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados. Se você não criar a instância de banco de dados primária, os endpoints do cluster de banco de dados permanecerão no status `creating`.

Chame a operação de API CreateDBInstance do RDS para criar a instância primária para o cluster de banco de dados. Inclua o nome de um cluster de banco de dados assim com o valor do parâmetro `DBClusterIdentifier`.

Cópia de snapshot de cluster de banco de dados

Com o Amazon Aurora, é possível copiar backups automatizados ou snapshots manuais de clusters de banco de dados. Depois de copiar um snapshot, a cópia é um snapshot manual. É possível fazer várias cópias de um backup automatizado ou snapshot manual, mas cada um deve ter um identificador exclusivo.

É possível copiar um snapshot dentro da mesma Região da AWS ou entre Regiões da AWS e também copiar snapshots compartilhados. Você pode copiar snapshots para outra Região da AWS ou conta em uma única etapa.

> **Note**
>
> A Amazon cobra com base na quantidade de dados de backup e snapshot do Amazon Aurora e no período em que você os mantém. Para obter informações sobre o armazenamento associado a backups e snapshots do Aurora, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora. Para obter informações da definição de preço sobre o armazenamento do Aurora, consulte Preços do Amazon RDS for Aurora.

Analise as limitações e as considerações para a cópia de snapshots do cluster de banco de dados. Para copiar snapshots do cluster de banco de dados, consulte um dos tópicos a seguir.

- Copiar um snapshot de cluster de banco de dados com o Console de gerenciamento da AWS
- Copiar um snapshot de cluster de banco de dados não criptografado usando a AWS CLI ou a API do Amazon RDS
- Copiar um snapshot de cluster de banco de dados criptografado usando a AWS CLI ou a API do Amazon RDS
- Copiar um snapshot de cluster de banco de dados entre contas

Copiar um snapshot de cluster de banco de dados com o Console de gerenciamento da AWS

Use os procedimentos neste tópico para copiar um snapshot de cluster de banco de dados. Se o mecanismo de banco de dados de origem for o Aurora, o snapshot será um snapshot de cluster de banco de dados.

Para cada conta da AWS, você pode copiar até cinco snapshots de cluster de banco de dados ao mesmo tempo de uma região da Região da AWS para outra. A cópia de snapshots de cluster de banco de dados criptografados e não criptografados é aceita. Se você copiar um snapshot de cluster de banco de dados para outra Região da AWS, criará um snapshot de cluster de banco de dados manual que é retido naquela Região da AWS. A cópia de um snapshot de cluster de banco de dados fora da Região da AWS de origem resultará em cobranças de transferência de dados do Amazon RDS.

Para ter mais informações sobre a definição e preço da transferência de dados, consulte Definição de preço do Amazon RDS.

Depois que a cópia do snapshot de cluster de banco de dados tiver sido criada na nova Região da AWS, ela se comportará da mesma forma como todos os outros snapshots de cluster de banco de dados na Região da AWS.

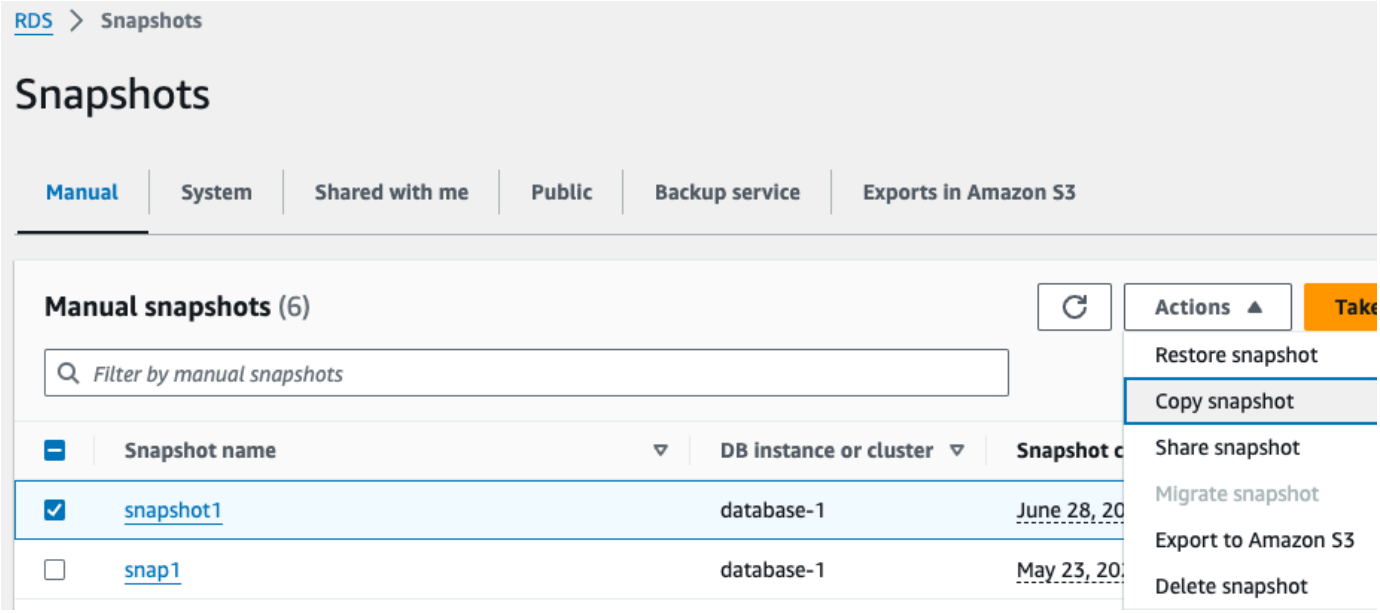
Este procedimento copia snapshots de cluster de banco de dados criptografados ou não na mesma Região da AWS ou entre regiões.

Para cancelar uma operação de cópia quando ela estiver em andamento, exclua o snapshot do cluster de banco de dados de destino enquanto ele estiver no status copying (cópia).

Antes de copiar um snapshot de cluster de banco de dados, revise Limitações e Considerações sobre a cópia de snapshots.

Para copiar um snapshot de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.
3. Selecione o snapshot de cluster de banco de dados a ser copiado.
4. Selecione Ações e, depois, Copiar snapshot.



5. (Opcional) Para copiar o snapshot de cluster de banco de dados para outra Região da AWS, escolha essa Região da AWS como a Destination Region (Região de destino).
6. Digite o nome da cópia do snapshot de cluster de banco de dados em New DB Snapshot Identifier (Novo identificador de DB snapshot).
7. Para copiar tags e valores do snapshot para a cópia do snapshot, escolha Copy Tags (Copiar tags).
8. Escolha Copy Snapshot (Copiar snapshot).

Limitações

Algumas limitações ao copiar snapshots:

- Não é possível copiar um snapshot das seguintes ou para elas Regiões da AWS:
  - China (Beijing)
  - China (Ningxia)
- Você pode copiar um snapshot entre a AWS GovCloud (Leste dos EUA) e AWS GovCloud (US-West). No entanto, não é possível copiar um snapshot entre essas regiões AWS GovCloud (US) e Regiões da AWS comerciais.
- Se você excluir um snapshot de origem antes que o snapshot de destino fique disponível, a cópia do snapshot poderá falhar. Verifique se o snapshot de destino possui um status `AVAILABLE` antes de excluir um snapshot de origem.

- É possível ter até cinco solicitações de cópia de snapshot em andamento para uma única região de destino por conta.
- Quando você solicita várias cópias de snapshot para a mesma instância de banco de dados de origem, elas são colocadas em fila internamente. As cópias solicitadas posteriormente não serão iniciadas enquanto as cópias de snapshots anteriores não forem concluídas. Para obter mais informações, consulte Por que a criação de snapshots do EC2 AMI ou do EBS está lenta? na Central de Conhecimento da AWS.
- Dependendo das Regiões da AWS envolvidas e da quantidade de dados a serem copiados, uma cópia de snapshot entre regiões pode levar horas para ser concluída. Em alguns casos, pode haver um grande número de solicitações de cópia de snapshot entre regiões a partir de uma determinada região de origem da . Nesses casos, o Amazon RDS pode colocar novas solicitações de cópia entre regiões vindas daquela região da de origem em uma fila até que algumas cópias em andamento sejam concluídas. Nenhuma informação de progresso é exibida sobre solicitações de cópia enquanto elas estão na fila. As informações sobre o andamento são exibidas quando a cópia é iniciada.
- O Aurora não comporta snapshots incrementais. As cópias de snapshot de cluster de bancos de dados do Aurora são sempre armazenadas como cópias completas. Uma cópia completa de snapshot contém todos os dados e metadados necessários para restaurar o cluster de banco de dados.

Considerações sobre a cópia de snapshots

As considerações a seguir devem ser observadas ao copiar snapshots.

Tópicos

- Considerações sobre a cópia de snapshots compartilhados
- Considerações sobre cópia de snapshots do cluster de banco de dados criptografado
- Considerações sobre cópia de snapshots entre regiões
- Considerações sobre grupos de parâmetros

Considerações sobre a cópia de snapshots compartilhados

Você pode copiar snapshots compartilhados com você por outras contas da AWS. Em alguns casos, você pode copiar um snapshot criptografado que foi compartilhado de outra conta da AWS. Nesses

casos, você deve ter acesso à AWS KMS key usada para criptografar o snapshot. Para obter mais informações, consulte Compartilhamento de snapshots criptografados.

Cópia entre regiões e entre contas em uma única etapa

Para copiar um snapshot entre regiões e entre contas em uma única ação, você deve primeiro compartilhá-lo com a conta da AWS de destino. Se o snapshot estiver criptografado, você também deverá compartilhar a chave do AWS KMS com a conta da AWS de destino. Se o snapshot for criptografado com a chave do AWS KMS padrão, você deverá primeiro copiá-lo para criptografá-lo novamente com uma chave gerenciada pelo cliente antes de compartilhá-lo com a conta de destino. Depois de compartilhado, você pode iniciar uma cópia para essa conta (na região ou entre regiões) da conta de destino.

Considerações sobre cópia de snapshots do cluster de banco de dados criptografado

É possível copiar um snapshot que tenha sido criptografado usando uma chave do KMS. Se você copiar um snapshot criptografado, a cópia desse snapshot também deverá ser criptografada. Se você copiar um snapshot criptografado dentro da mesma Região da AWS, poderá criptografar a cópia com a mesma chave do KMS que o snapshot original. Ou você pode especificar uma chave do KMS diferente.

Se você copiar um snapshot criptografado entre regiões, deverá especificar uma chave do KMS válida na de destino Região da AWS. Pode ser uma chave do KMS específica da região ou uma chave de várias regiões. Para obter mais informações sobre chaves do KMS de várias regiões, consulte Usar chaves de várias regiões no AWS KMS.

Para ter mais informações sobre o gerenciamento de chaves AWS KMS para o Amazon RDS, consulte AWS KMS keyGerenciamento de.

O snapshot de origem permanece criptografado ao longo do processo de cópia. Para obter mais informações, consulte Limitações dos clusters de banco de dados criptografados do Amazon Aurora.

> **Note**
>
> Não é possível criptografar os snapshots de cluster de bancos de dados do Amazon Aurora ao copiá-los.

Para copiar snapshots do cluster de banco de dados criptografados, consulte os tópicos a seguir.

- Copiar um snapshot de cluster de banco de dados criptografado usando a AWS CLI ou a API do Amazon RDS
- Copiar um snapshot de cluster de banco de dados entre contas

Considerações sobre cópia de snapshots entre regiões

É possível copiar snapshots de um cluster de banco de dados entre Regiões da AWS. No entanto, existem certas restrições e considerações para a cópia de snapshot entre regiões.

Dependendo das Regiões da AWS envolvidas e da quantidade de dados a serem copiados, uma cópia de snapshot entre regiões pode levar horas para ser concluída.

Em alguns casos, pode haver um grande número de solicitações de cópia de snapshot entre regiões de determinada de origem Região da AWS. Nesses casos, o Amazon RDS pode colocar novas solicitações de cópia entre regiões da Região da AWS de origem em uma fila até que algumas cópias em andamento sejam concluídas. Nenhuma informação de progresso é exibida sobre solicitações de cópia enquanto elas estão na fila. As informações sobre o andamento são exibidas quando a cópia é iniciada.

Taxas de transferência de dados se aplicam à cópia de snapshot entre regiões. A cópia de snapshots entre regiões cria cópias completas nos dados de destino, mas as cobranças de transferência de dados são incrementais. Os dados incrementais incluem os novos dados que foram adicionados ao banco de dados do cliente desde a última cópia, bem como todas as alterações feitas nos dados existentes. Para ter mais informações, consulte Creating backup copies across Regiões da AWS no Guia do desenvolvedor do AWS Backup.

> **Note**
>
> O Aurora copia a quantidade mínima de dados necessária para criar uma cópia completa de um snapshot na região de destino. Há taxas de transferência de dados na cópia de snapshots entre regiões.

Considerações sobre grupos de parâmetros

Ao copiar um snapshot entre regiões, a cópia não inclui o grupo de parâmetros usado pelo cluster de banco de dados original. Ao restaurar um snapshot para criar um cluster de banco de dados, esse cluster de banco de dados obtém o grupo de parâmetros padrão da Região da AWS na qual

foi criado. Para dar ao novo cluster de banco de dados os mesmos parâmetros que o original, faça o seguinte:

1. Na Região da AWS de destino, crie um grupo de parâmetros de cluster de banco de dados com as mesmas configurações que o cluster de banco de dados original. É possível usar um grupo de parâmetros já existente na nova Região da AWS.
2. Depois de restaurar o snapshot na Região da AWS de destino, modifique o novo cluster de bancos de dados e adicione o grupo de parâmetros novo ou existente da etapa anterior.

Copiar um snapshot de cluster de banco de dados não criptografado usando a AWS CLI ou a API do Amazon RDS

Use os procedimentos nas seções a seguir para copiar um snapshot de cluster de banco de dados não criptografado usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

Para cancelar uma operação de cópia quando ela estiver em andamento, exclua o snapshot do cluster de banco de dados de destino identificado por `--target-db-cluster-snapshot-identifier` ou `TargetDBClusterSnapshotIdentifier` enquanto ele estiver no status copying (cópia).

Console

Para copiar um snapshot de cluster de banco de dados usando o Console de gerenciamento da AWS, consulte Copiar um snapshot de cluster de banco de dados com o Console de gerenciamento da AWS.

AWS CLI

Para copiar um snapshot de cluster de banco de dados, use o comando copy-db-cluster-snapshot da AWS CLI. Ao copiar o snapshot para outra Região da AWS, execute o comando na Região da AWS na qual o snapshot será copiado.

As seguintes opções são usadas para copiar um snapshot de cluster de banco de dados não criptografado:

- `--source-db-cluster-snapshot-identifier`: o identificador do snapshot de cluster de banco de dados a ser copiado. Ao copiar o snapshot para outra Região da AWS, esse identificador deve estar no formato ARN da Região da AWS de origem.

- `--target-db-cluster-snapshot-identifier`: o identificador da nova cópia do snapshot de cluster de banco de dados.

O código a seguir cria uma cópia do snapshot de cluster de banco de dados do `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` chamado `myclustersnapshotcopy` na Região da AWS em que o comando é executado. Quando a cópia é feita, todas as tags do snapshot original são copiadas para a cópia do snapshot.

Example

Para Linux, macOS ou Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

Para Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

API do RDS

Para copiar um snapshot de cluster de banco de dados, use a operação CopyDBClusterSnapshot da API do Amazon RDS. Ao copiar o snapshot para outra Região da AWS, execute a ação na Região da AWS na qual o snapshot será copiado.

Os seguintes parâmetros são usados para copiar um snapshot de cluster de banco de dados não criptografado:

- `SourceDBClusterSnapshotIdentifier`: o identificador do snapshot de cluster de banco de dados a ser copiado. Ao copiar o snapshot para outra Região da AWS, esse identificador deve estar no formato ARN da Região da AWS de origem.

- `TargetDBClusterSnapshotIdentifier`: o identificador da nova cópia do snapshot de cluster de banco de dados.

O código a seguir cria uma cópia de um snapshot `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` denominado `myclustersnapshotcopy` na região Oeste dos EUA (Norte da Califórnia). Quando a cópia é feita, todas as tags do snapshot original são copiadas para a cópia do snapshot.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copiar um snapshot de cluster de banco de dados criptografado usando a AWS CLI ou a API do Amazon RDS

Use os procedimentos nas seções a seguir para copiar um snapshot de cluster de banco de dados criptografado usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

Para cancelar uma operação de cópia quando ela estiver em andamento, exclua o snapshot do cluster de banco de dados de destino identificado por `--target-db-cluster-snapshot-identifier` ou `TargetDBClusterSnapshotIdentifier` enquanto ele estiver no status copying (cópia).

Console

Para copiar um snapshot de cluster de banco de dados usando o Console de gerenciamento da AWS, consulte Copiar um snapshot de cluster de banco de dados com o Console de gerenciamento da AWS.

AWS CLI

Para copiar um snapshot de cluster de banco de dados, use o comando copy-db-cluster-snapshot da AWS CLI. Ao copiar o snapshot para outra Região da AWS, execute o comando na Região da AWS na qual o snapshot será copiado.

As seguintes opções são usadas para copiar um snapshot de cluster de banco de dados criptografado:

- `--source-db-cluster-snapshot-identifier` – O identificador do snapshot do cluster de banco de dados criptografado a ser copiado. Ao copiar o snapshot para outra Região da AWS, esse identificador deve estar no formato ARN da Região da AWS de origem.
- `--target-db-cluster-snapshot-identifier`: o identificador da nova cópia do snapshot de cluster de banco de dados criptografado.
- `--kms-key-id`: o identificador da chave do KMS da chave a ser usada para criptografar a cópia do snapshot do cluster de banco de dados.

  Opcionalmente, será possível usar essa opção se o snapshot do cluster de banco de dados estiver criptografado, se você copiar o snapshot na mesma Região da AWS e quiser especificar uma nova chave do KMS para criptografar a cópia. Caso contrário, a cópia do snapshot do cluster de banco de dados será criptografada com a mesma chave do KMS que o snapshot do cluster de banco de dados de origem.

  Use essa opção se o snapshot do cluster de banco de dados estiver criptografado e você estiver copiando o snapshot para outra Região da AWS. Nesse caso, você deve especificar uma chave do KMS para a Região da AWS de destino.

O exemplo de código a seguir copia o snapshot de cluster de banco de dados criptografado da região Oeste dos EUA (Oregon) na região Leste dos EUA (N. da Virgínia). O comando é chamado na região Leste dos EUA (N. da Virgínia).

Example

Para Linux, macOS ou Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --kms-key-id my-us-east-1-key
```

Para Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --kms-key-id my-us-east-1-key
```

O parâmetro `--source-region` é necessário quando você está copiando um snapshot de cluster de banco de dados criptografado entre as regiões GovCloud (Leste dos EUA) da AWS e GovCloud (Oeste dos EUA) da AWS. Em `--source-region`, especifique a Região da AWS da instância de banco de dados de origem. A Região da AWS especificada em `source-db-cluster-snapshot-identifier` deve corresponder à Região da AWS especificada como a `--source-region`.

Se a `--source-region` não estiver especificada, especifique um valor de `--pre-signed-url`. Uma presigned URL é um URL que contém uma solicitação assinada do Signature Versão 4 para o comando `copy-db-cluster-snapshot` chamado na Região da AWS de origem. Para saber mais sobre a opção `pre-signed-url`, consulte copy-db-cluster-snapshot na Referência de comandos da AWS CLI.

API do RDS

Para copiar um snapshot de cluster de banco de dados, use a operação CopyDBClusterSnapshot da API do Amazon RDS. Ao copiar o snapshot para outra Região da AWS, execute a ação na Região da AWS na qual o snapshot será copiado.

Os seguintes parâmetros são usados para copiar um snapshot de cluster de banco de dados criptografado:

- `SourceDBClusterSnapshotIdentifier` – O identificador do snapshot do cluster de banco de dados criptografado a ser copiado. Ao copiar o snapshot para outra Região da AWS, esse identificador deve estar no formato ARN da Região da AWS de origem.
- `TargetDBClusterSnapshotIdentifier`: o identificador da nova cópia do snapshot de cluster de banco de dados criptografado.

- `KmsKeyId`: o identificador da chave do KMS da chave a ser usada para criptografar a cópia do snapshot do cluster de banco de dados.

  Também é possível usar esse parâmetro se o snapshot do cluster de banco de dados estiver criptografado, se você copiar o snapshot na mesma Região da AWS e especificar uma nova chave do KMS a ser usada para criptografar a cópia. Caso contrário, a cópia do snapshot do cluster de banco de dados será criptografada com a mesma chave do KMS que o snapshot do cluster de banco de dados de origem.

  Use esse parâmetro se o snapshot do cluster de banco de dados estiver criptografado e você estiver copiando o snapshot para outra Região da AWS. Nesse caso, você deve especificar uma chave do KMS para a Região da AWS de destino.
- `PreSignedUrl`: ao copiar o snapshot para outra Região da AWS, especifique o parâmetro `PreSignedUrl`. O valor de `PreSignedUrl` deve ser um URL que contenha uma solicitação assinada do Signature versão 4 para que a ação `CopyDBClusterSnapshot` seja chamada na Região da AWS de origem da qual o snapshot do cluster de banco de dados é copiado. Para saber mais sobre o uso de URLs pré-assinados, consulte CopyDBClusterSnapshot.

O exemplo de código a seguir copia o snapshot de cluster de banco de dados criptografado da região Oeste dos EUA (Oregon) na região Leste dos EUA (N. da Virgínia). A ação é chamada na região Leste dos EUA (N. da Virgínia).

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards
%25253Aus-west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-
snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
```

```
        %2526X-Amz-Date%253D20161117T215409Z
        %2526X-Amz-Expires%253D3600
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

O parâmetro `PreSignedUrl` é necessário quando você está copiando um snapshot de cluster de banco de dados criptografado entre as regiões GovCloud (Leste dos EUA) da AWS e as regiões GovCloud (Oeste dos EUA) da AWS. O valor de `PreSignedUrl` deve ser um URL que contenha uma solicitação assinada do Signature versão 4 para que a operação `CopyDBClusterSnapshot` seja chamada na Região da AWS de origem da qual o snapshot do cluster de banco de dados é copiado. Para saber mais sobre como usar um URL pré-assinado, consulte CopyDBClusterSnapshot na Referência da API do Amazon RDS.

Para gerar um URL pré-assinado automaticamente, em vez de manualmente, use o comando copy-db-cluster-snapshot da AWS CLI com a opção `--source-region`.

Copiar um snapshot de cluster de banco de dados entre contas

Você pode habilitar outras contas da AWS para copiar os snapshots de cluster de banco de dados que você especifica usando as ações `ModifyDBClusterSnapshotAttribute` e `CopyDBClusterSnapshot` da API do Amazon RDS. Só é possível copiar snapshots de clusters de banco de dados entre contas na mesma Região da AWS. O processo de cópia entre contas funciona da seguinte forma, em que a Conta A disponibiliza o snapshot para cópia e a Conta B o está copiando.

1. Usando a Conta A, chame `ModifyDBClusterSnapshotAttribute`, especificando **`restore`** para o parâmetro `AttributeName` e o ID da Conta B para o parâmetro `ValuesToAdd`.

2. (Se o snapshot estiver criptografado) Usando a Conta A, atualize a política de chave para a chave do KMS, primeiro adicionando o ARN da Conta B como `Principal` e, em seguida, permita a ação `kms:CreateGrant`.
3. (Se o snapshot estiver criptografado) Usando a conta B, escolha ou crie um usuário e anexe uma política do IAM a esse usuário que o permita copiar um snapshot de cluster de banco de dados criptografado usando sua chave do KMS.
4. Usando a Conta B, chame `CopyDBClusterSnapshot` e use o parâmetro `SourceDBClusterSnapshotIdentifier` para especificar o ARN do snapshot de cluster de banco de dados a ser copiado, que deve incluir o ID da Conta A.

Para listar todas as contas da AWS que têm permissão para restaurar um snapshot de cluster de banco de dados, use a operação de API DescribeDBSnapshotAttributes ou DescribeDBClusterSnapshotAttributes.

Para remover a permissão de compartilhamento de uma conta da AWS, use a ação `ModifyDBSnapshotAttribute` ou `ModifyDBClusterSnapshotAttribute` com `AttributeName` definido como `restore` e o ID da conta para remoção no parâmetro `ValuesToRemove`.

Copiar um snapshot de cluster de banco de dados não criptografado para outra conta

Use o procedimento a seguir para copiar um snapshot de cluster de banco de dados não criptografado em outra conta na mesma Região da AWS.

1. Na conta de origem do snapshot de cluster de banco de dados, chame `ModifyDBClusterSnapshotAttribute`, especificando **`restore`** para o parâmetro `AttributeName`, e o ID da conta de destino para o parâmetro `ValuesToAdd`.

   Executar o exemplo a seguir usando a conta `987654321` permite que dois identificadores de conta da AWS `123451234512` e `123456789012`, restaurem o snapshot do cluster de banco de dados denominado `manual-snapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBClusterSnapshotAttribute
    &AttributeName=restore
    &DBClusterSnapshotIdentifier=manual-snapshot1
    &SignatureMethod=HmacSHA256&SignatureVersion=4
    &ValuesToAdd.member.1=123451234512
    &ValuesToAdd.member.2=123456789012
   ```

```
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Na conta de destino, chame `CopyDBClusterSnapshot` e use o parâmetro `SourceDBClusterSnapshotIdentifier` para especificar o ARN do snapshot de cluster de banco de dados a ser copiado, que deve incluir o ID da conta de origem.

   Executar o exemplo a seguir usando a conta `123451234512` copia o snapshot de cluster de banco de dados `aurora-cluster1-snapshot-20130805` da conta `987654321` e cria um snapshot de cluster de banco de dados chamado `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copiar um snapshot de cluster de banco de dados criptografado para outra conta

Use o procedimento a seguir para copiar um snapshot de cluster de banco de dados criptografado em outra conta na mesma Região da AWS.

1. Na conta de origem do snapshot de cluster de banco de dados, chame `ModifyDBClusterSnapshotAttribute`, especificando **`restore`** para o parâmetro `AttributeName`, e o ID da conta de destino para o parâmetro `ValuesToAdd`.

Executar o exemplo a seguir usando a conta `987654321` permite que dois identificadores de conta da AWS `123451234512` e `123456789012`, restaurem o snapshot do cluster de banco de dados denominado `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Na conta de origem do snapshot de cluster de banco de dados, crie uma chave do KMS personalizada na mesma Região da AWS que o snapshot de cluster de banco de dados criptografado. Ao criar a chave gerenciada pelo cliente, conceda acesso a ela para a Conta da AWS planejada. Para ter mais informações, consulte Criar uma chave gerenciada pelo cliente e conceder acesso a ela.
3. Copie e compartilhe o snapshot com a Conta da AWS planejada. Para ter mais informações, consulte Copiar e compartilhar o snapshot da conta de origem.
4. Na conta de destino, chame `CopyDBClusterSnapshot` e use o parâmetro `SourceDBClusterSnapshotIdentifier` para especificar o ARN do snapshot de cluster de banco de dados a ser copiado, que deve incluir o ID da conta de origem.

   Executar o exemplo a seguir usando a conta `123451234512` copia o snapshot de cluster de banco de dados `aurora-cluster1-snapshot-20130805` da conta `987654321` e cria um snapshot de cluster de banco de dados chamado `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
```

```
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Compartilhar um snapshot do cluster de banco de dados

Com o Amazon RDS, você pode compartilhar um snapshot manual do cluster de banco de dados das seguintes formas:

- O compartilhamento de um snapshot manual de cluster de banco de dados, seja criptografado ou não, permite que as contas da AWS autorizadas copiem o snapshot.
- Compartilhar um snapshot de cluster de banco de dados manual, criptografado ou não, permite às contas da AWS autorizadas restaurarem diretamente um cluster de banco de dados a partir do snapshot em vez de fazer uma cópia dele e restaurar a partir daí.

> **Note**
>
> Para compartilhar um snapshot automatizado do cluster de banco de dados, crie um snapshot manual do cluster de banco de dados copiando o snapshot automatizado e, em seguida, compartilhe essa cópia. Esse processo também se aplica aos recursos gerados pelo AWS Backup.

Para obter mais informações sobre a cópia de um snapshot, consulte Cópia de snapshot de cluster de banco de dados. Para obter mais informações sobre como restaurar uma instância de banco de dados de um snapshot de cluster de banco de dados, consulte Restauração de um snapshot de um cluster de banco de dados.

Para obter mais informações sobre a restauração de um cluster de banco de dados a partir de um snapshot de cluster de banco de dados, consulte Visão geral do backup e da restauração de um cluster de banco de dados do Aurora.

É possível compartilhar um snapshot manual com até 20 outras Contas da AWS.

Ao compartilhar snapshots manuais com outras Contas da AWS, a seguinte limitação é aplicada:

- Ao restaurar um cluster de banco de dados de um snapshot compartilhado usando a AWS Command Line Interface (AWS CLI) ou a API do Amazon RDS, especifique o nome do recurso da Amazon (ARN) do snapshot compartilhado como o identificador do snapshot.

Aprenda a compartilhar snapshots, snapshots públicos e snapshots criptografados nas seções a seguir. Você também pode aprender a interromper o compartilhamento de snapshots.

Tópicos

- Compartilhar um snapshot
- Compartilhamento de snapshots públicos
- Compartilhamento de snapshots criptografados
- Interromper o compartilhamento de snapshots

Compartilhar um snapshot

Você pode compartilhar um snapshot de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Usando o console do Amazon RDS, é possível compartilhar um snapshot manual de cluster de banco de dados com até vinte Contas da AWS. Você também pode usar o console para interromper o compartilhamento de um snapshot manual com uma ou mais contas.

Para compartilhar um snapshot manual do cluster de banco de dados usando o console do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.
3. Selecione o snapshot manual que você deseja compartilhar.
4. Em Actions (Ações), selecione Share Snapshot (Compartilhar snapshot).
5. Escolha uma das seguintes opções para DB snapshot visibility (Visibilidade do snapshot de banco de dados).

    - Se a origem não estiver criptografada, selecione Público para permitir que todas as Contas da AWS restaurem um cluster de banco de dados de um snapshot manual de cluster de banco de dados, ou escolha Privado para permitir que apenas as Contas da AWS especificadas restaurem um cluster de banco de dados do snapshot manual de cluster de banco de dados.

    > ⚠ Warning
    >
    > Se você definir Visibilidade do snapshot do banco de dados como Público, todas as Contas da AWS poderão restaurar um cluster de banco de dados de um snapshot

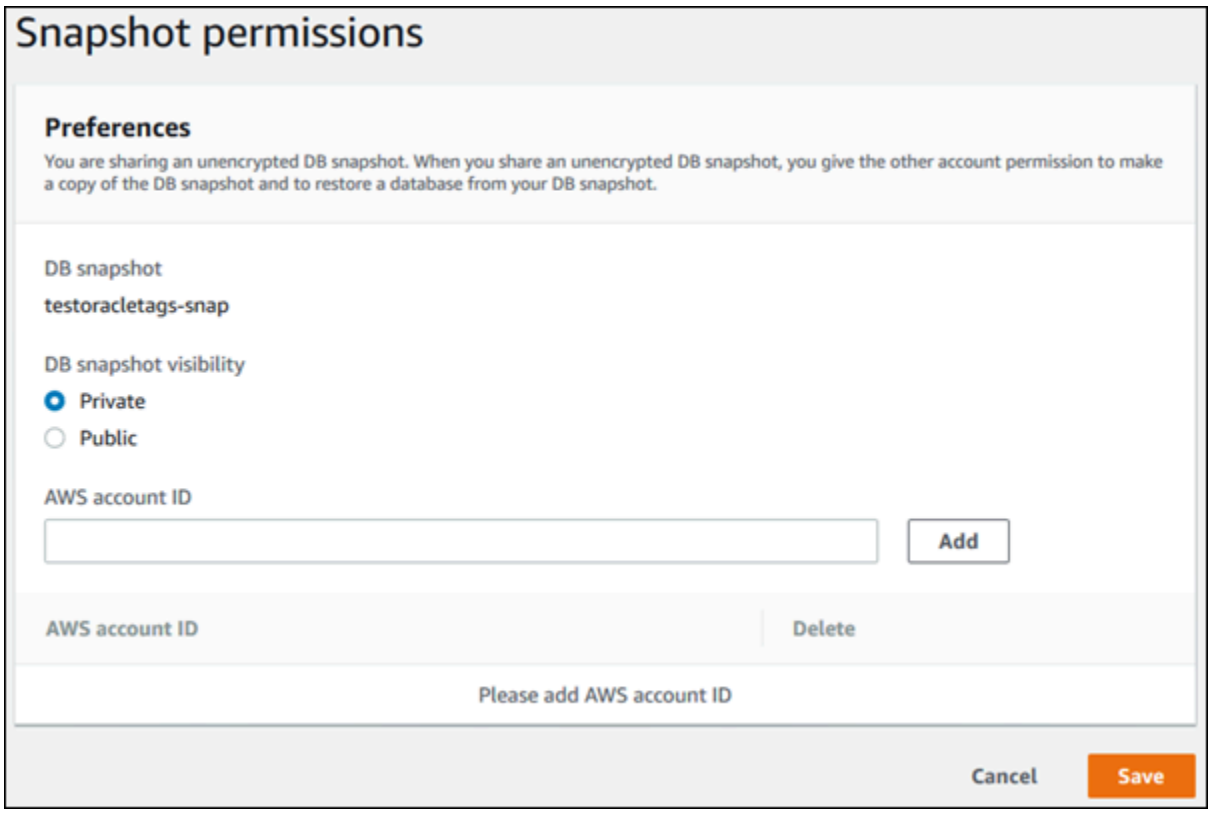
manual de cluster de banco de dados e ter acesso aos dados. Não compartilhe nenhum snapshot de cluster de banco de dados manual que contenha informações privadas, como Public (Público).
Para obter mais informações, consulte Compartilhamento de snapshots públicos.

- Se a origem estiver criptografada, DB snapshot visibility (Visibilidade do snapshot de banco de dados) será definida como Private (Privada) porque os snapshots criptografados não podem ser compartilhados como públicos.

> **Note**
>
> Snapshots criptografados com a AWS KMS key padrão não podem ser compartilhados. Para ter informações sobre como contornar esse problema, consulte Compartilhamento de snapshots criptografados.

6. Para ID da conta da AWS, digite o identificador da Conta da AWS à qual você deseja conceder permissão para restaurar um cluster de banco de dados do snapshot manual e, depois, selecione Adicionar. Repita a operação para incluir outros identificadores da Conta da AWS, até vinte Contas da AWS.

   Se você errar ao adicionar o identificador da Conta da AWS à lista de contas permitidas, saiba que é possível excluí-lo da lista escolhendo Excluir à direita do identificador incorreto da Conta da AWS.



7. Depois de adicionar os identificadores de todas as Contas da AWS às quais você deseja conceder a permissão para restaurar o snapshot manual, selecione Salvar para salvar as alterações.

AWS CLI

Para compartilhar um snapshot de cluster de banco de dados, utilize o comando `aws rds modify-db-cluster-snapshot-attribute`. Use o parâmetro `--values-to-add` para adicionar uma lista dos IDs das Contas da AWS autorizadas a restaurar o snapshot manual.

Example compartilhamento de um snapshot com uma única conta

O exemplo a seguir permite que o identificador da Conta da AWS, `123456789012`, restaure o snapshot de cluster de banco de dados chamado `cluster-3-snapshot`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier cluster-3-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

Para Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier cluster-3-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example compartilhamento de um snapshot com múltiplas contas

O exemplo a seguir permite que dois identificadores de Conta da AWS, `111122223333` e `444455556666`, restaurem o snapshot de cluster de banco de dados chamado `manual-cluster-snapshot1`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Para Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Ao usar o prompt de comando do Windows, você deve fazer o escape das aspas duplas (") no código JSON, prefixando-as com uma barra invertida (\).

Para listar as Contas da AWS habilitadas para restaurar um snapshot, utilize o comando `describe-db-cluster-snapshot-attributes` da AWS CLI.

API do RDS

Também é possível compartilhar um snapshot manual de cluster de banco de dados com outras Contas da AWS usando a API do Amazon RDS. Para fazer isso, chame a operação `ModifyDBClusterSnapshotAttribute`. Especifique `restore` para `AttributeName` e use o

parâmetro `ValuesToAdd` para adicionar uma lista dos IDs das Contas da AWS que têm autorização para restaurar o snapshot manual.

Para tornar um snapshot manual público e restaurável por todas as Contas da AWS, use o valor `all`. No entanto, tome cuidado para não adicionar o valor `all` a nenhum snapshot manual com informações privadas que não deseja disponibilizar para todas as Contas da AWS. Além disso, não especifique `all` para snapshots criptografados, pois não é possível torná-los públicos.

Para listar todas as Contas da AWS que têm permissão para restaurar um snapshot, utilize a operação `DescribeDBClusterSnapshotAttributes` da API.

Compartilhamento de snapshots públicos

É possível compartilhar um snapshot manual não criptografado como público, disponibilizando-o para todas as Contas da AWS. Ao compartilhar um snapshot como público, verifique se suas informações privadas não estão incluídas nos snapshots públicos.

Quando um snapshot é compartilhado publicamente, ele permite que todas as Contas da AWS copiem o snapshot e criem clusters de banco de dados dele.

Você não é cobrado pelo armazenamento de backup de snapshots públicos pertencentes a outras contas. Você é cobrado apenas pelos snapshots pertencentes a você.

Se você copiar um snapshot público, será o proprietário da cópia. Você será cobrado pelo armazenamento de backup da sua cópia do snapshot. Se você criar um cluster de banco de dados de um snapshot público, esse cluster de banco de dados será cobrado. Para obter informações sobre a definição de preço do Amazon Aurora, consulte a página Definição de preço do Aurora.

Você pode excluir somente os snapshots públicos pertencentes a você. Para excluir um snapshot compartilhado ou público, você deve fazer login na Conta da AWS à qual pertence esse snapshot.

Visualizar snapshots públicos pertencentes a outras Contas da AWS

É possível visualizar snapshots públicos pertencentes a outras contas em uma Região da AWS específica na guia Público na página Snapshots do console do Amazon RDS. Seus snapshots (aqueles pertencentes à sua conta) não aparecem nesta guia.

Para visualizar snapshots públicos

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.

3. Selecione a guia Public (Público).

   Os snapshots públicos são exibidos. Você pode ver qual conta possui um snapshot público na coluna Owner (Proprietário).

   > **Note**
   >
   > Talvez seja necessário modificar as preferências da página selecionando o ícone de engrenagem no canto superior direito da lista Public snapshots (Instantâneos públicos) para ver esta coluna.

Visualização dos seus próprios snapshots públicos

É possível usar o seguinte comando da AWS CLI (somente Unix) para visualizar os snapshots públicos pertencentes à Conta da AWS em uma Região da AWS específica.

```
aws rds describe-db-cluster-snapshots --snapshot-type public --include-public |
 grep account_number
```

A saída retornada será semelhante ao exemplo a seguir se você tiver instantâneos públicos.

```
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot1",
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot2",
```

Compartilhamento de snapshots públicos de versões obsoletas do mecanismo de banco de dados

Não é permitido restaurar nem copiar snapshots públicos de versões obsoletas do mecanismo de banco de dados. Para disponibilizar o snapshot público não compatível existente para restauração ou cópia, execute as seguintes etapas:

1. Marque o snapshot como privado.
2. Restaure o snapshot.
3. Faça upgrade do cluster de banco de dados restaurado para uma versão de mecanismo compatível.
4. Crie um snapshot.

5. Compartilhe novamente o snapshot publicamente.

Compartilhamento de snapshots criptografados

Você pode compartilhar snapshots do cluster de banco de dados que foram criptografados "em repouso" usando o algoritmo de criptografia AES-256, conforme descrito em Criptografar recursos do Amazon Aurora.

As seguintes restrições se aplicam ao compartilhamento de snapshots criptografados:

- Não é possível compartilhar snapshots criptografados como públicos.
- Não é possível compartilhar um snapshot criptografado usando a chave do KMS padrão da Conta da AWS que compartilhou o snapshot.

  Para ter mais informações sobre o gerenciamento de chaves AWS KMS para o Amazon RDS, consulte AWS KMS keyGerenciamento de.

Para contornar o problema da chave do KMS padrão, realize as seguintes tarefas:

1. Criar uma chave gerenciada pelo cliente e conceder acesso a ela.
2. Copiar e compartilhar o snapshot da conta de origem.
3. Copiar o snapshot compartilhado na conta de destino.

Criar uma chave gerenciada pelo cliente e conceder acesso a ela

Primeiro, você deve criar uma chave do KMS personalizada na mesma Região da AWS do snapshot do cluster de banco de dados criptografado. Ao criar a chave gerenciada pelo cliente, conceda acesso a ela a outra Conta da AWS.

Como criar uma chave gerenciada pelo cliente e conceder acesso a ela

1. Faça login no Console de gerenciamento da AWS pela Conta da AWS de origem.
2. Abra o console do AWS KMS em https://console.aws.amazon.com/kms.
3. Para alterar a Região da AWS, use o seletor de regiões no canto superior direito da página.
4. No painel de navegação, escolha Customer managed keys (Chaves gerenciadas pelo cliente).
5. Escolha Create key (Criar chave).

6. Na página Configurar chave:
    a. Em Tipo de chave, selecione Simétrico.
    b. Em Uso da chave, selecione Criptografar e descriptografar.
    c. Expanda Advanced options (Opções avançadas).
    d. Em Origem do material de chaves, selecione KMS.
    e. Em Regionalidade, selecione Chave de região única.
    f. Escolha Próximo.
7. Na página Adicionar rótulos:
    a. Para Alias, insira um nome de exibição para a chave do KMS, por exemplo, **`share-snapshot`**.
    b. (Opcional) Insira uma descrição para a chave do KMS.
    c. (Opcional) Adicione tags à chave do KMS.
    d. Escolha Próximo.
8. Na página Definir permissões administrativas da chave, escolha Próximo.
9. Na página Definir permissões de uso da chave:
    a. Em Outras Contas da AWS, selecione Adicionar outra Conta da AWS.
    b. Insira o ID da Conta da AWS à qual você deseja conceder acesso.

        É possível conceder acesso a várias Contas da AWS.
    c. Escolha Próximo.
10. Revise a chave do KMS e escolha Concluir.

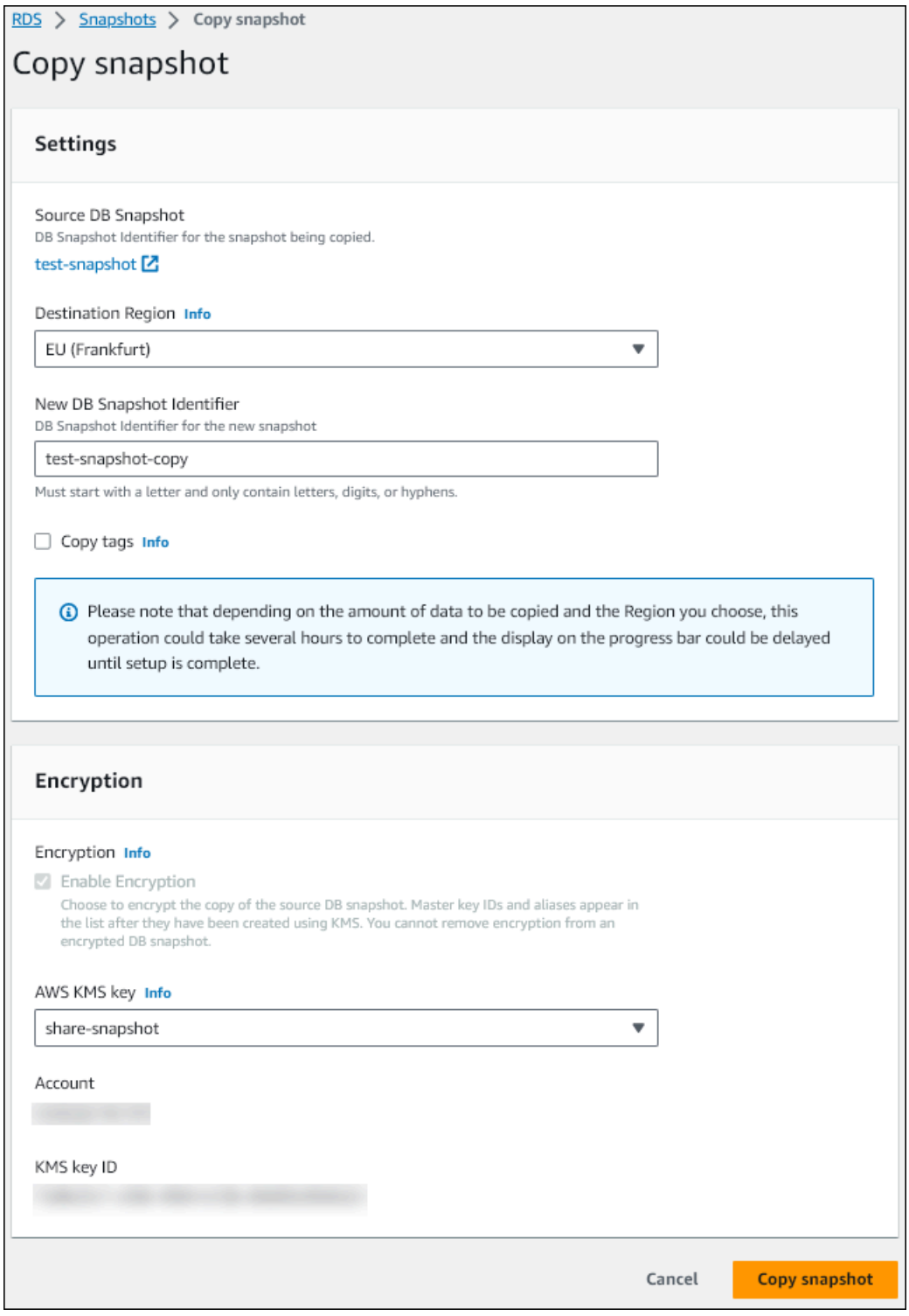
Copiar e compartilhar o snapshot da conta de origem

Depois, você deve copiar o snapshot do cluster de banco de dados de origem para um novo snapshot usando a chave gerenciada pelo cliente. Depois, você vai compartilhá-lo com a Conta da AWS de destino.

Como copiar e compartilhar o snapshot

1. Faça login no Console de gerenciamento da AWS pela Conta da AWS de origem.
2. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

3. No painel de navegação, escolha Snapshots.
4. Selecione o snapshot de cluster de banco de dados a ser copiado.
5. Para Actions (Ações), escolha Copy Snapshot (Copiar snapshot).
6. Na página Copiar snapshot:
    a. Em Região de destino, selecione a Região da AWS onde você criou a chave gerenciada pelo cliente no procedimento anterior.
    b. Digite o nome da cópia do snapshot de cluster de banco de dados em New DB Snapshot Identifier (Novo identificador de DB snapshot).
    c. Para AWS KMS key, selecione a chave gerenciada pelo cliente que você criou.



d. Escolha Copy snapshot (Copiar snapshot).

7. Quando a cópia do snapshot estiver disponível, selecione-a.
8. Em Actions (Ações), selecione Share Snapshot (Compartilhar snapshot).
9. Na página Permissões de snapshot:

a. Insira o ID da Conta da AWS com a qual você está compartilhando a cópia do snapshot e selecione Adicionar.
b. Escolha Salvar.

O snapshot é compartilhado.

Copiar o snapshot compartilhado na conta de destino

Agora você pode copiar o snapshot compartilhado na Conta da AWS de destino.

Como copiar o snapshot compartilhado

1. Faça login no Console de gerenciamento da AWS pela Conta da AWS de destino.
2. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
3. No painel de navegação, escolha Snapshots.
4. Selecione a guia Compartilhado comigo.
5. Selecione o snapshot compartilhado.
6. Para Actions (Ações), escolha Copy Snapshot (Copiar snapshot).
7. Escolha as configurações para copiar o snapshot como no procedimento anterior, mas use uma AWS KMS key que pertença à conta de destino.

   Escolha Copy snapshot (Copiar snapshot).

Interromper o compartilhamento de snapshots

Para parar de compartilhar um snapshot de cluster de banco de dados, é necessário remover a permissão da Conta da AWS de destino.

Console

Como interromper o compartilhamento de um snapshot manual de cluster de banco de dados com uma Conta da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.

3. Selecione o snapshot manual que você deseja interromper o compartilhamento.
4. Selecione Actions (Ações) e, depois, Share Snapshot (Compartilhar snapshot).
5. Para remover a permissão de uma Conta da AWS, selecione Excluir para o identificador da conta da AWS na lista de contas autorizadas.
6. Escolha Salvar para salvar as alterações.

CLI

Para remover um identificador Conta da AWS na lista, use o parâmetro `--values-to-remove`.

Example interrupção do compartilhamento do snapshot

O exemplo a seguir impede que o ID 444455556666 da Conta da AWS restaure o snapshot.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Para Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

API do RDS

Para remover a permissão de compartilhamento de uma Conta da AWS, use a operação `ModifyDBClusterSnapshotAttribute` com `AttributeName` definido como `restore` e o parâmetro `ValuesToRemove`. Para marcar um snapshot manual como privado, remova o valor `all` na lista de valores do atributo `restore`.

Exportar dados do cluster de banco de dados para o Amazon S3

É possível exportar dados de um cluster de banco de dados do Amazon Aurora para um bucket do Amazon S3. O processo de exportação é executado em segundo plano e não afeta a performance do cluster de banco de dados ativo.

Por padrão, todos os dados no cluster de banco de dados são exportados. No entanto, você pode optar por exportar conjuntos específicos de bancos de dados, esquemas ou tabelas.

O Amazon Aurora clona o cluster de banco de dados, extrai dados do clone e armazena-os em um bucket do Amazon S3. Os dados são armazenados em um formato Apache Parquet que é compactado e consistente. Em geral, os arquivos Parquet individuais têm cerca de 1 a 10 MB de tamanho.

A performance mais rápida que você pode obter com a exportação de dados de snapshot para o Aurora MySQL versão 2 e versão 3 não se aplica à exportação de dados de cluster de banco de dados. Para ter mais informações, consulte Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.

Você recebe cobrança pela exportação de todo o cluster de banco de dados, independentemente de exportar todos os dados ou parte deles. Para ter mais informações, consulte a página Definição de preço do Amazon Aurora.

Depois que os dados são exportados, você pode analisar os dados exportados diretamente por meio de ferramentas, como Amazon Athena ou Amazon Redshift Spectrum. Para ter mais informações sobre como usar o Athena para ler os dados do Parquet, consulte Parquet SerDe no Guia do usuário do Amazon Athena. Para ter mais informações sobre como usar o Redshift Spectrum para ler os dados do Parquet, consulte COPY de formatos de dados colunares no Guia do desenvolvedor de banco de dados do Amazon Redshift.

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e regiões para a exportação de dados de cluster de banco de dados para o S3, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de cluster para o Amazon S3.

Use o seguinte processo para exportar dados de cluster de banco de dados para um bucket do Amazon S3. Para obter mais detalhes, consulte as seções a seguir.

Visão geral da exportação de dados do cluster de banco de dados

1. Identifique o cluster de banco de dados cujos dados você deseja exportar.
2. Configure o acesso ao bucket do Amazon S3.

   Um bucket é um contêiner de objetos ou arquivos do Amazon S3. Para fornecer informações para acesso a um bucket, execute as seguintes etapas:

   a. Identifique o bucket do S3 para os dados do cluster de banco de dados devem ser exportados. O bucket do S3 deve estar na mesma região da AWS que o cluster de banco de dados. Para ter mais informações, consulte Identificar o bucket do Amazon S3 para exportar.

   b. Crie um perfil do AWS Identity and Access Management (IAM) que conceda à tarefa de exportação de cluster de banco de dados acesso ao bucket do S3. Para ter mais informações, consulte Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM.
3. Crie uma criptografia simétrica do AWS KMS key para a criptografia do lado do servidor. A chave do KMS é usada pela tarefa de exportação de cluster para configurar a criptografia do lado do servidor do AWS KMS ao gravar dados de exportação no S3.

   A política de chave do KMS deve incluir as permissões `kms:CreateGrant` e `kms:DescribeKey`. Para ter mais informações sobre o uso de chaves do KMS no Amazon Aurora, consulte AWS KMS keyGerenciamento de.

   Além disso, se você tiver uma instrução de negação na política de chaves do KMS, exclua explicitamente a entidade principal de serviço da AWS `export.rds.amazonaws.com`.

   Você pode utilizar uma chave do KMS na sua conta do AWS ou pode utilizar uma chave do KMS entre contas. Para ter mais informações, consulte Utilizar uma conta cruzada AWS KMS key.
4. Exporte o cluster de banco de dados para o Amazon S3 usando o console ou o comando `start-export-task` da CLI. Para ter mais informações, consulte Criar tarefas de exportação do cluster de banco de dados.
5. Para acessar os seus dados exportados no bucket do Amazon S3, consulte Como carregar, baixar e gerenciar objetos no Guia do usuário do Amazon Simple Storage Service.

Aprenda a configurar, exportar, monitorar, cancelar e solucionar problemas em tarefas de exportação de cluster de banco de dados nas seções a seguir.

Tópicos

- Considerações sobre exportações do cluster de banco de dados
- Configurar o acesso a um bucket do Amazon S3
- Criar tarefas de exportação do cluster de banco de dados
- Monitorar tarefas de exportação do cluster de banco de dados
- Cancelar uma tarefa de exportação de cluster de banco de dados
- Solucionar problemas do cluster de banco de dados

Considerações sobre exportações do cluster de banco de dados

Use as seções a seguir para saber sobre as limitações, as convenções de nomenclatura de arquivos, além da conversão e do armazenamento de dados ao exportar dados do cluster de banco de dados para o Amazon S3.

Tópicos

- Limitações
- Convenção de nomenclatura de arquivos
- Formato de conversão e armazenamento de dados

Limitações

A exportação de dados de cluster de banco de dados para o Amazon S3 apresenta as seguintes limitações:

- Não é possível executar várias tarefas de exportação para o mesmo cluster de banco de dados simultaneamente. Isso se aplica a exportações totais e parciais.
- Você pode ter até cinco tarefas de exportação de snapshot de banco de dados simultâneas em andamento por Conta da AWS.
- Os clusters de banco de dados do Aurora Serverless v1 não são compatíveis com exportações para o S3.
- O Aurora MySQL e o Aurora PostgreSQL são compatíveis com exportações para o S3 somente para o modo de mecanismo provisionado.
- As exportações para o S3 não são compatíveis com prefixos do S3 contendo dois pontos (:).

- Os seguintes caracteres no caminho do arquivo do S3 são convertidos em sublinhados (_) durante a exportação:

```
\ ` " (space)
```

- Se um banco de dados, esquema ou tabela tiver caracteres em seu nome diferentes dos a seguir, a exportação parcial não será aceita. No entanto, você pode exportar o cluster de banco de dados inteiro.
  - Letras latinas (A–Z)
  - Dígitos (0–9)
  - Símbolo do dólar ($)
  - Sublinhado (_)
- Espaços ( ) e alguns caracteres não são compatíveis em nomes de colunas de tabelas de banco de dados. As tabelas com os seguintes caracteres em nomes de colunas são ignoradas durante a exportação:

```
, ; { } ( ) \n \t = (space)
```

- As tabelas com barras (/) em seus nomes são ignoradas durante a exportação.
- As tabelas temporárias e não registradas do Aurora PostgreSQL são ignoradas durante a exportação.
- Se os dados contiverem um objeto grande, como um BLOB ou um CLOB, com tamanho próximo ou superior a 500 MB, ocorrerá uma falha na exportação.
- Se uma tabela contiver uma linha grande próxima ou maior que 2 GB, a tabela será ignorada durante a exportação.
- Em relação a exportações parciais, a lista `ExportOnly` tem um tamanho máximo de 200 KB.
- É altamente recomendável que você use um nome exclusivo para cada tarefa de exportação. Se você não usar um nome de tarefa exclusivo, poderá receber a seguinte mensagem de erro:

  ExportTaskAlreadyExistsFault: An error occurred (ExportTaskAlreadyExists) when calling the StartExportTask operation: The export task with the ID *xxxxx* already exists [ExportTaskAlreadyExistsFault: ocorreu um erro (ExportTaskAlreadyExists) ao chamar a operação StartExportTask: a tarefa de exportação com o ID xxxxx já existe].
- Como algumas tabelas podem ser ignoradas, recomendamos que você verifique a contagem de linhas e tabelas nos dados após a exportação.

Convenção de nomenclatura de arquivos

Os dados exportados para tabelas específicas são armazenados no formato *base_prefix*/*files*, onde o prefixo base é o seguinte:

```
export_identifier/database_name/schema_name.table_name/
```

Por exemplo:

```
export-1234567890123-459/rdststcluster/mycluster.DataInsert_7ADB5D19965123A2/
```

Os arquivos de saída usam a seguinte convenção de nomenclatura, em que *partition_index* é alfanumérico:

```
partition_index/part-00000-random_uuid.format-based_extension
```

Por exemplo:

```
1/part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
    a/part-00000-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
```

A convenção de nomenclatura de arquivos está sujeita a alterações. Portanto, ao ler as tabelas de destino, recomendamos que você leia tudo dentro do prefixo base para a tabela.

Formato de conversão e armazenamento de dados

Ao exportar um cluster de banco de dados para um bucket do Amazon S3, o Amazon Aurora converte, exporta e armazena os dados no formato Parquet. Para ter mais informações, consulte Conversão de dados ao exportar para um bucket do Amazon S3.

Configurar o acesso a um bucket do Amazon S3

Você identifica o bucket do Amazon S3, depois fornece à tarefa de exportação do cluster de banco de dados permissão para acessá-lo.

Tópicos

- Identificar o bucket do Amazon S3 para exportar
- Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM
- Utilizar um bucket do Amazon S3 entre contas

Identificar o bucket do Amazon S3 para exportar

Identifique o bucket do Amazon S3 para o qual exportar os dados de cluster de banco de dados. Use um bucket do S3 existente ou crie um novo bucket do S3.

> **Note**
>
> O bucket do S3 deve estar na mesma região da AWS que o cluster de banco de dados.

Para ter mais informações sobre como trabalhar com buckets do Amazon S3, consulte o seguinte no Guia do usuário do Amazon Simple Storage Service:

- Como visualizar as propriedades de um bucket do S?
- Como habilitar a criptografia padrão em um bucket do Amazon S?
- Como criar um bucket do S?

Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM

Antes de exportar dados de cluster de banco de dados para o Amazon S3, forneça a permissão de acesso de gravação ao bucket do Amazon S3 às tarefas de exportação.

Para conceder essa permissão, crie uma política do IAM que forneça acesso ao bucket e, depois, crie um perfil do IAM e anexe a política ao perfil. Posteriormente, você pode atribuir o perfil do IAM à tarefa de exportação do cluster de banco de dados.

> **Important**
>
> Se você planeja usar o Console de gerenciamento da AWS para exportar seu cluster de banco de dados, poderá optar por criar a política do IAM e o perfil automaticamente ao exportar o cluster de banco de dados. Para instruções, consulte Criar tarefas de exportação do cluster de banco de dados.

Como fornecer às tarefas acesso ao Amazon S3

1. Crie uma política do IAM. Essa política fornece as permissões de bucket e objeto que permitem que sua tarefa de exportação de cluster de banco de dados acesse o Amazon S3.

   Na política, inclua as ações necessárias a seguir para permitir a transferência de arquivos do Amazon Aurora para um bucket do S3:

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   Na política, inclua os recursos a seguir para identificar o bucket do S3 e os objetos no bucket. A lista de recursos a seguir mostra o formato do nome de recurso da Amazon (ARN) para acessar o Amazon S3.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Para ter mais informações sobre como criar uma política do IAM para o Amazon Aurora, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM. Consulte também Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

   O comando da AWS CLI a seguir cria uma política do IAM denominada `ExportPolicy` com essas opções. Ele concede acesso a um bucket denominado *`amzn-s3-demo-bucket`*.

   > **Note**
   >
   > Depois de criar a política, anote o ARN da política. O ARN será necessário para uma etapa posterior, quando você anexar a política a um perfil do IAM.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crie um perfil do IAM para que o Aurora possa assumir esse perfil do IAM em seu nome a fim de acessar os buckets do Amazon S3. Para ter mais informações, consulte Criar um perfil para delegar permissões a um usuário do IAM no Guia do usuário do IAM.

   O exemplo a seguir mostra como usar o comando da AWS CLI para criar uma função chamada `rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Anexe a política do IAM que você criou ao perfil do IAM que você criou.

O seguinte comando da AWS CLI anexa a política criada anteriormente à função chamada `rds-s3-export-role`. Substitua *`your-policy-arn`* pelo ARN da política que você anotou em uma etapa anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-export-role
```

Utilizar um bucket do Amazon S3 entre contas

É possível utilizar buckets do S3 entre contas da AWS. Para obter mais informações, consulte Utilizar um bucket do Amazon S3 entre contas.

Criar tarefas de exportação do cluster de banco de dados

Crie tarefas de exportação para exportar dados do cluster de banco de dados do Aurora para um bucket do Amazon S3. Você pode ter até cinco tarefas de exportação de cluster de banco de dados simultâneas em andamento por Conta da AWS.

> **Note**
>
> A exportação dos dados do cluster de banco de dados pode levar um tempo, dependendo do tipo e do tamanho do banco de dados. A tarefa de exportação primeiro clona e escala todo o banco de dados antes de extrair os dados para o Amazon S3. O andamento da tarefa durante essa fase é exibido como Starting (Iniciando). Quando a tarefa muda para a exportação de dados para o S3, o andamento é exibido como In progress (Em andamento). O tempo necessário para que a exportação seja concluída depende dos dados armazenados no banco de dados. Por exemplo, tabelas com chaves primárias numéricas bem distribuída ou colunas de índice serão exportadas de maneira mais rápida. As tabelas que não contêm uma coluna adequada para particionamento e as tabelas com somente um índice em uma coluna baseada em string levam mais tempo porque a exportação usa um processo em thread único mais lento.

Você pode exportar os dados de cluster de banco de dados para o Amazon S3 usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Se você usar uma função do Lambda para exportar os dados do cluster de banco de dados, adicione a ação `kms:DescribeKey` à política da função do Lambda. Para ter mais informações, consulte Permissões doAWS Lambda.

Console

A opção do console Export to Amazon S3 (Exportar para o Amazon S3) é exibido somente para clusters de banco de dados que podem ser exportados para o Amazon S3. Um cluster de banco de dados pode não estar disponível para exportação devido aos seguintes motivos:

- O mecanismo de banco de dados não tem suporte para exportação do S3.
- A versão do cluster de banco de dados não é compatível com a exportação do S3.
- A exportação do S3 não é aceita na região da AWS onde o cluster de banco de dados foi criado.

Como exportar dados do cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Selecione o cluster de banco de dados cujos dados você deseja exportar.
4. Em Actions (Ações), escolha Export to Amazon S3 (Exportar para o Amazon S3).

   A janela Export to Amazon S3 (Exportar para o Amazon S3) é exibida.
5. Em Export identifier (Identificador de exportação), insira um nome para identificar a tarefa de exportação. Esse valor também é usado para o nome do arquivo criado no bucket do S3.
6. Escolha os dados a serem exportados:

   - Selecione All (Todos) para exportar todos os dados do cluster de banco de dados.
   - Selecione Partial (Parcial) para exportar partes específicas do cluster de banco de dados. Para identificar quais partes do cluster exportar, insira um ou mais bancos de dados, esquemas ou tabelas para Identifiers (Identificadores), separados por espaços.

     Use o formato a seguir:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
     [.tablen]
     ```

Por exemplo:

```
mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
 mydatabase2.myschema2.mytable2
```

7. Em S3 bucket (Bucket do S3), escolha o bucket para o qual exportar.

   Para atribuir os dados exportados a um caminho de pasta no bucket do S3, insira o caminho opcional para o S3 prefix (Prefixo do S3).

8. Em IAM role (Perfil do IAM), escolha um perfil que conceda a você acesso de gravação ao bucket do S3 escolhido ou crie um novo perfil.

   - Se você criou um perfil seguindo as etapas em Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM, escolha esse perfil.
   - Se você não criou um perfil que conceda acesso de gravação ao bucket do S3 escolhido, selecione Create a new role (Criar um perfil) para criar o perfil automaticamente. Depois, insira um nome para o perfil em Nome do perfil do IAM.

9. Em KMS key (Chave do KMS), insira o ARN da chave a ser usada para criptografar os dados exportados.

10. Escolha Export to Amazon S3 (Exportar para o Amazon S3).

AWS CLI

Para exportar dados do cluster de banco de dados para o Amazon S3 usando a AWS CLI, use o comando start-export-task com as seguintes opções necessárias:

- `--export-task-identifier`
- `--source-arn`: o nome do recurso da Amazon (ARN) do cluster de banco de dados
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Nos exemplos a seguir, a tarefa de exportação é chamada *`my-cluster-export`*, que exporta os dados para um bucket do S3 chamado *`amzn-s3-demo-destination-bucket`*.

Example

Para Linux, macOS ou Unix:

```
aws rds start-export-task \
    --export-task-identifier my-cluster-export \
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Para Windows:

```
aws rds start-export-task ^
    --export-task-identifier my-DB-cluster-export ^
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Segue um exemplo de saída.

```
{
    "ExportTaskIdentifier": "my-cluster-export",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "IamRoleArn": "arn:aws:iam:123456789012:role/ExportTest",
    "KmsKeyId": "my-key",
    "Status": "STARTING",
    "PercentProgress": 0,
    "TotalExtractedDataInGB": 0,
}
```

Para fornecer um caminho de pasta no bucket do S3 para a exportação do cluster de banco de dados, inclua a opção `--s3-prefix` no comando start-export-task.

API do RDS

Para exportar dados do cluster de banco de dados para o Amazon S3 usando a API do Amazon RDS, use a operação StartExportTask com os seguintes parâmetros necessários:

- `ExportTaskIdentifier`

- `SourceArn`: o ARN do cluster de banco de dados
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitorar tarefas de exportação do cluster de banco de dados

Você pode monitorar as exportações de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI e a API do RDS.

Console

Como monitorar exportações do cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Exports in Amazon S3 (Exportações no Amazon S3).

   As exportações do cluster de banco de dados são indicadas na coluna Source type (Tipo de origem). O status da exportação é exibido na coluna Status.
3. Para visualizar informações detalhadas sobre uma exportação de cluster de banco de dados específica, selecione a tarefa de exportação.

AWS CLI

Para monitorar exportações de cluster de banco de dados usando a AWS CLI, utilize o comando describe-export-tasks.

O exemplo a seguir mostra como exibir informações atuais sobre todas as exportações do cluster de banco de dados.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
```

```
            "TaskEndTime": "2022-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:parameter-groups-
test"
        },
{
            "Status": "COMPLETE",
            "TaskStartTime": "2022-10-31T20:58:06.998Z",
            "TaskEndTime": "2022-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2022-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket amzn-s3-demo-bucket2 isn't located in the
 current AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-30-06-45"
```

```
        }
    ]
}
```

Para exibir informações sobre uma tarefa de exportação específica, inclua a opção `--export-task-identifier` com o comando `describe-export-tasks`. Para filtrar a saída, inclua a opção `--Filters`. Para obter mais opções, consulte o comando describe-export-tasks.

API do RDS

Para exibir informações sobre exportações de cluster de banco de dados usando a API do Amazon RDS, use a operação DescribeExportTasks.

Para rastrear a conclusão do fluxo de trabalho de exportação ou acionar outro fluxo de trabalho, você pode assinar tópicos do Amazon Simple Notification Service. Para ter mais informações sobre Amazon SNS, consulte Trabalhar com a notificação de eventos do Amazon RDS.

Cancelar uma tarefa de exportação de cluster de banco de dados

Você pode cancelar uma tarefa de exportação de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Note**
>
> O cancelamento de uma tarefa de exportação não remove os dados que foram exportados para o Amazon S3. Para obter informações sobre como excluir os dados usando o console, consulte Como excluir objetos de um bucket do S3? Para excluir os dados usando a CLI, use o comando delete-object.

Console

Como cancelar uma tarefa de exportação de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Exports in Amazon S3 (Exportações no Amazon S3).

   As exportações do cluster de banco de dados são indicadas na coluna Source type (Tipo de origem). O status da exportação é exibido na coluna Status.

3. Selecione a tarefa de exportação que você deseja cancelar.
4. Escolha Cancelar.
5. Escolha Cancel export task (Cancelar tarefa de exportação) na página de confirmação.

AWS CLI

Para cancelar uma tarefa de exportação usando a AWS CLI, use o comando cancel-export-task. O comando requer a opção `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my-export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my-export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:export-example-1"
}
```

API do RDS

Para cancelar uma tarefa de exportação usando a API do Amazon RDS, use a operação CancelExportTask com o parâmetro `ExportTaskIdentifier`.

Solucionar problemas do cluster de banco de dados

Use as seções a seguir para ajudar a solucionar mensagens de falha e erros de permissão do PostgreSQL para tarefas de exportação de clusters de banco de dados para o Amazon S3.

Mensagens de falha de tarefas de exportação do Amazon S3

A tabela a seguir descreve as mensagens que são retornadas quando ocorrem falhas nas tarefas de exportação do Amazon S3.

Mensagem de falha	Descrição
Falha ao encontrar ou acessar o cluster de banco de dados de origem: [nome do cluster]	O cluster de banco de dados de origem não pode ser clonado.
Ocorreu um erro interno desconhecido.	O processamento da falha devido a um erro, uma exceção ou uma falha desconhecida.
Ocorreu um erro interno desconhecido ao gravar os metadados da tarefa de exportação no bucket do S3 [nome do bucket].	O processamento da falha devido a um erro, uma exceção ou uma falha desconhecida.
A exportação do RDS falhou ao gravar os metadados da tarefa de exportação porque ela não pode assumir o perfil do IAM [ARN do perfil].	A tarefa de exportação assume seu perfil do IAM para validar se pode gravar metadados no seu bucket do S3. Se a tarefa não puder assumir seu perfil do IAM, ela falhará.
A exportação do RDS falhou ao gravar os metadados da tarefa de exportaçã o no bucket do S3 [nome do bucket] utilizando o perfil do IAM [ARN do perfil] com a chave do KMS [ID da chave]. Código de erro: [código de erro]	Uma ou mais permissões estão ausentes e, portanto, a tarefa de exportação não consegue acessar o bucket do S3. Essa mensagem de falha é gerada quando você recebe um dos seguintes códigos de erro: • `AWSSecurityTokenServiceException` com o código de erro `AccessDenied` • `AmazonS3Exception` com o código de erro `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` ou `KMS.DisabledException` Esses códigos de erro indicam que as configurações estão definidas incorretamente para o perfil do IAM, o bucket do S3 ou a chave do KMS.
O perfil do IAM [ARN do perfil] não está autorizado a chamar [ação do S3] no	A política do IAM está incorretamente configurada. A permissão para a ação específica do S3 no bucket

Mensagem de falha	Descrição
bucket do S3 [nome do bucket]. Revise suas permissões e tente novamente a exportação.	do S3 está ausente, o que causa falha na tarefa de exportação.
A verificação da chave do KMS falhou. Verifique as credenciais na sua chave do KMS e tente novamente.	A verificação de credenciais da chave KMS do falhou.
Falha na verificação de credenciais do S3. Verifique as permissões no bucket do S3 e a política do IAM.	A verificação de credenciais do S3 falhou.
O bucket do S3 [nome do bucket] não é válido. Ele não está localizado na Região da AWS atual ou não existe. Reveja o nome do bucket do S3 e tente exportar novamente.	O bucket do S3 não é válido.
O bucket do S3 [nome do bucket] não está localizado na Região da AWS atual. Reveja o nome do bucket do S3 e tente exportar novamente.	O bucket da S3 está na Região da AWS incorreta.

Solucionar problemas de erros de permissões do PostgreSQL

Ao exportar bancos de dados PostgreSQL para o Amazon S3, é possível ver um erro `PERMISSIONS_DO_NOT_EXIST` informando que determinadas tabelas foram ignoradas. Esse erro geralmente ocorre quando o superusuário, especificado ao criar o cluster de banco de dados, não tem permissões para acessar essas tabelas.

Para corrigir esse erro, execute o seguinte comando:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Para ter mais informações sobre privilégios de superusuário, consulte Privilégios da conta de usuário mestre.

Exportar dados de snapshot de cluster de banco de dados para o Amazon S3

É possível exportar dados de snapshots de clusters de banco de dados para um bucket do Amazon S3. O processo de exportação é executado em segundo plano e não afeta a performance do cluster de banco de dados ativo.

Ao exportar um snapshot de cluster de banco de dados, o Amazon Aurora extrai os dados do snapshot e os armazena em um bucket do Amazon S3. Você pode exportar snapshots manuais e snapshots automatizados do sistema. Por padrão, todos os dados no snapshot são exportados. No entanto, você pode optar por exportar conjuntos específicos de bancos de dados, esquemas ou tabelas.

> **Note**
>
> A exportação de dados de um snapshot do cluster de banco de dados exige a restauração do snapshot. Os tempos de restauração são afetados por vários fatores, como o volume de tráfego de rede que uma Região da AWS recebe em relação à largura de banda disponível. Quando há um aumento repentino no tráfego, é possível ocorrer tempos de conclusão mais longos do que o esperado.
>
> Uma alternativa para reduzir os tempos de exportação do S3 para bancos de dados Aurora é a exportação dinâmica do cluster de banco de dados para o S3. A exportação do cluster de banco de dados tem horários de início mais curtos do que a exportação de snapshots de banco de dados, porque não há necessidade de restaurar o snapshot. Para ter mais informações, consulte Exportar dados do cluster de banco de dados para o Amazon S3.

Os dados são armazenados em um formato Apache Parquet que é compactado e consistente. Em geral, os arquivos Parquet individuais têm cerca de 1 a 10 MB de tamanho.

Depois que os dados são exportados, você pode analisar os dados exportados diretamente por meio de ferramentas, como Amazon Athena ou Amazon Redshift Spectrum. Para ter mais informações sobre como usar o Athena para ler os dados do Parquet, consulte Parquet SerDe no Guia do usuário do Amazon Athena. Para ter mais informações sobre como usar o Redshift Spectrum para ler os dados do Parquet, consulte COPY de formatos de dados colunares no Guia do desenvolvedor de banco de dados do Amazon Redshift.

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e regiões para a exportação de snapshots de cluster de banco de dados para o S3, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a exportação de dados de snapshots para o Amazon S3.

Use o seguinte processo para exportar dados de um snapshot de banco de dados para um bucket do Amazon S3. Para obter mais detalhes, consulte as seções a seguir.

Visão geral da exportação de dados de snapshot

1. Identifique o snapshot a ser exportado.

   Use um snapshot existente, manual ou automatizado, ou crie um snapshot manual de uma instância de banco de dados.

2. Configure o acesso ao bucket do Amazon S3.

   Um bucket é um contêiner de objetos ou arquivos do Amazon S3. Para fornecer informações para acesso a um bucket, execute as seguintes etapas:

   a. Identifique o bucket do S3 para o qual o snapshot deve ser exportado. O bucket do S3 deve estar na mesma região da AWS que o snapshot. Para ter mais informações, consulte Identificar o bucket do Amazon S3 para exportar.

   b. Crie uma função do AWS Identity and Access Management (IAM) que conceda à tarefa de exportação de snapshot acesso ao bucket do S3. Para ter mais informações, consulte Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM.

3. Crie uma criptografia simétrica do AWS KMS key para a criptografia do lado do servidor. A chave do KMS é usada pela tarefa de exportação de snapshot para configurar a criptografia do lado do servidor do AWS KMS ao gravar dados de exportação no S3.

   A política de chave do KMS deve incluir as permissões `kms:CreateGrant` e `kms:DescribeKey`. Para ter mais informações sobre o uso de chaves do KMS no Amazon Aurora, consulte AWS KMS keyGerenciamento de.

   Além disso, se você tiver uma instrução de negação na política de chaves do KMS, exclua explicitamente a entidade principal de serviço da AWS `export.rds.amazonaws.com`.

   Você pode utilizar uma chave do KMS na sua conta do AWS ou pode utilizar uma chave do KMS entre contas. Para ter mais informações, consulte Utilizar uma conta cruzada AWS KMS key.

4. Exporte o snapshot para o Amazon S3 usando o console ou o comando `start-export-task` da CLI. Para ter mais informações, consulte Criar tarefas de exportação de snapshot.
5. Para acessar os seus dados exportados no bucket do Amazon S3, consulte Como carregar, baixar e gerenciar objetos no Guia do usuário do Amazon Simple Storage Service.

Aprenda a configurar, exportar, monitorar, cancelar e solucionar problemas em tarefas de exportação de snapshots de cluster de banco de dados nas seções a seguir.

Tópicos

- Considerações sobre exportações de snapshot de cluster de banco de dados
- Configurar o acesso a um bucket do Amazon S3
- Criar tarefas de exportação de snapshot
- Monitorar exportações de snapshots
- Cancelar uma tarefa de exportação de snapshot
- Performance de exportação no Aurora MySQL
- Solução de problemas com a exportação de snapshots

Considerações sobre exportações de snapshot de cluster de banco de dados

Limitações

A exportação de dados de snapshots de banco de dados para o Amazon S3 apresenta as seguintes limitações:

- Você não pode executar várias tarefas de exportação para o mesmo snapshot de cluster de banco de dados simultaneamente. Isso se aplica a exportações totais e parciais.
- Você pode ter até cinco tarefas de exportação de snapshot de banco de dados simultâneas em andamento por Conta da AWS.
- Não é possível restaurar dados de snapshots exportados de clusters de banco de dados do Aurora Serverless v1 para o S3.
- As exportações para o S3 não são compatíveis com prefixos do S3 contendo dois pontos (:).
- Os seguintes caracteres no caminho do arquivo do S3 são convertidos em sublinhados (_) durante a exportação:

```
\ ` " (space)
```

- Se um banco de dados, esquema ou tabela tiver caracteres em seu nome diferentes dos a seguir, a exportação parcial não será aceita. No entanto, você pode exportar o snapshot de banco de dados inteiro.
  - Letras latinas (A–Z)
  - Dígitos (0–9)
  - Símbolo do dólar ($)
  - Sublinhado (_)
- Espaços ( ) e alguns caracteres não são compatíveis em nomes de colunas de tabelas de banco de dados. As tabelas com os seguintes caracteres em nomes de colunas são ignoradas durante a exportação:

```
, ; { } ( ) \n \t = (space)
```

- As tabelas com barras (/) em seus nomes são ignoradas durante a exportação.
- As tabelas temporárias e não registradas do Aurora PostgreSQL são ignoradas durante a exportação.
- Se os dados contiverem um objeto grande, como um BLOB ou um CLOB, com tamanho próximo ou superior a 500 MB, ocorrerá uma falha na exportação.
- Se uma tabela contiver uma linha grande próxima ou maior que 2 GB, a tabela será ignorada durante a exportação.
- Em relação a exportações parciais, a lista `ExportOnly` tem um tamanho máximo de 200 KB.
- É altamente recomendável que você use um nome exclusivo para cada tarefa de exportação. Se você não usar um nome de tarefa exclusivo, poderá receber a seguinte mensagem de erro:

  ExportTaskAlreadyExistsFault: An error occurred (ExportTaskAlreadyExists) when calling the StartExportTask operation: The export task with the ID *xxxxx* already exists [ExportTaskAlreadyExistsFault: ocorreu um erro (ExportTaskAlreadyExists) ao chamar a operação StartExportTask: a tarefa de exportação com o ID xxxxx já existe].
- Você pode excluir um snapshot enquanto exporta seus dados para o S3, mas ainda é cobrado pelo armazenamento desse snapshot até que a tarefa de exportação seja concluída.
- Não é possível restaurar dados de snapshots exportados do S3 para um novo cluster de banco de dados.

Convenção de nomenclatura de arquivos

Os dados exportados para tabelas específicas são armazenados no formato *base_prefix*/*files*, onde o prefixo base é o seguinte:

```
export_identifier/database_name/schema_name.table_name/
```

Por exemplo:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

Existem duas convenções de nomenclatura para arquivos.

- Convenção atual:

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  O índice do lote é um número de sequência que representa um lote de dados lidos da tabela. Se não conseguirmos particionar a tabela em pequenos blocos para serem exportados paralelamente, haverá vários índices de lote. O mesmo acontecerá se a tabela for particionada em várias tabelas. Haverá vários índices de lote, um para cada partição da tabela principal.

  Se conseguirmos particionar a tabela em pequenos blocos para serem lidos paralelamente, haverá apenas a pasta 1 de índices de lote.

  Na pasta de índices de lote, há um ou mais arquivos Parquet que contêm os dados da sua tabela. O prefixo do nome do arquivo Parquet é `part-`*partition_index*. Se a tabela estiver particionada, haverá vários arquivos que começam com o índice de partição `00000`.

  Pode haver lacunas na sequência do índice de partição. Isso acontece porque cada partição é obtida de uma consulta por intervalos na tabela. Se não houver dados no intervalo dessa partição, esse número de sequência será ignorado.

  Por exemplo, suponha que a coluna `id` seja a chave primária da tabela e os valores mínimo e máximo sejam `100` e `1000`. Quando tentamos exportar essa tabela com nove partições, a lemos com consultas paralelas, como as seguintes:

  ```
  SELECT * FROM table WHERE id <= 100 AND id < 200
   SELECT * FROM table WHERE id <= 200 AND id < 300
  ```

Isso deve gerar nove arquivos, de `part-00000-`*`random_uuid`*`.gz.parquet` a `part-00008-`*`random_uuid`*`.gz.parquet`. No entanto, se não houver linhas com IDs entre `200` e `350`, uma das partições concluídas estará vazia e nenhum arquivo será criado para ela. No exemplo anterior, `part-00001-`*`random_uuid`*`.gz.parquet` não foi criado.

- Convenção mais antiga:

```
part-partition_index-random_uuid.format-based_extension
```

É igual à convenção atual, mas sem o prefixo *`batch_index`*, por exemplo:

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
 part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
 part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

A convenção de nomenclatura de arquivos está sujeita a alterações. Portanto, ao ler as tabelas de destino, recomendamos que você leia tudo dentro do prefixo base para a tabela.

Conversão de dados ao exportar para um bucket do Amazon S3

Ao exportar um snapshot de banco de dados para um bucket do Amazon S3, o Amazon Aurora converte os dados para o formato Parquet e exporta e armazena os dados nesse formato. Para ter mais informações sobre o Parquet, consulte o site Apache Parquet.

O Parquet armazena todos os dados como um dos seguintes tipos primitivos:

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY: uma matriz de bytes de comprimento variável, também conhecida como binário
- FIXED_LEN_BYTE_ARRAY: uma matriz de bytes de comprimento fixo usada quando os valores têm um tamanho constante

Os tipos de dados Parquet são poucos para reduzir a complexidade de leitura e gravação do formato. O Parquet fornece tipos lógicos para estender os tipos primitivos. Um tipo lógico é implementado como uma anotação com os dados em um campo de metadados `LogicalType`. A anotação de tipo lógico explica como interpretar o tipo primitivo.

Quando o tipo lógico `STRING` anota um tipo `BYTE_ARRAY`, ele indica que a matriz de bytes deve ser interpretada como uma string de caracteres codificada em UTF-8. Depois que uma tarefa de exportação é concluída, o Amazon Aurora notificará você se uma conversão de string tiver ocorrido. Os dados subjacentes exportados são sempre os mesmos que os dados da origem. No entanto, devido à diferença de codificação em UTF-8, alguns caracteres podem aparecer diferentes da fonte quando lidos em ferramentas como a Athena.

Para ter mais informações, consulte Definições de tipos lógicos do Parquet na documentação do Parquet.

Tópicos

- Mapeamento de tipos de dados MySQL para o Parquet
- Mapeamento de tipo de dados PostgreSQL para Parquet

Mapeamento de tipos de dados MySQL para o Parquet

A tabela a seguir mostra o mapeamento dos tipos de dados MySQL para tipos de dados Parquet quando os dados são convertidos e exportados para o Amazon S3.

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
Tipos de dados numéricos			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	O Parquet é compatível apenas com tipos assinados , portanto, o mapeamento requer um byte adicional (8 mais 1) para

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
			armazenar o tipo BIGINT_UNSIGNED.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL (p,s)	Se o valor de origem for menor que 2^{31}, ele será armazenado como INT32.
	INT64	DECIMAL (p,s)	Se o valor de origem for 2^{31} ou maior, mas inferior a 2^{63}, ele será armazenado como INT64.
	FIXED_LEN _BYTE_ARRAY(N)	DECIMAL (p,s)	Se o valor de origem for 2^{63} ou superior, ele será armazenad o como FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	O Parquet não é compatível com precisão decimal maior que 38. O valor decimal é convertid o em uma string em um tipo BYTE_ARRA Y e codificado como UTF8.
DOUBLE	DOUBLE		
FLOAT	DOUBLE		

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		
NUMERIC	INT32	DECIMAL (p,s)	Se o valor de origem for menor que 2^{31}, ele será armazenado como INT32.
	INT64	DECIMAL (p,s)	Se o valor de origem for 2^{31} ou maior, mas inferior a 2^{63}, ele será armazenado como INT64.
	FIXED_LEN _ARRAY(N)	DECIMAL (p,s)	Se o valor de origem for 2^{63} ou superior, ele será armazenad o como FIXED_LEN _BYTE_ARRAY(N).

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
	BYTE_ARRAY	STRING	O Parquet não é compatível com precisão numérica maior que 38. Esse valor numérico é convertido em uma string em um tipo BYTE_ARRAY e codificado como UTF8.
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
Tipos de dados de string			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
Tipos de dados de data e hora			
DATE	BYTE_ARRAY	STRING	Uma data é convertida em uma string em um tipo BYTE_ARRAY e codificada como UTF8.
DATETIME	INT64	TIMESTAMP_MICROS	
TIME	BYTE_ARRAY	STRING	Um tipo TIME é convertido em uma string em um BYTE_ARRAY e codificado como UTF8.
TIMESTAMP	INT64	TIMESTAMP_MICROS	

Tipo de dados de origem	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de conversão
YEAR	INT32		
Tipos de dados geométricos			
GEOMETRY	BYTE_ARRAY		
GEOMETRYCOLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		
Tipo de dados do JSON			
JSON	BYTE_ARRAY	STRING	

Mapeamento de tipo de dados PostgreSQL para Parquet

A tabela a seguir mostra o mapeamento de tipos de dados PostgreSQL para tipos de dados Parquet quando os dados são convertidos e exportados para o Amazon S3.

Tipo de dados do PostgreSQL	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de mapeamento
Tipos de dados numéricos			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	Um tipo DECIMAL é convertido em

Tipo de dados do PostgreSQL	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de mapeamento
			uma string em um tipo BYTE_ARRAY e codificado como UTF8. Essa conversão é para evitar complicações devido à precisão dos dados e valores de dados que não são um número (NaN).
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
String e tipos de dados relacionados			

Tipo de dados do PostgreSQL	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de mapeamento
ARRAY	BYTE_ARRAY	STRING	Uma matriz é convertida em uma string e codificada como BINARY (UTF8). Essa conversão é para evitar complicações devido à precisão dos dados, os valores dos dados que não são um número (NaN) e os valores de dados de tempo.
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
XML	BYTE_ARRAY	STRING	

Tipo de dados do PostgreSQL	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de mapeamento
Tipos de dados de data e hora			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
Tipos de dados geométricos			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
Tipos de dados JSON			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
Outros tipos de dados			

Tipo de dados do PostgreSQL	Tipo primitivo do Parquet	Anotação de tipo lógico	Notas de mapeamento
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	Tipo de dados de rede
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	Tipo de dados de rede
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	N/D		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Configurar o acesso a um bucket do Amazon S3

Você identifica o bucket do Amazon S3, depois fornece ao snapshot permissão para acessá-lo.

Tópicos

- Identificar o bucket do Amazon S3 para exportar
- Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM
- Utilizar um bucket do Amazon S3 entre contas
- Utilizar uma conta cruzada AWS KMS key

Identificar o bucket do Amazon S3 para exportar

Identifique o bucket do Amazon S3 para o qual exportar o snapshot de banco de dados. Use um bucket do S3 existente ou crie um novo bucket do S3.

> **Note**
>
> O bucket do S3 para o qual exportar deve estar na mesma região da AWS que o snapshot.

Para ter mais informações sobre como trabalhar com buckets do Amazon S3, consulte o seguinte no Guia do usuário do Amazon Simple Storage Service:

- Como visualizar as propriedades de um bucket do S?
- Como habilitar a criptografia padrão em um bucket do Amazon S?
- Como criar um bucket do S?

Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM

Antes de exportar dados de snapshot de banco de dados para o Amazon S3, forneça permissões de acesso de gravação ao bucket do Amazon S3 às tarefas de exportação.

Para conceder essa permissão, crie uma política do IAM que forneça acesso ao bucket e, depois, crie um perfil do IAM e anexe a política ao perfil. Posteriormente, você atribui o perfil do IAM à tarefa de exportação de snapshot.

> **Important**
>
> Se você planeja usar o Console de gerenciamento da AWS para exportar o snapshot, poderá optar por criar a política do IAM e a função automaticamente ao exportar o snapshot. Para instruções, consulte Criar tarefas de exportação de snapshot.

Como fornecer acesso ao Amazon S3 às tarefas de snapshot de banco de dados

1. Crie uma política do IAM. Essa política fornece as permissões de bucket e objeto que permitem que sua tarefa de exportação de snapshot acesse o Amazon S3.

Na política, inclua as ações necessárias a seguir para permitir a transferência de arquivos do Amazon Aurora para um bucket do S3:

- `s3:PutObject*`
- `s3:GetObject*`
- `s3:ListBucket`
- `s3:DeleteObject*`
- `s3:GetBucketLocation`

Na política, inclua os recursos a seguir para identificar o bucket do S3 e os objetos no bucket. A lista de recursos a seguir mostra o formato do nome de recurso da Amazon (ARN) para acessar o Amazon S3.

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Para ter mais informações sobre como criar uma política do IAM para o Amazon Aurora, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM. Consulte também Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

O comando da AWS CLI a seguir cria uma política do IAM denominada `ExportPolicy` com essas opções. Ele concede acesso a um bucket denominado *amzn-s3-demo-bucket*.

> **Note**
>
> Depois de criar a política, anote o ARN da política. O ARN será necessário para uma etapa posterior, quando você anexar a política a um perfil do IAM.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crie um perfil do IAM para que o Aurora possa assumir esse perfil do IAM em seu nome a fim de acessar os buckets do Amazon S3. Para ter mais informações, consulte Criar um perfil para delegar permissões a um usuário do IAM no Guia do usuário do IAM.

   O exemplo a seguir mostra como usar o comando da AWS CLI para criar uma função chamada `rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Anexe a política do IAM que você criou ao perfil do IAM que você criou.

   O seguinte comando da AWS CLI anexa a política criada anteriormente à função chamada `rds-s3-export-role`. Substitua *`your-policy-arn`* pelo ARN da política que você anotou em uma etapa anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Utilizar um bucket do Amazon S3 entre contas

É possível utilizar buckets do Amazon S3 entre contas da AWS. Para utilizar um bucket entre contas, adicione uma política de bucket para permitir o acesso ao perfil do IAM que você está utilizando para as exportações do S3. Para ter mais informações, consulte Exemplo 2: proprietário do bucket concedendo permissões para o bucket entre contas.

- Vincule uma política de bucket ao bucket, como mostra o exemplo a seguir.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

Utilizar uma conta cruzada AWS KMS key

Você pode utilizar um AWS KMS key entre contas para criptografar exportações do Amazon S3. Primeiro, você adiciona uma política de chaves à conta local e depois adiciona políticas do IAM na conta externa. Para ter mais informações, consulte o tópico sobre como Permitir que usuários de outras contas utilizem uma chave do KMS.

Para utilizar uma chave do KMS entre contas

1. Adicione uma política de chaves à conta local.

   O exemplo a seguir concede a `ExampleRole` e `ExampleUser` na conta externa 444455556666 permissões na conta local 123456789012.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

2. Adicione políticas do IAM à conta externa.

   O seguinte exemplo de política do IAM permite que a entidade principal use a chave do KMS na conta 123456789012 para operações criptográficas. Para conceder essa permissão a `ExampleRole` e `ExampleUser` na conta 444455556666, vincule a política a eles nessa conta.

```
{
```

```
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Criar tarefas de exportação de snapshot

Crie tarefas de exportação de snapshot para exportar dados do snapshot para um bucket do Amazon S3. Você pode ter até cinco tarefas de exportação de snapshot de banco de dados simultâneas em andamento por Conta da AWS.

> **Note**
>
> A exportação de snapshots do RDS pode levar um tempo, dependendo do tipo e do tamanho do banco de dados. A tarefa de exportação primeiro restaura e escala todo o banco de dados antes de extrair os dados para o Amazon S3. O andamento da tarefa durante essa fase é exibido como Starting (Iniciando). Quando a tarefa muda para a exportação de dados para o S3, o andamento é exibido como In progress (Em andamento).
>
> O tempo necessário para que a exportação seja concluída depende dos dados armazenados no banco de dados. Por exemplo, tabelas com chaves primárias numéricas bem distribuída ou colunas de índice serão exportadas de maneira mais rápida. Tabelas sem uma coluna adequada para particionamento e tabelas com somente um índice em uma coluna baseada em string demorarão mais. Esse tempo de exportação mais demorado ocorre porque a exportação utiliza um processo de thread único mais lento.

Você pode exportar um snapshot de banco de dados para o Amazon S3 usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Se você usar uma função do Lambda para exportar um snapshot, adicione a ação `kms:DescribeKey` à política da função do Lambda. Para ter mais informações, consulte Permissões doAWS Lambda.

Console

A opção de console Export to Amazon S3 (Exportar para o Amazon S3) é exibido somente para snapshots que podem ser exportados para o Amazon S3. Um snapshot pode não estar disponível para exportação devido aos seguintes motivos:

- O mecanismo de banco de dados não tem suporte para exportação do S3.
- A versão da instância de banco de dados não tem suporte para exportação do S3.
- A exportação do S3 não é aceita na região da AWS onde o snapshot foi criado.

Para exportar um snapshot de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Snapshots.
3. Nas guias, escolha o tipo de snapshot que deseja exportar.
4. Na lista de snapshots, escolha o snapshot que deseja exportar.
5. Em Actions (Ações), escolha Export to Amazon S3 (Exportar para o Amazon S3).

   A janela Export to Amazon S3 (Exportar para o Amazon S3) é exibida.
6. Em Export identifier (Identificador de exportação), insira um nome para identificar a tarefa de exportação. Esse valor também é usado para o nome do arquivo criado no bucket do S3.
7. Escolha os dados a serem exportados:

   - Escolha All (Tudo) para exportar todos os dados do snapshot.
   - Escolha Partial (Parcial) para exportar partes específicas do snapshot. Para identificar quais partes do snapshot exportar, insira um ou mais bancos de dados, esquemas ou tabelas para Identifiers (Identificadores), separados por espaços.

     Use o formato a seguir:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
     [.tablen]
     ```

Por exemplo:

```
mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
 mydatabase2.myschema2.mytable2
```

8. Em S3 bucket (Bucket do S3), escolha o bucket para o qual exportar.

   Para atribuir os dados exportados a um caminho de pasta no bucket do S3, insira o caminho opcional para o S3 prefix (Prefixo do S3).
9. Em IAM role (Perfil do IAM), escolha um perfil que conceda a você acesso de gravação ao bucket do S3 escolhido ou crie um novo perfil.

   - Se você criou um perfil seguindo as etapas em Fornecer acesso a um bucket do Amazon S3 usando um perfil do IAM, escolha esse perfil.
   - Se você não criou um perfil que conceda acesso de gravação ao bucket do S3 escolhido, selecione Create a new role (Criar um perfil) para criar o perfil automaticamente. Depois, insira um nome para o perfil em Nome do perfil do IAM.
10. Em AWS KMS key, insira o ARN da chave a ser usada para criptografar os dados exportados.
11. Escolha Export to Amazon S3 (Exportar para o Amazon S3).

AWS CLI

Para exportar um snapshot de banco de dados para o Amazon S3 usando a AWS CLI, use o comando start-export-task com as seguintes opções necessárias:

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Nos exemplos a seguir, a tarefa de exportação do snapshot é denominada *`my_snapshot_export`*, que exporta um snapshot para um bucket do S3 chamado *`amzn-s3-demo-destination-bucket`*.

Example

Para Linux, macOS ou Unix:

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Para Windows:

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Segue um exemplo de saída.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

Para fornecer um caminho de pasta no bucket do S3 para a exportação do snapshot, inclua a opção `--s3-prefix` no comando start-export-task.

API do RDS

Para exportar um snapshot de banco de dados para o Amazon S3, usando a API do Amazon RDS, use a operação StartExportTask com os seguintes parâmetros necessários:

- `ExportTaskIdentifier`
- `SourceArn`

- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitorar exportações de snapshots

Você pode monitorar as exportações de snapshots de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI e a API do RDS.

Console

Como monitorar exportações de snapshots de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Exports in Amazon S3 (Exportações no Amazon S3).

   As exportações de snapshot de banco de dados são indicadas na coluna Source type (Tipo de origem). O status da exportação é exibido na coluna Status.
3. Para visualizar informações detalhadas sobre uma exportação de snapshot específica, selecione a tarefa de exportação.

AWS CLI

Para monitorar exportações de snapshots de banco de dados usando o AWS CLI, use o comando describe-export-tasks.

O exemplo a seguir mostra como exibir informações atuais sobre todas as exportações de snapshots.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
```

```
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket my-exports isn't located in the current AWS
 Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
```

```
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

Para exibir informações sobre uma exportação de snapshot específica, inclua a opção `--export-task-identifier` com o comando `describe-export-tasks`. Para filtrar a saída, inclua a opção `--Filters`. Para obter mais opções, consulte o comando describe-export-tasks.

API do RDS

Para exibir informações sobre exportações de snapshots de banco de dados usando a API do Amazon RDS, use a operação DescribeExportTasks .

Para rastrear a conclusão do fluxo de trabalho de exportação ou acionar outro fluxo de trabalho, você pode assinar tópicos do Amazon Simple Notification Service. Para ter mais informações sobre Amazon SNS, consulte Trabalhar com a notificação de eventos do Amazon RDS.

Cancelar uma tarefa de exportação de snapshot

Você pode cancelar uma tarefa de exportação de snapshot de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Note**
>
> O cancelamento de uma tarefa de exportação de snapshot não remove os dados que foram exportados para o Amazon S3. Para obter informações sobre como excluir os dados usando o console, consulte Como excluir objetos de um bucket do S3? Para excluir os dados usando a CLI, use o comando delete-object.

Console

Como cancelar uma tarefa de exportação de snapshot

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Exports in Amazon S3 (Exportações no Amazon S3).

As exportações de snapshot de banco de dados são indicadas na coluna Source type (Tipo de origem). O status da exportação é exibido na coluna Status.

3. Escolha a tarefa de exportação do snapshot que você deseja cancelar.
4. Escolha Cancelar.
5. Escolha Cancel export task (Cancelar tarefa de exportação) na página de confirmação.

AWS CLI

Para cancelar uma tarefa de exportação de snapshot usando a AWS CLI, use o comando cancel-export-task. O comando requer a opção `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
}
```

API do RDS

Para cancelar uma tarefa de exportação de snapshot usando a API do Amazon RDS, use a operação CancelExportTask com o parâmetro `ExportTaskIdentifier`.

Performance de exportação no Aurora MySQL

Os snapshots de cluster de banco de dados Aurora MySQL versão 2 e versão 3 usam um mecanismo de exportação avançado para melhorar a performance e reduzir o tempo de exportação. O mecanismo inclui otimizações, como vários segmentos de exportação e consulta paralela do Aurora MySQL para aproveitar a arquitetura de armazenamento compartilhado do Aurora. As

otimizações são aplicadas de forma adaptável, dependendo do tamanho e da estrutura do conjunto de dados.

Você não precisa ativar a consulta paralela para usar o processo de exportação com maior rapidez, mas o processo tem as mesmas limitações da consulta paralela. Além disso, alguns valores de dados não são compatíveis, como datas em que é o dia do mês é `0` ou o ano é `0000`. Para ter mais informações, consulte Consulta paralela do Amazon Aurora MySQL.

Quando as otimizações de performance são aplicadas, você também pode ver arquivos Parquet muito maiores (cerca de 200 GB) para exportações do Aurora MySQL versões 2 e 3.

Se o processo de exportação mais rápido não puder ser usado, por exemplo, devido a tipos ou valores de dados incompatíveis, o Aurora mudará automaticamente para um modo de exportação com um único segmento sem consulta paralela. Dependendo do processo utilizado e da quantidade de dados a serem exportados, a performance da exportação pode variar.

Solução de problemas com a exportação de snapshots

Use as seções a seguir para ajudar a solucionar mensagens de falha e erros de permissão do PostgreSQL para tarefas de exportação de clusters de banco de dados para o Amazon S3.

Mensagens de falha de tarefas de exportação do Amazon S3

A tabela a seguir descreve as mensagens que são retornadas quando ocorrem falhas nas tarefas de exportação do Amazon S3.

Mensagem de falha	Descrição
Ocorreu um erro interno desconhecido.	O processamento da falha devido a um erro, uma exceção ou uma falha desconhecida.
Ocorreu um erro interno desconhecido ao gravar os metadados da tarefa de exportação no bucket do S3 [nome do bucket].	O processamento da falha devido a um erro, uma exceção ou uma falha desconhecida.
A exportação do RDS falhou ao gravar os metadados da tarefa de exportação porque ela não pode assumir o perfil do IAM [ARN do perfil].	A tarefa de exportação assume seu perfil do IAM para validar se pode gravar metadados no seu bucket do S3. Se a tarefa não puder assumir seu perfil do IAM, ela falhará.

Mensagem de falha	Descrição
A exportação do RDS falhou ao gravar os metadados da tarefa de exportaçã o no bucket do S3 [nome do bucket] utilizando o perfil do IAM [ARN do perfil] com a chave do KMS [ID da chave]. Código de erro: [código de erro]	Uma ou mais permissões estão ausentes e, portanto, a tarefa de exportação não consegue acessar o bucket do S3. Essa mensagem de falha é gerada quando você recebe um dos seguintes códigos de erro: • `AWSSecurityTokenServiceException` com o código de erro `AccessDenied` • `AmazonS3Exception` com o código de erro `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` ou `KMS.DisabledException` Esses códigos de erro indicam que as configurações estão definidas incorretamente para o perfil do IAM, o bucket do S3 ou a chave do KMS.
O perfil do IAM [ARN do perfil] não está autorizado a chamar [ação do S3] no bucket do S3 [nome do bucket]. Revise suas permissões e tente novamente a exportação.	A política do IAM está incorretamente configurada. A permissão para a ação específica do S3 no bucket do S3 está ausente, o que causa falha na tarefa de exportação.
A verificação da chave do KMS falhou. Verifique as credenciais na sua chave do KMS e tente novamente.	A verificação de credenciais da chave KMS do falhou.
Falha na verificação de credenciais do S. Verifique as permissões no bucket do S3 e a política do IAM.	A verificação de credenciais do S3 falhou.
O bucket do S3 [nome do bucket] não é válido. Ele não está localizado na Região da AWS atual ou não existe. Reveja o nome do bucket do S3 e tente exportar novamente.	O bucket do S3 não é válido.

Mensagem de falha	Descrição
O bucket do S3 [nome do bucket] não está localizado na atual Região da AWS. Reveja o nome do bucket do S3 e tente exportar novamente.	O bucket da S3 está na Região da AWS incorreta.

Solucionar problemas de erros de permissões do PostgreSQL

Ao exportar bancos de dados PostgreSQL para o Amazon S3, é possível ver um erro `PERMISSIONS_DO_NOT_EXIST` informando que determinadas tabelas foram ignoradas. Esse erro geralmente ocorre quando o superusuário, especificado ao criar a instância de banco de dados, não tem permissões para acessar essas tabelas.

Para corrigir esse erro, execute o seguinte comando:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Para ter mais informações sobre privilégios de superusuário, consulte Privilégios da conta de usuário mestre.

Restaurar um cluster de banco de dados para um horário especificado

É possível restaurar seu cluster de banco de dados para um momento específico criando um novo cluster de banco de dados.

Ao restaurar um cluster de banco de dados para um momento específico, você pode escolher o grupo de segurança padrão da nuvem privada virtual (VPC). Ou você pode aplicar um grupo de segurança personalizado da VPC ao seu cluster de banco de dados.

Os clusters de banco de dados restaurados são associados automaticamente ao cluster de banco de dados e aos grupos de parâmetros de banco de dados padrão. Porém, você pode aplicar grupos de parâmetros personalizados especificando-os durante uma restauração.

O Amazon Aurora carrega continuamente os registros de log dos clusters de banco de dados no Amazon S3. Para visualizar o tempo restaurável mais recente de um cluster de banco de dados, use o comando AWS CLI describe-db-clusters e confira o valor retornado no campo `LatestRestorableTime` para o cluster de banco de dados.

É possível fazer a restauração para qualquer momento dentro do período de retenção de backup. Para consultar o tempo restaurável mais antigo de um cluster de banco de dados, use o comando AWS CLI describe-db-clusters e confira o valor retornado no campo `EarliestRestorableTime` para o cluster de banco de dados.

O período de retenção de backup do cluster de banco de dados restaurado é o mesmo do cluster de banco de dados de origem.

> **Note**
>
> As informações neste tópico se aplicam ao Amazon Aurora. Para obter informações sobre como restaurar uma instância de banco de dados do Amazon RDS, consulte Restaurar uma instância de banco de dados para um tempo especificado.
> Para ter mais informações sobre como fazer backup e restauração de um cluster de banco de dados Aurora, consulte Visão geral do backup e da restauração de um cluster de banco de dados do Aurora.
> Para o Aurora MySQL, você pode recuperar um cluster de banco de dados provisionado em um cluster de banco de dados Aurora Serverless. Para obter mais informações, consulte Restaurar um cluster de banco de dados do Aurora Serverless v1.

> Também é possível usar o AWS Backup para gerenciar backups de clusters de banco de dados do Amazon Aurora. Se o cluster de banco de dados estiver associado a um plano de backup no AWS Backup, esse plano de backup será usado para recuperação para um ponto no tempo. Para mais informações, consulte Restaurar um cluster de banco de dados para um horário especificado usando o AWS Backup.

Consulte informações sobre como restaurar um cluster de banco de dados do Aurora ou um cluster global com uma versão do Suporte estendido do RDS, consulte Restauração de um cluster de banco de dados do Aurora ou um cluster global com o Suporte estendido do Amazon RDS.

Restaure um cluster de banco de dados para um horário especificado usando um backup automatizado, um backup automatizado retido ou o AWS Backup.

Tópicos

- Restaurar um cluster de banco de dados para um ponto no tempo
- Restaurar um cluster de banco de dados em um horário especificado usando um backup automatizado retido
- Restaurar um cluster de banco de dados para um horário especificado usando o AWS Backup

Restaurar um cluster de banco de dados para um ponto no tempo

Você pode restaurar um cluster de banco de dados para um momento específico usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para restaurar um cluster de banco de dados para um horário específico

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Automated backups (Backups automatizados).

   Os backups automatizados são exibidos na guia Current Region (região atual).

3. Escolha o cluster de banco de dados do que você deseja restaurar.
4. Em Actions (Ações), escolha Restore to point in time (Restaurar para point-in-time).

   A janela Restore to point in time (Restaurar para point-in-time) é exibida.
5. Escolha Latest restorable time (Hora da última restauração) para restaurar no último horário possível ou escolha Custom (Personalizar) para escolher um horário.

   Se você escolher Custom (Personalizar), insira a data e a hora para a qual deseja restaurar o cluster.

   > **Note**
   >
   > Os horários são mostrados no fuso horário local, que é indicado por um deslocamento do Tempo Universal Coordenado (UTC). Por exemplo, UTC-5 é a Hora Padrão do Leste dos EUA/Horário de Verão Central.
6. Em Identificador do cluster de banco de dados, digite o nome do cluster de banco de dados restaurado de destino. O nome deve ser exclusivo.
7. Escolha outras opções conforme necessário, como a classe da instância de banco de dados e a configuração de armazenamento do cluster de banco de dados.

   Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.
8. Escolha Restore to point in time (Restaurar para point-in-time).

AWS CLI

Para restaurar um cluster de banco de dados para um horário especificado, use o comando restore-db-cluster-to-point-in-time da AWS CLI para criar um cluster de banco de dados.

Você pode especificar outras configurações. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

Se as tags forem fornecidas na solicitação, elas serão aplicadas ao cluster de banco de dados restaurado. Se as tags não forem fornecidas na solicitação e se o cluster de banco de dados de origem estiver ativo na região e tiver tags, o Aurora adicionará as tags mais recentes do cluster de banco de dados de origem ao cluster de banco de dados restaurado.

Example

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
       --source-db-cluster-identifier mysourcedbcluster \
       --db-cluster-identifier mytargetdbcluster \
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
       --source-db-cluster-identifier mysourcedbcluster ^
       --db-cluster-identifier mytargetdbcluster ^
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se você usar o console para restaurar um cluster de banco de dados para um momento específico, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a AWS CLI para restaurar um cluster de banco de dados para um momento específico, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados.

Chame o comando da AWS CLI create-db-instance para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados como o valor da opção `--db-cluster-identifier`.

API do RDS

Para restaurar um cluster de bancos de dados em um horário específico, chame a operação `RestoreDBClusterToPointInTime` da API do Amazon RDS com os seguintes parâmetros:

- `SourceDBClusterIdentifier`

- `DBClusterIdentifier`
- `RestoreToTime`

> **Important**
>
> Se você usar o console para restaurar um cluster de banco de dados para um momento específico, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a API do RDS para restaurar um cluster de banco de dados para um horário específico, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados.
>
> Chame a operação de API do RDS CreateDBInstance para criar a instância primária para o cluster de banco de dados. Inclua o nome de um cluster de banco de dados assim com o valor do parâmetro `DBClusterIdentifier`.

Restaurar um cluster de banco de dados em um horário especificado usando um backup automatizado retido

Você poderá restaurar um cluster de banco de dados usando um backup automatizado retido depois de excluir o cluster de banco de dados de origem, se o backup estiver dentro do período de retenção do cluster de origem. O processo é semelhante à restauração de um cluster de banco de dados que usa um backup automatizado.

> **Note**
>
> Não é possível restaurar um cluster de banco de dados do Aurora Serverless v1 usando esse procedimento, porque os backups automatizados dos clusters do Aurora Serverless v1 não são retidos.

Console

Para restaurar um cluster de banco de dados para um horário específico

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

2. No painel de navegação, escolha Automated backups (Backups automatizados).
3. Selecione a guia Retido.

4. Escolha o cluster de banco de dados do que você deseja restaurar.
5. Em Actions (Ações), escolha Restore to point in time (Restaurar para point-in-time).

   A janela Restore to point in time (Restaurar para point-in-time) é exibida.
6. Escolha Latest restorable time (Hora da última restauração) para restaurar no último horário possível ou escolha Custom (Personalizar) para escolher um horário.

   Se você escolher Custom (Personalizar), insira a data e a hora para a qual deseja restaurar o cluster.

   > **Note**
   >
   > Os horários são mostrados no fuso horário local, que é indicado por um deslocamento do Tempo Universal Coordenado (UTC). Por exemplo, UTC-5 é a Hora Padrão do Leste dos EUA/Horário de Verão Central.

7. Em Identificador do cluster de banco de dados, digite o nome do cluster de banco de dados restaurado de destino. O nome deve ser exclusivo.
8. Escolha outras opções conforme necessário, como a classe da instância de banco de dados.

   Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.
9. Escolha Restore to point in time (Restaurar para point-in-time).

AWS CLI

Para restaurar um cluster de banco de dados para um horário especificado, use o comando restore-db-cluster-to-point-in-time da AWS CLI para criar um cluster de banco de dados.

Você pode especificar outras configurações. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

A marcação de recursos é compatível com esta operação. Quando você usa a opção `--tags`, as tags do cluster de banco de dados de origem são ignoradas e as fornecidas são usadas. Caso contrário, as tags mais recentes do cluster de origem serão usadas.

Example

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE \
    --db-cluster-identifier mytargetdbcluster \
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE ^
    --db-cluster-identifier mytargetdbcluster ^
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se você usar o console para restaurar um cluster de banco de dados para um momento específico, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a AWS CLI para restaurar um cluster de banco de dados para um momento específico, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados.

Chame o comando da AWS CLI create-db-instance para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados como o valor da opção `--db-cluster-identifier`.

API do RDS

Para restaurar um cluster de bancos de dados em um horário específico, chame a operação `RestoreDBClusterToPointInTime` da API do Amazon RDS com os seguintes parâmetros:

- `SourceDbClusterResourceId`
- `DBClusterIdentifier`
- `RestoreToTime`

> **Important**
>
> Se você usar o console para restaurar um cluster de banco de dados para um momento específico, o Amazon RDS criará automaticamente a instância primária (leitura) para o cluster de banco de dados. Se você usar a API do RDS para restaurar um cluster de banco de dados para um horário específico, você deverá criar explicitamente a instância primária para o cluster de banco de dados. A instância primária é a primeira instância criada em um cluster de banco de dados.
>
> Chame a operação de API do RDS CreateDBInstance para criar a instância primária para o cluster de banco de dados. Inclua o nome de um cluster de banco de dados assim com o valor do parâmetro `DBClusterIdentifier`.

Restaurar um cluster de banco de dados para um horário especificado usando o AWS Backup

Você pode usar o AWS Backup para gerenciar backups automatizados e, em seguida, restaurá-los em um horário especificado. Para isso, você cria um plano de backup em AWS Backup e atribui o cluster de banco de dados como um recurso. Em seguida, você ativa backups contínuos para PITR na regra de backup. Para obter mais informações sobre planos e regras de backup, consulte o Guia do desenvolvedor de backup da AWS.

Habilitar backups contínuos no AWS Backup

Os backups contínuos são habilitados nas regras de backup.

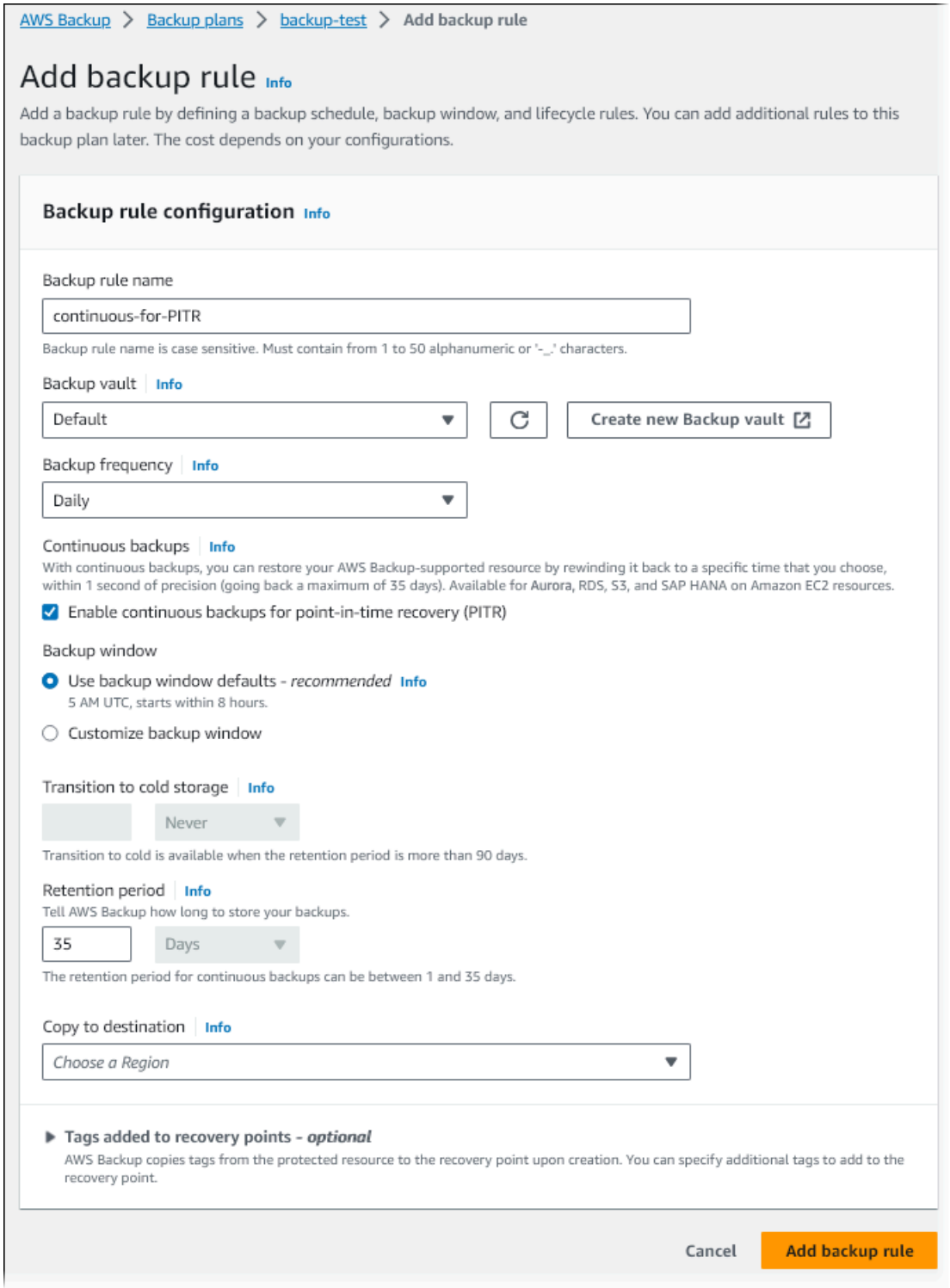
Como habilitar backups contínuos para PITR

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS Backup em https://console.aws.amazon.com/backup.
2. No painel de navegação, selecione Planos de backup.
3. Em Nome do plano de backup, selecione o plano que você usa para fazer backup do cluster de banco de dados.

4. Na seção Regras de backup, escolha Adicionar regra de backup.

   A página Adicionar regra de backup é exibida.

5. Marque a caixa de seleção Habilitar backups contínuos para recuperação para um ponto no tempo (PITR).



6. Escolha outras configurações conforme necessário e, em seguida, selecione Adicionar regra de backup.

Restaurar por meio de um backup contínuo no AWS Backup

A restauração para um ponto específico é feita por meio de um cofre de backup.

Console

É possível restaurar um cluster de banco de dados para um horário especificado.

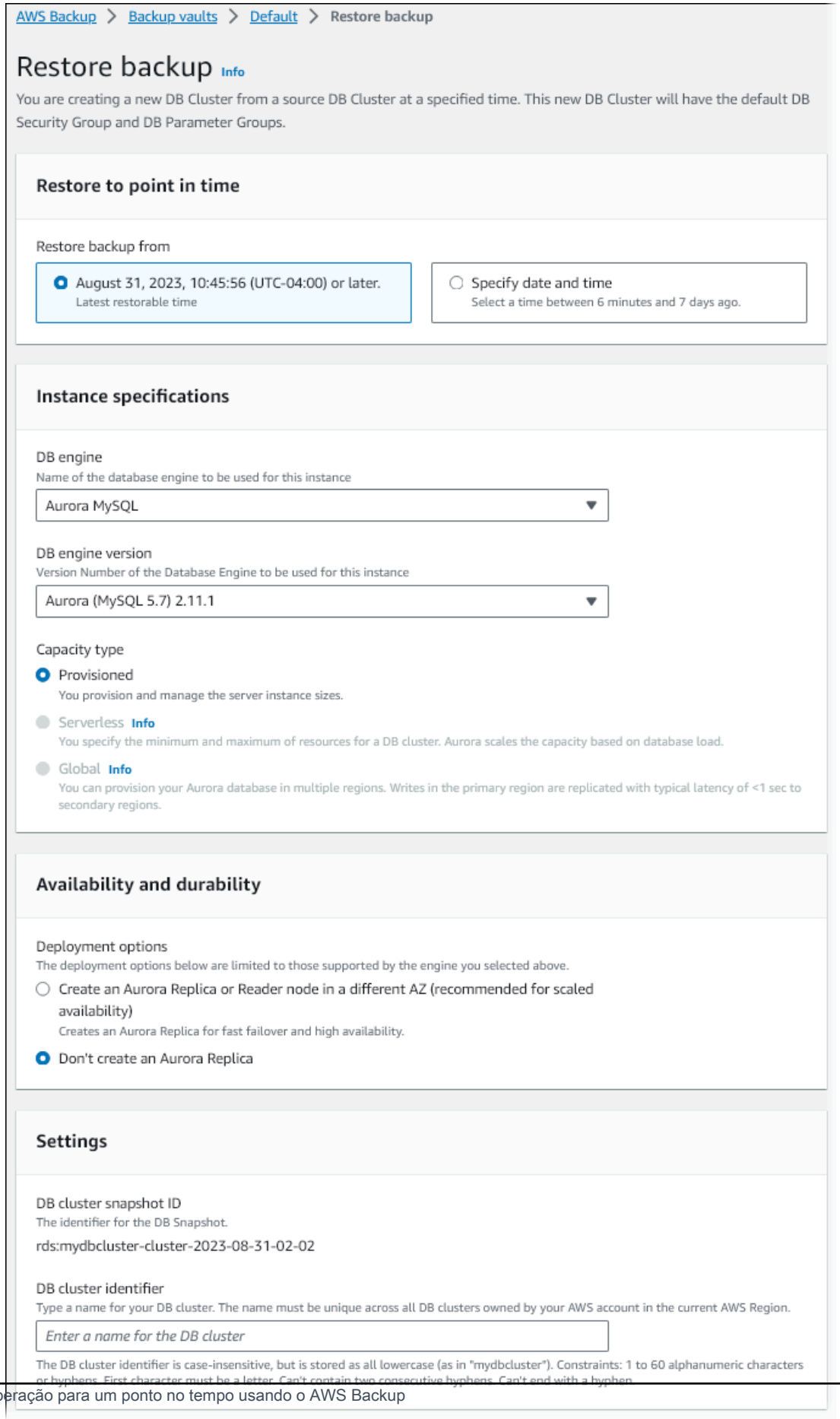
Como restaurar por meio de um backup contínuo no AWS Backup

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS Backup em https://console.aws.amazon.com/backup.
2. No painel de navegação, selecione Cofres de Backup.
3. Escolha o cofre de backup que contém o backup contínuo; por exemplo, Padrão.

   A página de detalhes do cofre de backup é exibida.
4. Em Pontos de recuperação, selecione o ponto de recuperação para o backup automatizado.

   Ele tem o tipo de backup Contínuo e um nome com `continuous:cluster-`*`AWS-Backup-job-number`*.
5. Em Ações, escolha Reiniciar.

   A página Restaurar backup é exibida.


AWS Backup > Backup vaults > Default > Restore backup
Restore backup Info
You are creating a new DB Cluster from a source DB Cluster at a specified time. This new DB Cluster will have the default DB Security Group and DB Parameter Groups.
Restore to point in time
Restore backup from
August 31, 2023, 10:45:56 (UTC-04:00) or later.
Latest restorable time
Specify date and time
Select a time between 6 minutes and 7 days ago.
Instance specifications
DB engine
Name of the database engine to be used for this instance
Aurora MySQL
DB engine version
Version Number of the Database Engine to be used for this instance
Aurora (MySQL 5.7) 2.11.1
Capacity type
Provisioned
You provision and manage the server instance sizes.
Serverless Info
You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.
Global Info
You can provision your Aurora database in multiple regions. Writes in the primary region are replicated with typical latency of <1 sec to secondary regions.
Availability and durability
Deployment options
The deployment options below are limited to those supported by the engine you selected above.
Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
Don't create an Aurora Replica
Settings
DB cluster snapshot ID
The identifier for the DB Snapshot.
rds:mydbcluster-cluster-2023-08-31-02-02
DB cluster identifier
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.
Enter a name for the DB cluster
The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

6. Para Restaurar para um ponto no tempo, selecioneEspecificar data e hora para restaurar para um horário específico.
7. Escolha outras configurações conforme necessário para restaurar o cluster de banco de dados e, em seguida, escolha Restaurar backup.

   A página Tarefas é exibida, mostrando o painel Tarefas de restauração. Uma mensagem na parte superior da página fornece informações sobre o trabalho de restauração.

Depois que o cluster de banco de dados for restaurado, adicione a instância de banco de dados primária (gravadora) a ele. Chame o comando da AWS CLI create-db-instance para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados assim com o valor do parâmetro `--db-cluster-identifier`.

CLI

Use o comando start-restore-job da AWS CLI para restaurar o cluster de banco de dados para um horário especificado. Os seguintes parâmetros são obrigatórios:

- `--recovery-point-arn`: o nome do recurso da Amazon (ARN) referente ao ponto de recuperação do qual se deve restaurar.
- `--resource-type`: use `Aurora`.
- `--iam-role-arn`: o ARN referente ao perfil do IAM que você usa para operações AWS Backup.
- `--metadata`: os metadados que você usa para restaurar o cluster de banco de dados. Os seguintes parâmetros são obrigatórios:
  - `DBClusterIdentifier`
  - `Engine`
  - `RestoreToTime` ou `UseLatestRestorableTime`

O exemplo a seguir mostra como restaurar um cluster de banco de dados para um horário especificado.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
```

```
--metadata '{"DBClusterIdentifier":"backup-pitr-test","Engine":"aurora-
mysql","RestoreToTime":"2023-09-01T17:00:00.000Z"}'
```

O exemplo a seguir mostra como restaurar um cluster de banco de dados para o último momento restaurável.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-latest","Engine":"aurora-
mysql","UseLatestRestorableTime":"true"}'
```

Depois que o cluster de banco de dados for restaurado, adicione a instância de banco de dados primária (gravadora) a ele. Chame o comando da AWS CLI create-db-instance para criar a instância primária do seu cluster de banco de dados. Inclua o nome de um cluster de banco de dados assim com o valor do parâmetro `--db-cluster-identifier`.

Excluir um snapshot de cluster de banco de dados

É possível excluir snapshots de cluster de banco de dados gerenciados pelo Amazon RDS quando eles não são mais necessários.

> **Note**
>
> Para excluir backups gerenciados pelo AWS Backup, use o console do AWS Backup. Para obter mais informações sobre o AWS Backup, consulte o Guia do desenvolvedor do AWS Backup.

Excluir um snapshot de cluster de banco de dados

É possível excluir um snapshot de cluster de banco de dados usando o console, a AWS CLI ou a API do RDS.

Para excluir um snapshot compartilhado ou público, você deve fazer login na conta da AWS que tem esse snapshot.

Console

Para excluir um snapshot de cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Snapshots.
3. Escolha o snapshot de cluster de banco de dados que você deseja excluir.
4. Em Actions (Ações), selecione Delete Snapshot (Excluir snapshot).
5. Escolha Delete (Excluir) na página de confirmação.

AWS CLI

É possível excluir um snapshot de cluster de banco de dados usando o comando da AWS CLI Delete-DB-cluster-snapshot.

As seguintes opções são usadas para excluir um snapshot de cluster de banco de dados.

- `--db-cluster-snapshot-identifier` – O identificador do snapshot de cluster de banco de dados.

Example

O código a seguir exclui o snapshot de cluster de banco de dados `mydbclustersnapshot`.

Para Linux, macOS ou Unix:

```
aws rds delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Para Windows:

```
aws rds delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API do RDS

É possível excluir um snapshot de cluster de banco de dados usando a operação DeleteDBClusterSnapshot da API do Amazon RDS.

Os seguintes parâmetros são usados para excluir um snapshot de cluster de banco de dados.

- `DBClusterSnapshotIdentifier`: o identificador do snapshot de cluster de banco de dados.

Tutorial: Restaurar um cluster de banco de dados do Amazon Aurora de um snapshot de cluster de banco de dados

Um cenário comum ao trabalhar com o Amazon Aurora é ter uma instância de banco de dados com a qual você trabalha ocasionalmente, mas que não precisa usar em tempo integral. Por exemplo, você pode usar um cluster de banco de dados para armazenar os dados de um relatório que você executa somente trimestralmente. Uma maneira de economizar dinheiro nesse cenário é obter um snapshot do cluster de banco de dados após a conclusão do relatório. Depois, você exclui o cluster de banco de dados e o restaura quando precisar carregar novos dados e executar o relatório durante o próximo trimestre.

Ao restaurar um cluster de banco de dados, você fornece o nome do snapshot do cluster de banco de dados do qual deseja restaurar. Depois, você fornece um nome para o novo cluster de banco de dados criado na operação de restauração. Para obter mais informações sobre a restauração de clusters de banco de dados de snapshots, consulte Restauração de um snapshot de um cluster de banco de dados.

Neste tutorial, também atualizamos o cluster de banco de dados restaurado do Aurora MySQL versão 2 (compatível com o MySQL 5.7) para o Aurora MySQL versão 3 (compatível com o MySQL 8.0).

Restaure um cluster de banco de dados para um horário especificado de um snapshot do cluster de banco de dados usando o console do Amazon RDS ou a AWS CLI.

Para ter informações sobre o gerenciamento de chaves do AWS KMS para o Amazon RDS, consulte AWS KMS keyGerenciamento de.

Tópicos

- Tutorial: Restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando o console do Amazon RDS
- Tutorial: Restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando a AWS CLI

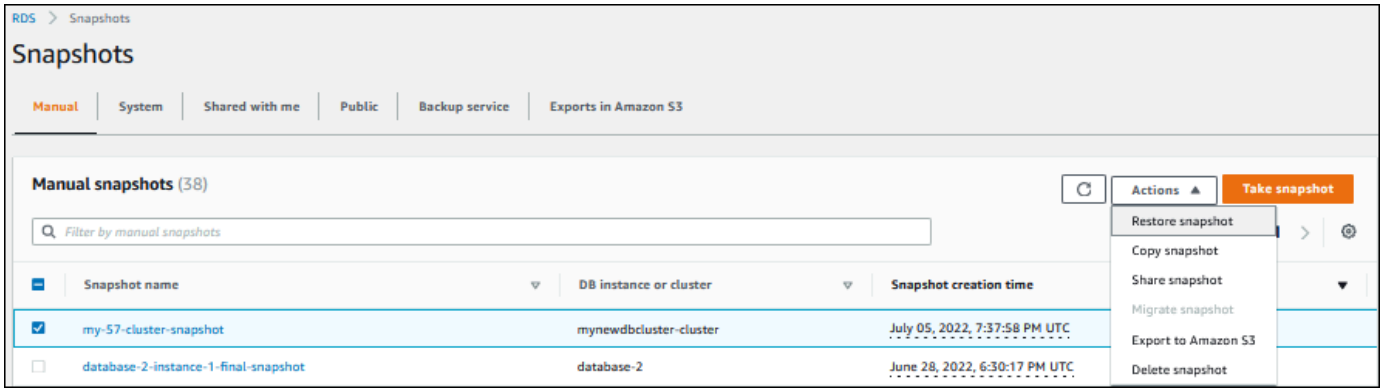
Tutorial: Restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando o console do Amazon RDS

Neste tutorial, você vai restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando o console do Amazon RDS. Quando você restaura um cluster de banco de dados de um snapshot usando o Console de gerenciamento da AWS, a instância de banco de dados primária (gravadora) também é criada.

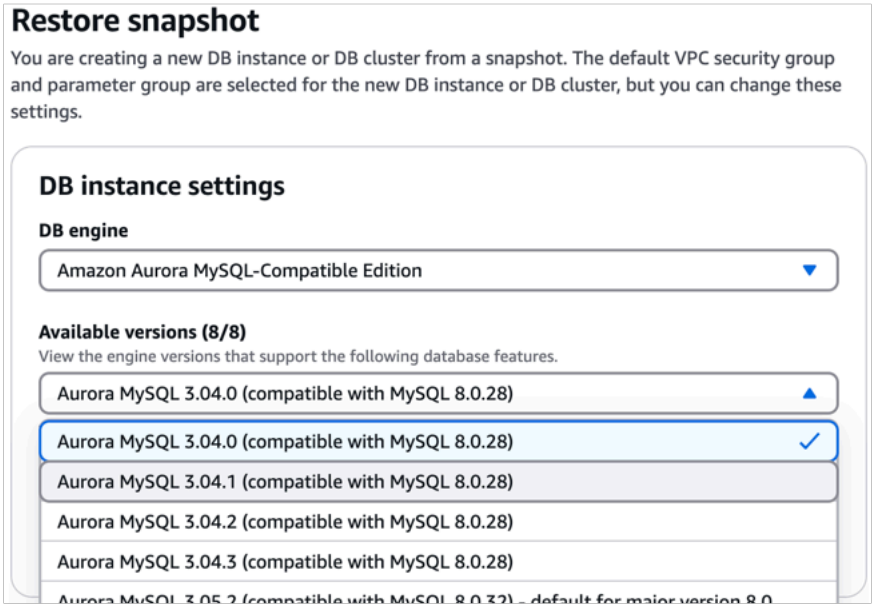
> **Note**
>
> Enquanto a instância de banco de dados primária está sendo criada, ela aparece como uma instância de leitura, mas se torna uma instância de gravação após a criação.

Para restaurar um cluster de banco de dados de um snapshot do cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Snapshots.
3. Escolha o snapshot do cluster de banco de dados a partir do qual você deseja restaurar.
4. Em Actions (Ações), escolha Restore snapshot (Restaurar snapshot).



   A página Restore snapshot (Restaurar snapshot) é exibida.
5. Em DB instance settings (Configurações da instância de banco de dados), faça o seguinte:
   a. Use a configuração padrão para DB engine (Mecanismo de banco de dados).
   b. Para Versões disponíveis, escolha uma versão compatível com MySQL 8.0, como Aurora MySQL 3.04.0 (compatível com MySQL 8.0.28).



6. Em Settings (Configurações), para DB instance identifier (Identificador de instância de banco de dados), insira o nome exclusivo que você quer usar na instância de banco de dados restaurada, por exemplo, **`my-80`**.

   > **Note**
   >
   > Para criar o identificador do cluster de banco de dados, o Amazon RDS anexa `-cluster` ao identificador de instância de banco de dados que você especificar.

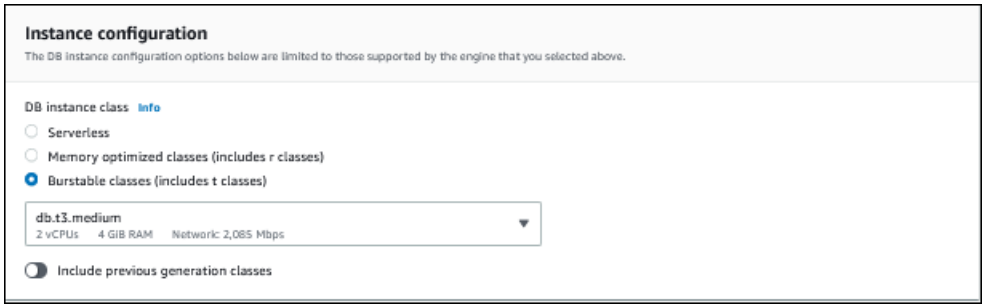
7. Em Connectivity (Conectividade, use as configurações padrão para o seguinte:

   - Nuvem privada virtual (VPC)
   - DB subnet group (Grupo de subredes do banco de dados)
   - Acesso público
   - VPC security group (firewall) [Grupo de segurança da VPC (firewall)]

8. Escolha a DB instance class (Classe da instância de banco de dados)

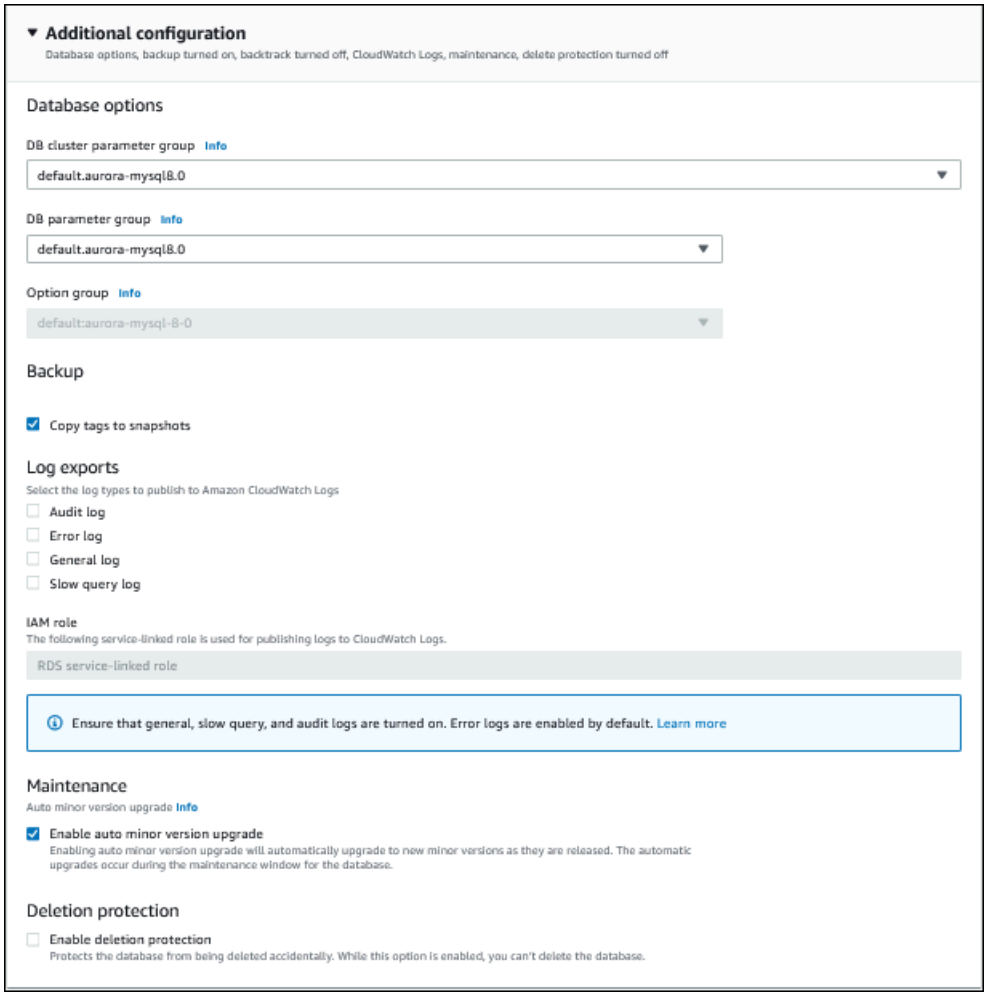
   Para este tutorial, escolha Burstable classes (includes t classes) [Classes com capacidade de intermitência (inclui classes t)], depois escolha db.t3.medium.

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.



9. Para Database authentication (Autenticação do banco de dados), use a configuração padrão.
10. Para Encryption (Criptografia), use as configurações padrão.

    Se a fonte do cluster de banco de dados do snapshot tiver sido criptografado, o cluster de banco de dados restaurado também será criptografado. Não é possível fazer isso sem criptografia.
11. Expanda Additional configuration (Configuração adicional) na parte inferior da página.



12. Faça as seguintes escolhas:

    a. Para este tutorial, use o valor padrão para DB cluster parameter group (Grupo de parâmetros do cluster de banco de dados).

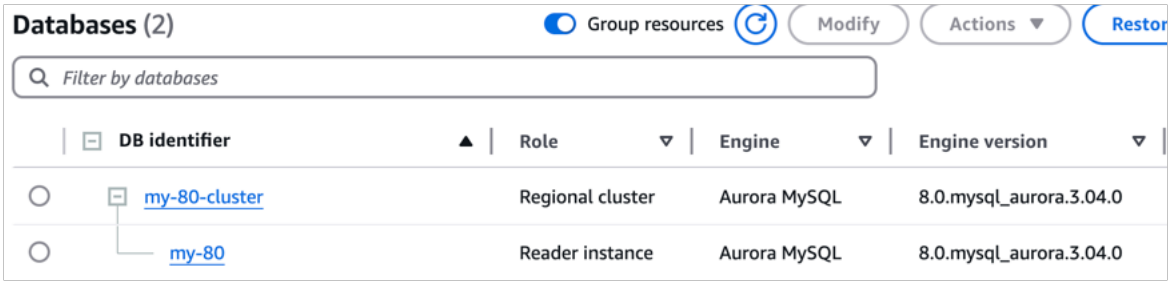
    b. Para este tutorial, use o valor padrão para DB parameter group (Grupo de parâmetros do banco de dados).

    c. Para Log exports (Exportações de logs), marque todas as caixas de seleção.

    d. Para Deletion protection (Proteção contra exclusão), escolha Enable deletion protection (Habilitar proteção contra exclusão).

13. Escolha Restore DB Instance.

A página Databases (Banco de dados) exibe o cluster de banco de dados restaurado, com um status de `Creating`.



Enquanto a instância de banco de dados primária está sendo criada, ela aparece como uma instância de leitura, mas se torna uma instância de gravação após a criação.

Tutorial: Restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando a AWS CLI

Neste tutorial, você vai restaurar um cluster de banco de dados por meio de um snapshot do cluster de banco de dados usando a AWS CLI. A restauração de um cluster de banco de dados de um snapshot usando a AWS CLI acontece em duas etapas:

1. Restaurar o cluster de banco de dados usando o comando restore-db-cluster-from-snapshot
2. Criar a instância de banco de dados primária (gravadora) usando o comando create-db-instance

Restaurar o cluster de banco de dados

Use o comando `restore-db-cluster-from-snapshot`. São necessárias as seguintes opções:

- `--db-cluster-identifier`: o nome do cluster de banco de dados restaurado.
- `--snapshot-identifier`: o nome do snapshot de banco de dados do qual deseja restaurar.
- `--engine`: o mecanismo de banco de dados do cluster de banco de dados restaurado. Deve ser compatível com o mecanismo de banco de dados do cluster de banco de dados de origem.

  As opções são as seguintes:

  - `aurora-mysql`: compatível com Aurora MySQL 5.7 e 8.0.
  - `aurora-postgresql`: compatível com Aurora PostgreSQL.

  Neste exemplo, usamos `aurora-mysql`.
- `--engine-version`: a versão do cluster de banco de dados restaurado. Neste exemplo, usamos uma versão compatível com MySQL 8.0.

O exemplo a seguir restaura um cluster de banco de dados compatível com o Aurora MySQL 8.0 chamado `my-new-80-cluster` de um snapshot de cluster de banco de dados denominado `my-57-cluster-snapshot`.

Como restaurar o cluster de banco de dados

- Use um dos seguintes comandos.

  Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-80-cluster \
    --snapshot-identifier my-57-cluster-snapshot \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.02.0
```

  Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier my-new-80-cluster ^
    --snapshot-identifier my-57-cluster-snapshot ^
    --engine aurora-mysql ^
    --engine-version 8.0.mysql_aurora.3.02.0
```

A saída será semelhante à seguinte.

```
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "eu-central-1b",
            "eu-central-1c",
            "eu-central-1a"
        ],
        "BackupRetentionPeriod": 14,
        "DatabaseName": "",
        "DBClusterIdentifier": "my-new-80-cluster",
        "DBClusterParameterGroup": "default.aurora-mysql8.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
```

```
        "Endpoint": "my-new-80-cluster.cluster-############.eu-
central-1.rds.amazonaws.com",
        "ReaderEndpoint": "my-new-80-cluster.cluster-ro-############.eu-
central-1.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "Port": 3306,
        "MasterUsername": "admin",
        "PreferredBackupWindow": "01:55-02:25",
        "PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
        "ReadReplicaIdentifiers": [],
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "HostedZoneId": "Z1RLNU0EXAMPLE",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/#######-5ccc-49cc-8aaa-
############",
        "DbClusterResourceId": "cluster-ZZ12345678ITSJUSTANEXAMPLE",
        "DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:my-new-80-
cluster",
        "AssociatedRoles": [],
        "IAMDatabaseAuthenticationEnabled": false,
        "ClusterCreateTime": "2022-07-05T20:45:42.171000+00:00",
        "EngineMode": "provisioned",
        "DeletionProtection": false,
        "HttpEndpointEnabled": false,
        "CopyTagsToSnapshot": false,
        "CrossAccountClone": false,
        "DomainMemberships": [],
        "TagList": []
    }
}
```

Criar a instância de banco de dados primária (gravadora)

Para criar a instância de banco de dados primária (gravadora), use o comando `create-db-instance`. São necessárias as seguintes opções:

- `--db-cluster-identifier`: o nome do cluster de banco de dados restaurado.
- `--db-instance-identifier`: o nome da instância de banco de dados primária.
- `--db-instance-class`: o nome da classe da instância de banco de dados primária. Neste exemplo, usamos `db.t3.medium`.

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.

- `--engine`: o mecanismo de banco de dados da instância de banco de dados primária. Ele deve ser o mesmo mecanismo de banco de dados usado pelo cluster de banco de dados restaurado.

  As opções são as seguintes:

  - `aurora-mysql`: compatível com Aurora MySQL 5.7 e 8.0.
  - `aurora-postgresql`: compatível com Aurora PostgreSQL.

  Neste exemplo, usamos `aurora-mysql`.

O exemplo a seguir cria uma instância de banco de dados primária (gravadora) chamada `my-new-80-cluster-instance` no cluster de banco de dados restaurado compatível com o Aurora MySQL 8.0 denominado `my-new-80-cluster`.

Como criar a instância de banco de dados primária

- Use um dos seguintes comandos.

  Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-80-cluster \
    --db-instance-identifier my-new-80-cluster-instance \
    --db-instance-class db.t3.medium \
    --engine aurora-mysql
```

  Para Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-80-cluster ^
    --db-instance-identifier my-new-80-cluster-instance ^
    --db-instance-class db.t3.medium ^
    --engine aurora-mysql
```

A saída será semelhante à seguinte.


```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-new-80-cluster-instance",
        "DBInstanceClass": "db.t3.medium",
        "Engine": "aurora-mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 1,
        "PreferredBackupWindow": "01:55-02:25",
        "BackupRetentionPeriod": 14,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.aurora-mysql8.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2305ca49",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1a"
```

```
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1b"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1c"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sat:02:41-sat:03:11",
        "PendingModifiedValues": {},
        "MultiAZ": false,
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:aurora-mysql-8-0",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": false,
        "StorageType": "aurora",
        "DbInstancePort": 0,
        "DBClusterIdentifier": "my-new-80-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:534026745191:key/#######-5ccc-49cc-8aaa-
############",
        "DbiResourceId": "db-5C6UT5PU0YETANOTHEREXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
```

```
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:eu-central-1:123456789012:db:my-new-80-cluster-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": [],
        "TagList": []
    }
}
```

Monitorar métricas em um cluster do Amazon Aurora

O Amazon Aurora usa um cluster de servidores de banco de dados replicados. Normalmente, monitorar um cluster do Aurora requer a verificação da integridade de várias instâncias de banco de dados. As instâncias podem ter funções especializadas, lidando em sua maioria com operações de gravação, operações somente leitura ou uma combinação das duas. Você também monitora a integridade geral do cluster medindo o atraso da replicação. Esse é o tempo para que as alterações feitas por uma instância de banco de dados estejam disponíveis para as outras instâncias.

Tópicos

- Plano de monitoramento
- Linha de base de performance
- Orientações de performance
- Ferramentas de monitoramento do Amazon Aurora
- Visualizar o status da instância
- Recomendações do Amazon Aurora
- Visualizar métricas no console do Amazon RDS
- Visualizar métricas combinadas com o painel do Insights de Performance
- Monitorar métricas do Amazon Aurora com o Amazon CloudWatch
- Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights
- Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora
- Analisar anomalias de desempenho do Aurora com o Amazon DevOps Guru para Amazon RDS
- Monitorar métricas do SO com o monitoramento avançado
- Referência de métricas do Amazon Aurora

Plano de monitoramento

Antes de iniciar o monitoramento , crie um plano de monitoramento. Esse plano deve responder às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?

- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

Linha de base de performance

Para atingir suas metas de monitoramento, é necessário estabelecer uma linha de base. Para fazer isso, meça a performance em diferentes condições de carga em vários momentos em seu ambiente do Amazon Aurora. É possível monitorar métricas como as seguintes:

- Taxa de transferência na rede
- Conexões de cliente
- E/S para operações de leitura, gravação ou metadados
- Saldos de crédito intermitentes para suas instâncias de banco de dados

Recomendamos armazenar dados históricos de performance para o Amazon Aurora. Usando os dados armazenados, é possível comparar a performance atual com as tendências anteriores. Também é possível distinguir padrões de performance normais de anomalias e criar técnicas para resolver problemas.

Orientações de performance

Em geral, os valores aceitáveis para as métricas de performance dependem do que a aplicação está fazendo em relação à sua linha de base. Investigue variações consistentes ou tendenciais de sua linha de base. Muitas vezes, as métricas a seguir são a origem dos problemas de performance:

- Alto consumo de CPU ou RAM – valores altos para o consumo de CPU ou RAM podem ser adequados, se estiverem de acordo com seus objetivos em relação ao aplicativo (como taxa de transferência ou concorrência).
- Consumo de espaço em disco – inspecione o consumo de espaço em disco caso o espaço usado seja consistentemente igual ou superior a 85% do espaço total no disco. Veja se é possível excluir dados da instância ou arquivar dados em um sistema diferente para liberar mais espaço.
- Tráfego de rede – em relação ao tráfego de rede, fale com o administrador do sistema para entender qual taxa de transferência é esperada para sua rede de domínio e conexão com a

Internet. Inspecione o tráfego de rede caso a taxa de transferência seja consistentemente menor do que a esperada.

- Conexões do banco de dados: se você observar números elevados de conexões de usuários e também diminuições na performance da instância e no tempo de resposta, considere restringir as conexões do banco de dados. O melhor número de conexões de usuários para sua instância de banco de dados varia conforme a classe da instância e a complexidade das operações em execução. Para determinar o número de conexões de banco de dados, associe sua instância de banco de dados a um grupo de parâmetros cujo parâmetro `User Connections` esteja definido como um valor diferente de 0 (ilimitado). Você pode usar um parameter group existente ou criar um novo. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.
- Métricas de IOPS – os valores esperados para as métricas de IOPS dependem da especificação do disco e da configuração do servidor, por isso, use sua linha de base para saber os valores típicos. Inspecione caso os valores sejam consistentemente diferentes da sua linha de base. Para obter a melhor performance de IOPS, confira se o seu conjunto de trabalho típico se adéqua à memória para minimizar as operações de leitura e gravação.

Quando a performance estiver fora da linha de base estabelecida, talvez seja necessário fazer alterações para otimizar a disponibilidade do banco de dados para sua workload. Por exemplo, talvez você precise alterar a classe de sua instância de banco de dados. Ou talvez seja necessário alterar o número de instâncias de banco de dados e réplicas de leitura disponíveis para clientes.

Ferramentas de monitoramento do Amazon Aurora

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do Amazon Aurora e suas outras soluções da AWS. A AWS fornece várias ferramentas de monitoramento para observar o Amazon Aurora, gerar relatórios quando algo estiver errado e executar ações automáticas quando for apropriado.

Tópicos

- Ferramentas de monitoramento automatizadas
- Ferramentas de monitoramento manual

Ferramentas de monitoramento automatizadas

Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Tópicos

- Status e recomendações de cluster do Amazon Aurora
- Métricas do Amazon CloudWatch para Amazon Aurora
- Insights de Performance do Amazon RDS e monitoramento do sistema operacional
- Serviços integrados

Status e recomendações de cluster do Amazon Aurora

É possível usar as seguintes ferramentas automatizadas para supervisionar o Amazon Aurora e gerar relatórios quando algo estiver errado:

- Status do cluster do Amazon Aurora: exibe detalhes sobre o status atual do cluster usando o console do Amazon RDS, a AWS CLI ou a API do RDS.
- Amazon Aurora recomendações — responda a recomendações automatizadas para recursos de banco de dados, como instâncias de banco de dados, clusters de banco de dados, e grupo de parâmetros de cluster de banco de dados. Para obter mais informações, consulte Recomendações do Amazon Aurora.

Métricas do Amazon CloudWatch para Amazon Aurora

O Amazon Aurora integra-se ao Amazon CloudWatch para oferecer recursos adicionais de monitoramento.

- Amazon CloudWatch: esse serviço monitora seus recursos da AWS e as e aplicações executadas na AWS em tempo real. É possível usar os seguintes recursos do Amazon CloudWatch com o Amazon Aurora:
  - Métricas do Amazon CloudWatch: o Amazon Aurora envia métricas automaticamente para o CloudWatch a cada minuto, para cada banco de dados ativo. Você não recebe cobranças adicionais para métricas do Amazon RDS no CloudWatch. Para obter mais informações, consulte Métricas do Amazon CloudWatch para o Amazon Aurora
  - Alarmes do Amazon CloudWatch– Você pode assistir a uma única métrica do Amazon Aurora em um período de tempo específico. Depois, você pode realizar uma ou mais ações com base no valor da métrica em relação a um limite definido.

Insights de Performance do Amazon RDS e monitoramento do sistema operacional

É possível usar as seguintes ferramentas automatizadas para monitorar a performance do Amazon Aurora:

- Amazon RDS Performance Insights: avalie a carga no banco de dados e determine quando e onde tomar medidas. Para obter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.
- Monitoramento avançado do Amazon RDS: observe as métricas em tempo real para o sistema operacional. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.

Serviços integrados

Os seguintes serviços da AWS estão integrados ao Amazon Aurora:

- O Amazon EventBridge é um serviço de barramento de eventos sem servidor que facilita a conexão de aplicações a dados de diversas origens. Para obter mais informações, consulte Monitorar eventos do Amazon Aurora.

- O Amazon CloudWatch Logs permite monitorar, armazenar e acessar os arquivos de log de instâncias do Amazon Aurora, do CloudTrail e de outras fontes. Para obter mais informações, consulte Monitorar arquivos de log do Amazon Aurora.
- O AWS CloudTrail captura chamadas de API e eventos relacionados feitos por sua conta da Conta da AWS ou em nome dela e entrega os arquivos de log a um bucket do Amazon S3 que você especificar. Para obter mais informações, consulte Monitorar chamadas de API do Amazon Aurorano AWS CloudTrail.
- O Database Activity Streams é um recurso do Amazon Aurora que fornece uma transmissão quase em tempo real da atividade em seu cluster de banco de dados . Para obter mais informações, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.
- O DevOps Guru for Amazon RDS é um recurso do Amazon DevOps Guru que aplica machine learning a métricas do Performance Insights para bancos de dados Amazon Aurora. Para obter mais informações, consulte Analisar anomalias de desempenho do Aurora com o Amazon DevOps Guru para Amazon RDS.

Ferramentas de monitoramento manual

É necessário monitorar manualmente os itens que os alarmes do CloudWatch não cobrem. Os painéis do console do Amazon RDS, do CloudWatch do AWS Trusted Advisor e outros painéis do console AWS apresentam uma visão rápida do estado do ambiente da AWS. Recomendamos que você também verifique os arquivos de log de sua instância de banco de dados.

- Na console do Amazon RDS, é possível monitorar os seguintes itens de seus recursos:
  - O número de conexões a uma instância de banco de dados
  - A quantidade de operações de leitura e gravação em uma instância de banco de dados
  - A quantidade de armazenamento que uma instância de banco de dados está utilizando
  - A quantidade de memória e CPU em uso para uma instância de banco de dados
  - A quantidade de tráfego de rede de e para uma instância de banco de dados
- No painel do Trusted Advisor, você pode rever as seguintes verificações de otimização de custos, segurança, tolerância a falhas e melhoria de performance:
  - Amazon RDS Idle DB Instances
  - Amazon RDS Security Group Access Risk
  - Amazon RDS Backups
  - Amazon RDS Multi-AZ

- Acessibilidade da instância de bancos de dados Aurora

  Para obter mais informações sobre essas verificações, consulte Melhores práticas do Trusted Advisor (verificações).

- A página inicial do CloudWatch mostra:
  - Alertas e status atual
  - Gráficos de alertas e recursos
  - Estado de integridade do serviço

  Além disso, é possível usar o CloudWatch para fazer o seguinte:
  - Criar painéis personalizados para monitorar os serviços de seu interesse.
  - Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências.
  - Pesquise e procure todas as métricas de recursos da AWS.
  - Criar e editar alertas para ser notificado sobre problemas.

Visualizar o status da instância

Usando o console do Amazon RDS, é possível acessar rapidamente o status do cluster do banco de dados.

Tópicos

- Visualizar um cluster de bancos de dados Amazon Aurora
- Visualizar o status do cluster do banco de dados
- Visualizar o status de uma instância de banco de dados em um cluster do Aurora

Visualizar um cluster de bancos de dados Amazon Aurora

Você tem várias opções para exibir informações sobre seus clusters de bancos de dados Amazon Aurora e as instâncias de banco de dados nesses clusters.

- Você pode visualizar clusters de banco de dados e instâncias de banco de dados no console do Amazon RDS escolhendo Databases (Bancos de dados) no painel de navegação.
- Você pode obter informações sobre os clusters e as instâncias de banco de dados usando a AWS Command Line Interface (AWS CLI).
- Você pode obter informações de clusters e instâncias de banco de dados usando a API do Amazon RDS.

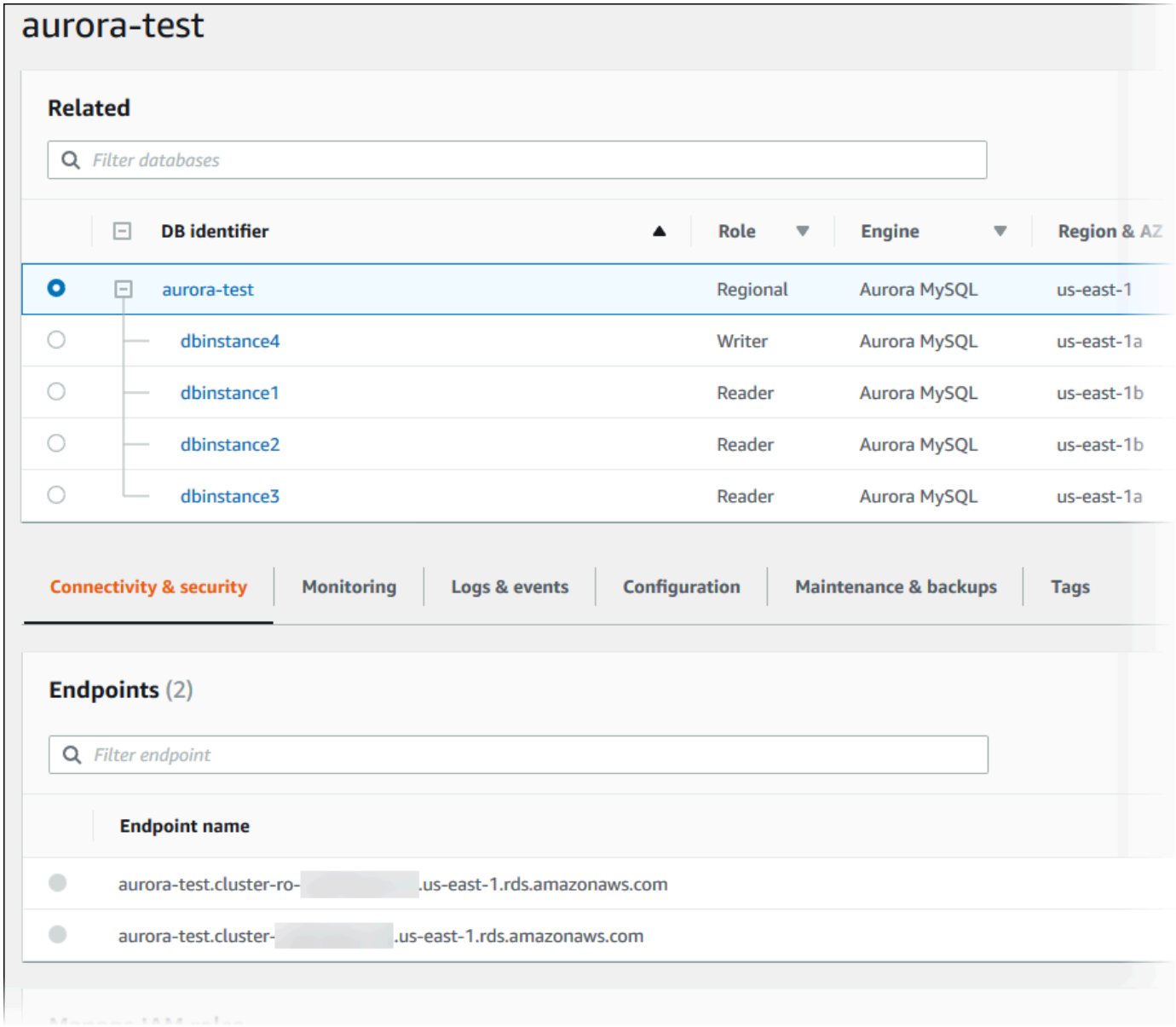
Console

No console Amazon RDS, você pode ver detalhes sobre um cluster de banco de dados escolhendo Databases (Bancos de dados) no painel de navegação do console. Também é possível ver detalhes sobre instâncias de banco de dados que são membros de um cluster de banco de dados Amazon Aurora.

Como visualizar ou modificar clusters de banco de dados no console do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o nome do cluster de banco de dados Aurora que você deseja visualizar na lista.

   Por exemplo, a imagem a seguir mostra a página de detalhes do cluster de banco de dados chamado `aurora-test`. O cluster de banco de dados tem quatro instâncias de bancos de dados mostradas na lista DB identifier (Identificador de banco de dados). Esses requisitos de atualização ainda serão aplicáveis mesmo que você desative o retrocesso para o cluster do `dbinstance4` 1.*.



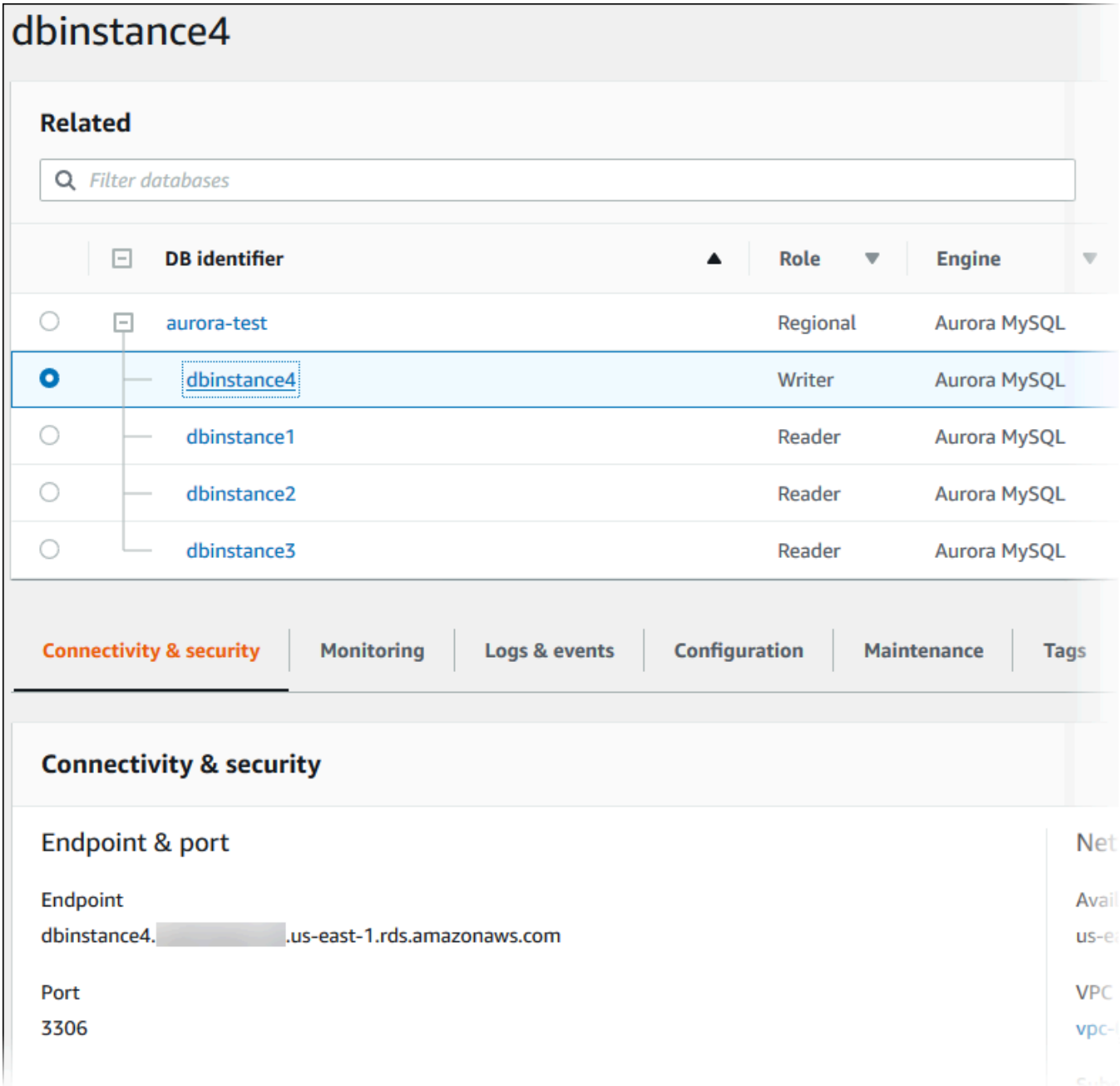
4. Para modificar um cluster de banco de dados, selecione-o na lista e escolha Modify (Modificar).

Como visualizar ou modificar instâncias de banco de dados de um cluster de banco de dados no console do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Execute um destes procedimentos:

- Para visualizar uma instância de banco de dados, escolha uma da lista que seja membro do cluster de banco de dados Aurora.

  Por exemplo, se você escolher o identificador de instância de banco de dados `dbinstance4`, o console exibirá a página de detalhes da instância de banco de dados `dbinstance4`, como mostra a imagem a seguir.



- Para modificar uma instância de banco de dados, escolha a instância de banco de dados na lista e escolha Modify (Modificar). Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

AWS CLI

Para visualizar as informações do cluster de banco de dados usando a AWS CLI, use o comando describe-db-clusters. Por exemplo, o seguinte comando da AWS CLI lista as informações de cluster de banco de dados para todos os clusters de banco de dados na região `us-east-1` de modificação para a conta configurada da AWS.

```
aws rds describe-db-clusters --region us-east-1
```

O comando retornará a seguinte saída se a sua AWS CLI estiver configurada para saída JSON.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-
east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
```

```
                {
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "aurora-sample.cluster-123456789012.us-
east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
```

```
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API do RDS

Para visualizar as informações do cluster de banco de dados usando a API do Amazon RDS, use a operação DescribeDBClusters.

Visualizar o status do cluster do banco de dados

O status de um cluster de banco de dados indica seu estado operacional atual. É possível visualizar o status de um cluster de banco de dados e as instâncias do cluster usando o console, a AWS CLI ou a API do Amazon RDS.

> **Note**
>
> O Aurora também usa outro status chamado status de manutenção, que é mostrado na coluna Manutenção do console do Amazon RDS. Esse valor indica o status de todos os patches de manutenção que precisam ser aplicados a um cluster de banco de dados. O status da manutenção é independente do status do cluster de banco de dados. Para ter mais informações sobre o status de manutenção, consulte Aplicar atualizações para um clusterde banco de dados.

Encontre os valores possíveis do status de clusters de banco de dados na tabela a seguir.

Status do cluster de banco de dados	Faturado	Descrição
disponíveis	Faturado	O cluster de banco de dados está disponíve l para modificações. Quando um cluster do Aurora Serverless está disponível e pausado, você será cobrado somente pelo armazenam ento.
Backing-up	Faturado	O backup do cluster de banco de dados está sendo executado no momento.
Backtracking	Faturado	O retorno do cluster de banco de dados está sendo executado no momento. Esse status se aplica somente ao Aurora MySQL.
Cloning-failed	Não faturado	Houve falha na clonagem de um cluster de banco de dados.

Status do cluster de banco de dados	Faturado	Descrição
Criando	Não faturado	O cluster de banco de dados está sendo criado. O cluster de banco de dados fica inacessível enquanto está sendo criado.
Excluindo	Não faturado	O cluster de banco de dados está sendo excluído.
Failing-over	Faturado	Um failover da instância primária para uma réplica do Aurora está sendo executado.
Inaccessible-encryption-credentials	Não faturado	A AWS KMS key usada para criptografar ou descriptografar o cluster de banco de dados não pode ser acessada nem recuperada.
Inaccessible-encryption-credentials-recoverable	Faturado para armazenamento	A chave do KMS usada para criptografar ou descriptografar o cluster de banco de dados não pode ser acessada. No entanto, se a chave do KMS estiver ativa, reiniciar o cluster de banco de dados poderá recuperá-la. Para ter mais informações, consulte Criptografar um cluster de banco de dados do Amazon Aurora.
Manutenção	Faturado	O Amazon RDS está aplicando uma atualização de manutenção no cluster de banco de dados. Esse status é usado para a manutenção em nível de cluster de banco de dados que o RDS programa com antecedência.
Migrating	Faturado	Um snapshot do cluster de banco de dados está sendo restaurado em um cluster de banco de dados.
Migration-failed	Não faturado	Ocorreu uma falha na migração.

Status do cluster de banco de dados	Faturado	Descrição
Modificar	Faturado	O cluster de banco de dados está sendo modificado devido a uma solicitação do cliente para modificá-lo.
Promoting	Faturado	Uma réplica de leitura está sendo promovida para um cluster de banco de dados autônomo.
Preparing-data-migration	Faturado	O Amazon RDS está se preparando para migrar dados para o Aurora.
Renomeação	Faturado	O cluster de banco de dados está sendo renomeado devido a uma solicitação do cliente para renomeá-lo.
Resetting-master-credentials	Faturado	As credenciais principais do cluster de banco de dados estão sendo redefinidas devido a uma solicitação do cliente para redefini-las.
Starting	Faturado para armazenamento	O cluster de banco de dados está iniciando.
Interrompido	Faturado para armazenamento	O cluster de banco de dados foi interrompido.
Parando	Faturado para armazenamento	O cluster de banco de dados está sendo interrompido.

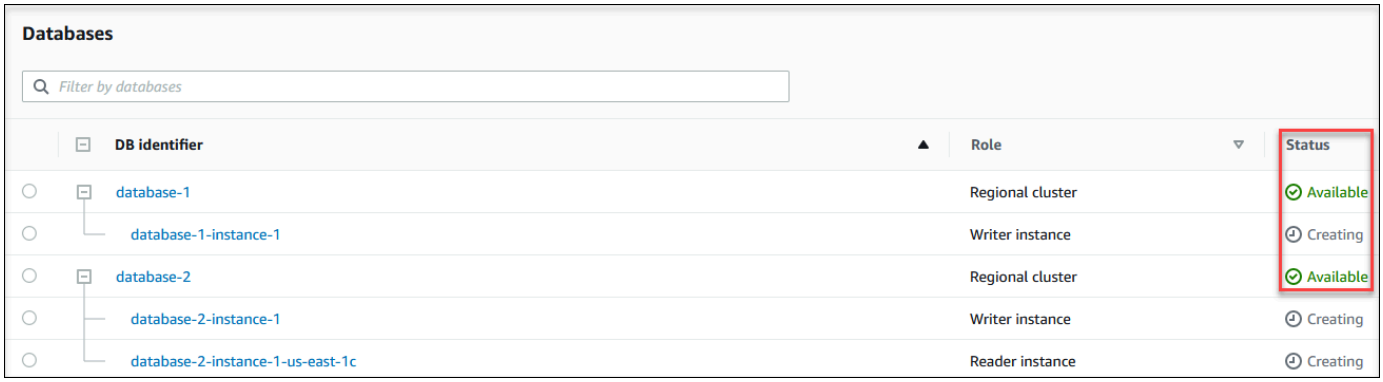
Status do cluster de banco de dados	Faturado	Descrição
Storage-optimization	Faturado	Sua instância de banco de dados está sendo modificada para alterar o tamanho ou o tipo de armazenamento. A instância de banco de dados está totalmente operacional. No entanto, quando o status de sua instância de banco de dados for storage-optimization (otimizaç ão do performance), você não poderá solicitar alterações de armazenamento na sua instância de banco de dados. O processo de otimizaçã o de armazenamento geralmente é curto, mas, às vezes, pode levar até 24 horas ou mais.
Update-iam-db-auth	Faturado	A autorização do IAM para o cluster de banco de dados está sendo atualizada.
Upgrading	Faturado	A versão do mecanismo do cluster de banco de dados ou sistema operacional está sendo atualizada.
upgrade_failed	Não faturado	O cluster de banco de dados não foi atualizad o para uma versão aceita. O Aurora cria um snapshot final com o prefixo `rds-final` .

Console

Como visualizar o status de um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.

   A página Databases (Bancos de dados) aparece com a lista de clusters de banco de dados. Para cada cluster de banco de dados, o valor do status é exibido.



CLI

Para visualizar apenas o status dos clusters de banco de dados, use a consulta a seguir em AWS CLI.

```
aws rds describe-db-clusters --query 'DBClusters[*].[DBClusterIdentifier,Status]' --
output table
```

Visualizar o status de uma instância de banco de dados em um cluster do Aurora

O status de uma instância de banco de dados em um cluster do Aurora indica o estado operacional atual. É possível usar os procedimentos a seguir para visualizar o status de uma instância de banco de dados de um cluster no console do Amazon RDS, o comando AWS CLI ou a operação de API.

> **Note**
>
> O Amazon RDS também usa outro status chamado status de manutenção, que é mostrado na coluna Maintenance (Manutenção) do console do Amazon RDS. Este valor indica o status de todos os patches de manutenção que precisarem ser aplicados a uma instância de banco de dados. O status de manutenção é independente do status da instância do banco de dados. Para ter mais informações sobre o status de manutenção, consulte Aplicar atualizações para um clusterde banco de dados.

Encontre os valores possíveis do status de instâncias de banco de dados na tabela a seguir. Essa tabela mostra se você será cobrado pela instância de banco de dados e pelo armazenamento, apenas pelo armazenamento ou se não será cobrado. Para todos os status de instância de banco de dados, você sempre será cobrado pelo uso de backup.

Status da instância de banco de dados	Faturado	Descrição
available	Faturado	A instância de banco de dados está disponível para modificações.
backing-up	Faturado	No momento, está sendo feito o backup da instância de banco de dados.
retrocesso	Faturado	A instância de banco de dados está sendo retrocedida no momento. Esse status se aplica somente ao Aurora MySQL.
configuring-enhanced-monitoring	Faturado	O monitoramento avançado está sendo habilitado ou desabilitado para essa instância de banco de dados.

Status da instância de banco de dados	Faturado	Descrição
configuring-iam-database-auth	Faturado	AWS Identity and Access Management A autenticação do banco de dados do (IAM) está sendo habilitada ou desabilitada para essa instância de banco de dados.
configuring-log-exports	Faturado	A publicação dos arquivos de log no Amazon CloudWatch Logs está sendo habilitada ou desabilitada para essa instância de banco de dados.
converting-to-vpc	Faturado	A instância de banco de dados está sendo convertida de uma instância de banco de dados que não esteja em uma Amazon Virtual Private Cloud (Amazon VPC) em uma instância de banco de dados que esteja em uma Amazon VPC.
creating	Não faturado (sem PITR) Faturado (somente PITR)	A instância de banco de dados está sendo criada. A instância de banco de dados permanece inacessível enquanto é criada. Se você restaurar um banco de dados durante a recuperação para um ponto no tempo (PITR), a cobrança será feita enquanto o banco de dados estiver no estado criando. Esse é o único cenário em que o estado criando incorre em cobranças.
delete-precheck	Não faturado	O Amazon RDS está validando que as réplicas de leitura podem ser excluídas com segurança.
deleting	Não faturado	A instância de banco de dados está sendo excluída.
com falha	Não faturado	A instância de banco de dados falhou e o Amazon RDS não pode recuperá-la. Execute uma restauração point-in-time no último momento restaurável da instância de banco de dados para recuperar os dados.

Status da instância de banco de dados	Faturado	Descrição
inaccessible-encryption-credentials	Não faturado	A AWS KMS key usada para criptografar ou descriptografar a instância de banco de dados não pode ser acessada nem recuperada.
inaccessible-encryption-credentials-recoverable	Faturado para armazenamento	A chave do KMS usada para criptografar ou descriptografar a instância de banco de dados não pode ser acessada. No entanto, se a chave do KMS estiver ativa, ela pode ser recuperada reiniciando a instância de banco de dados. Para obter mais informações, consulte Criptografar um cluster de banco de dados do Amazon Aurora.
incompatible-create	Não faturado	O Amazon RDS está tentando criar uma instância de banco de dados, mas não pode fazer isso porque os recursos são incompatíveis com sua instância de banco de dados. Esse status poderá ocorrer se, por exemplo, o perfil de instância do seu banco de dados não tiver as permissões corretas.
incompatible-network	Não faturado	O Amazon RDS está tentando realizar uma ação de recuperação em uma instância de banco de dados, mas não pode fazer isso porque a VPC está em um estado que impede a conclusão da ação. Este status pode ocorrer se, por exemplo, todos os endereços IP disponíveis em uma sub-rede estiverem em uso e o Amazon RDS não puder obter um endereço IP para a instância de banco de dados.

Status da instância de banco de dados	Faturado	Descrição
incompatible-option-group	Faturado	O Amazon RDS tentou aplicar uma alteração de grupo de opções, mas não pôde fazê-lo. O Amazon RDS não poderá reverter para o estado anterior do grupo de opções. Para ter mais informações, confira a lista Recent Events (Eventos recentes) da instância de banco de dados. Este status pode ocorrer se, por exemplo, o grupo de opções contém uma opção como TDE e a instância de banco de dados não contém informações criptografadas.
incompatible-parameters	Faturado	O Amazon RDS não pode iniciar a instância de banco de dados porque os parâmetros especificados no parameter group de banco de dados da instância de banco de dados não são compatíveis com a instância. Reverta as alterações de parâmetro ou as torne compatíveis com a instância de banco de dados para retomar o acesso à instância de banco de dados. Para ter mais informações sobre parâmetros incompatíveis, confira a lista Recent Events (Eventos recentes) da instância de banco de dados.
incompatible-restore	Não faturado	O Amazon RDS não pode fazer uma restauração point-in-time. As causas comuns para esse status incluem o uso de tabelas temporáriasou o uso de tabelas MyISAM com MySQL.

Status da instância de banco de dados	Faturado	Descrição
insuficiência de capacidade	Não faturado	O Amazon RDS não consegue criar a instância porque não há capacidade suficiente disponível no momento. Para criar sua instância de banco de dados na mesma AZ com o mesmo tipo de instância , exclua a instância de banco de dados, aguarde algumas horas e tente criá-la novamente. Como alternativa, crie uma nova instância usando uma classe de instância diferente ou AZ.
manutenção	Faturado	O Amazon RDS está aplicando uma atualização de manutenção na instância de banco de dados. Este status é usado para a manutenção de nível de instância que o RDS agenda com antecedência.
modifying	Faturado	A instância de banco de dados está sendo modificad a por causa de uma solicitação do cliente.
moving-to-vpc	Faturado	A instância de banco de dados está sendo movida para uma nova Amazon Virtual Private Cloud (Amazon VPC).
reinicialização	Faturado	A instância de banco de dados está sendo reinicial izada por causa de uma solicitação do cliente ou de um processo do Amazon RDS que exige a reinicial ização da instância.
resetting-master-c redentials	Faturado	As credenciais principais da instância de banco de dados estão sendo redefinidas por causa de uma solicitação do cliente.
renaming	Faturado	A instância de banco de dados está sendo renomeada por causa de uma solicitação do cliente.

Status da instância de banco de dados	Faturado	Descrição
restore-error	Faturado	A instância de banco de dados encontrou um erro ao tentar restaurar para um determinado point-in-time ou de um snapshot.
starting	Faturado para armazenamento	A instância do banco de dados está iniciando.
interrompido	Faturado para armazenamento	A instância do banco de dados está interrompida.
interromper	Faturado para armazenamento	A instância do banco de dados está sendo interrompida.
storage-config-upgrade	Faturado	A configuração do sistema de arquivos de armazenamento da instância de banco de dados está sendo atualizada. Esse status só se aplica a bancos de dados verdes em uma implantação azul/verde ou a réplicas de leitura de instâncias de banco de dados.
storage-full	Faturado	A instância de banco de dados alcançou a alocação da capacidade de armazenamento. Esse é um status crítico e recomendamos que você corrija esse problema imediatamente. Para fazer isso, aumente seu armazenamento modificando a instância de banco de dados. Para evitar essa situação, configure os alarmes do Amazon CloudWatch para adverti-lo quando o espaço de armazenamento estiver ficando baixo.

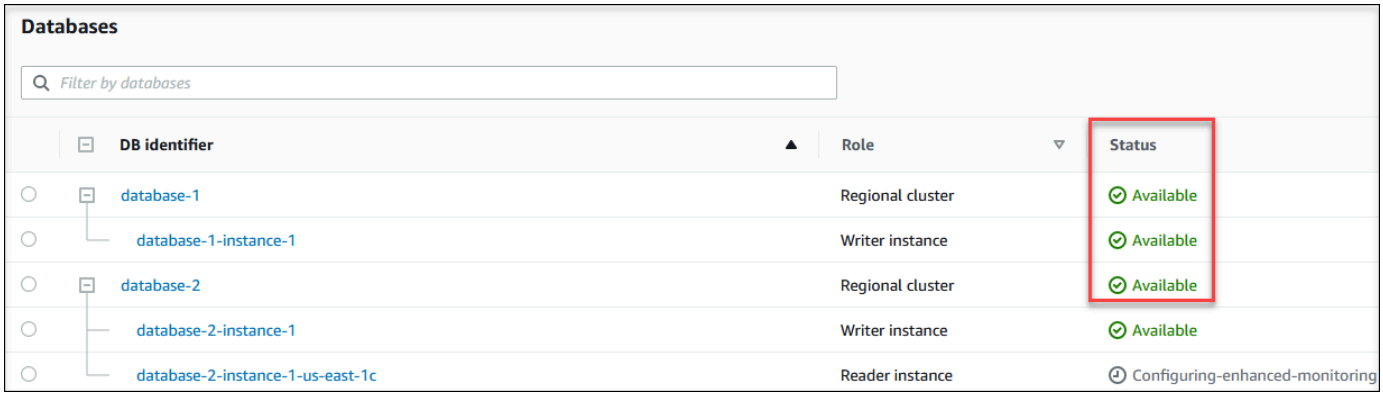
Status da instância de banco de dados	Faturado	Descrição
storage-initialization	Faturado	A instância de banco de dados está carregando blocos de dados do Amazon S3 para otimizar o desempenho do volume após a restauração de um snapshot. Ela permanece disponível para operações , mas o desempenho pode não estar na capacidade máxima até que a inicialização seja concluída.
storage-optimization	Faturado	O Amazon RDS está otimizando o armazenamento de sua instância de banco de dados. O processo de otimização de armazenamento geralmente é curto, mas, às vezes, pode levar até 24 horas ou mais. Durante a otimização do armazenamento, a instância de banco de dados permanece disponíve l. A otimização do armazenamento é um processo em segundo plano que não afeta a disponibilidade da instância.
upgrading	Faturado	A versão do mecanismo do banco de dados ou sistema operacional está sendo atualizada.
upgrade_failed	Não faturado	A instância de banco de dados não foi atualizada para uma versão aceita. O Aurora cria um snapshot final com o prefixo `rds-final` .

Console

Como visualizar o status de uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.

   A página Databases (Bancos de dados) aparece com a lista de instâncias de banco de dados. Para cada instância de banco de dados em um cluster, o valor do status é exibido.



CLI

Para visualizar a instância de banco de dados e suas informações de status usando a AWS CLI, utilize o comando describe-db-instances. Por exemplo, o comando AWS CLI a seguir lista todas as informações de instâncias de banco de dados.

```
aws rds describe-db-instances
```

Para visualizar uma instância de banco de dados específica e seu status, chame o comando describe-db-instances com a seguinte opção:

- `DBInstanceIdentifier`: o nome da instância de banco de dados.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

Para visualizar apenas o status de todas as instâncias de banco de dados, utilize a consulta a seguir na AWS CLI.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

solicitações de

Para visualizar o status da instância de banco de dados usando a API do Amazon RDS, chame a operação DescribeDBInstances.

Recomendações do Amazon Aurora

O Amazon Aurora fornece recomendações automatizadas para recursos de banco de dados, como instâncias de banco de dados, clusters de banco de dados, e grupo de parâmetros de cluster de banco de dados. Essas recomendações fornecem orientações de práticas recomendadas, analisando a configuração do cluster de banco de dados, a configuração, o uso e os dados de performance da instância de banco de dados.

O Insights de Performance do Amazon RDS monitora métricas específicas e cria limites automaticamente analisando quais níveis são considerados possivelmente problemáticos para um recurso específico. Quando novos valores de métricas ultrapassam um limite predefinido em um período específico, o Performance Insights gera uma recomendação proativa. Essa recomendação ajuda a evitar um impacto futuro na performance do banco de dados. Por exemplo, a recomendação “Ocioso na transação” é gerada para instâncias do Aurora PostgreSQL quando as sessões conectadas ao banco de dados não estão realizando trabalho ativo, mas podem manter os recursos do banco de dados bloqueados. Para receber recomendações proativas, é necessário ativar o Performance Insights com um período de retenção de nível pago. Para ter informações sobre como ativar o Performance Insights, consulte Ativar e desativar o Insights de Performance para o Aurora. Para ter informações sobre preços e retenção de dados para o Performance Insights, consulte Preços e retenção de dados para o Performance Insights.

O DevOps Guru para RDS monitora determinadas métricas para detectar quando o comportamento de uma métrica se torna altamente incomum ou anômalo. Essas anomalias são relatadas como insights reativos com recomendações. Por exemplo, o DevOps Guru para RDS pode recomendar que você aumente a capacidade da CPU ou investigue eventos de espera que estão contribuindo para a carga do banco de dados. O DevOps Guru para RDS também fornece recomendações proativas baseadas em limites. Para essas recomendações, é necessário ativar o DevOps Guru para RDS. Para ter informações sobre como ativar o DevOps Guru para RDS, consulte Ativar o DevOps Guru e especificar a cobertura de recursos.

As recomendações estarão em qualquer um dos seguintes status: ativas, rejeitadas, pendentes ou resolvidas. As recomendações resolvidas ficam disponíveis por 365 dias.

É possível ver ou ignorar as recomendações. É possível aplicar uma recomendação ativa baseada em configuração imediatamente, programá-la para a próxima janela de manutenção ou descartá-la. Para recomendações proativas baseadas em limites e baseadas em machine learning, é necessário analisar a causa sugerida do problema e, depois, realizar as ações recomendadas para corrigir o problema.

As recomendações são aceitas nas seguintes Regiões da AWS:

- Leste dos EUA (Ohio)
- Leste dos EUA (Norte da Virgínia)
- Oeste dos EUA (Norte da Califórnia)
- Oeste dos EUA (Oregon)
- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Paris)
- Europa (Estocolmo)
- América do Sul (São Paulo)

Aprenda a visualizar, aplicar, ignorar e modificar recomendações do Amazon Aurora nas seções a seguir.

Tópicos

- Visualizar as recomendações Amazon Aurora
- Aplicar as recomendações do Amazon Aurora
- Ignorar as recomendações do Amazon Aurora
- Modificar as recomendações ignoradas do Amazon Aurora para recomendações ativas
- Recomendações da referência do Amazon Aurora

Visualizar as recomendações Amazon Aurora

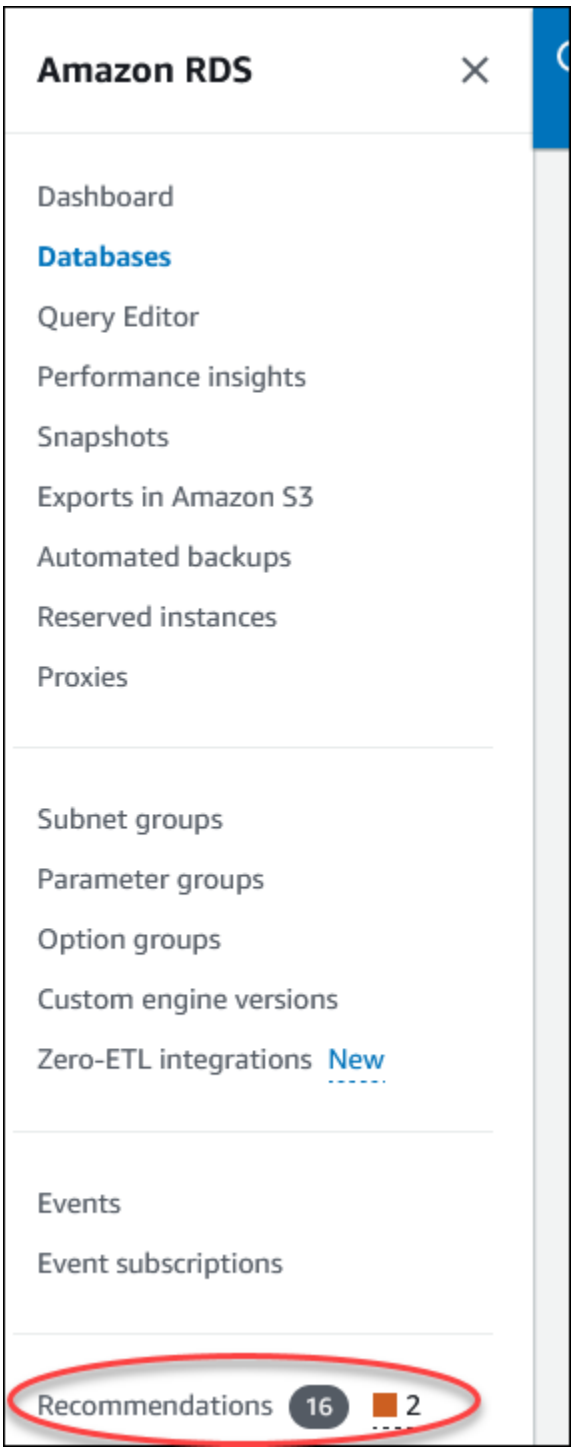
Usando o console do Amazon RDS, é possível visualizar as recomendações do Amazon Aurora para os recursos de banco de dados. Para um cluster de banco de dados, as recomendações aparecem para o cluster de banco de dados e as instâncias.

Console

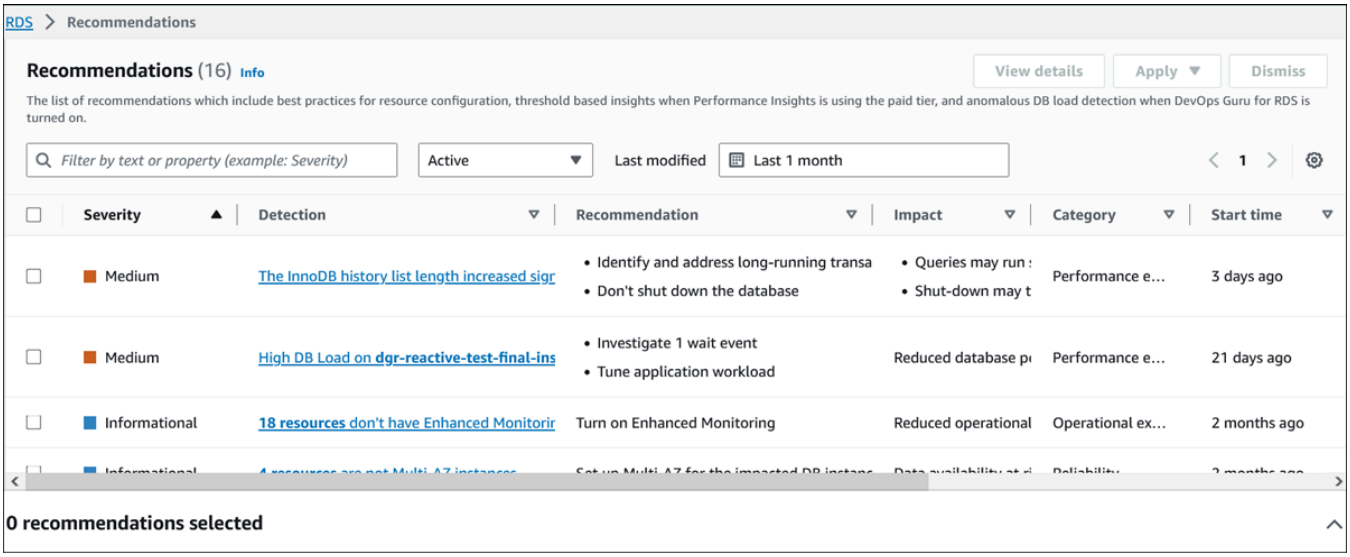
Como visualizar recomendações do Amazon Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, execute qualquer uma das seguintes opções:

   - Selecione Recomendações. O número de recomendações ativas para os recursos e o número de recomendações com maior gravidade geradas no último mês estão disponíveis ao lado de Recomendações. Para encontrar o número de recomendações ativas para cada gravidade, selecione o número que mostra a maior gravidade.

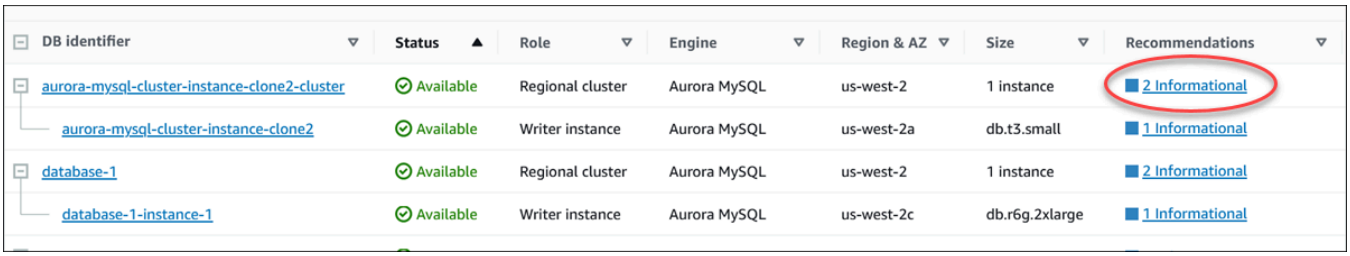


Por padrão, a página Recomendações exibe uma lista de novas recomendações no último mês. O Amazon Aurora fornece recomendações para todos os recursos na conta e as classifica de acordo com a gravidade.

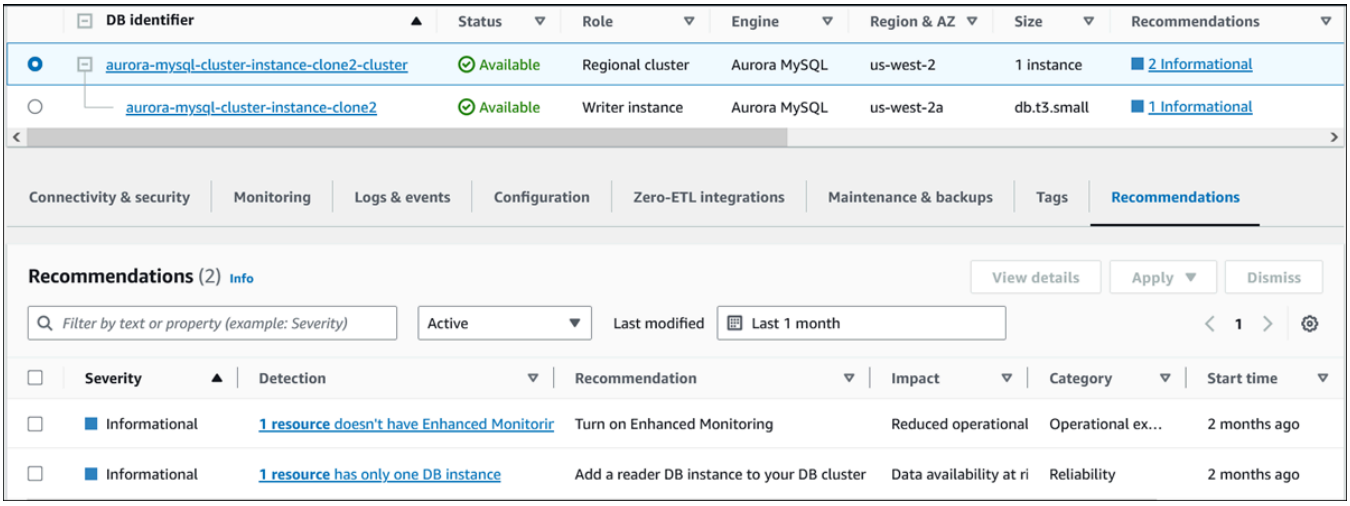


É possível selecionar uma recomendação para ver uma seção na parte inferior da página que contém os recursos afetados e detalhes de como a recomendação será aplicada.

- Na página Bancos de dados, selecione Recomendações para um recurso.



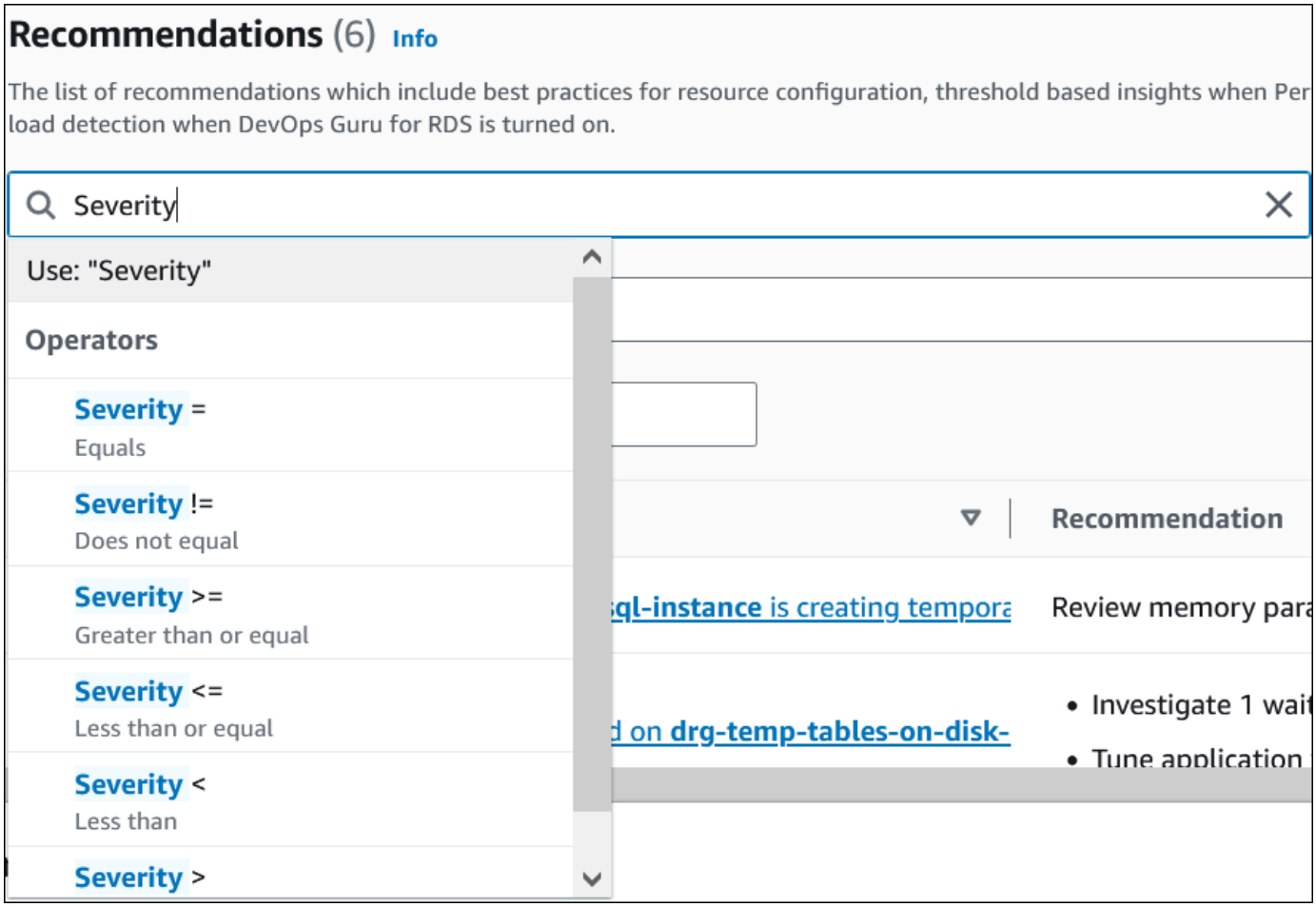
A guia Recomendações exibe as recomendações e os detalhes do recurso selecionado.



Os detalhes a seguir estão disponíveis para as recomendações:

- Gravidade: o nível de implicação do problema. Os níveis de gravidade são Alto, Médio, Baixo e Informativo.
- Detecção: o número de recursos afetados e uma breve descrição do problema. Selecione este link para ver a recomendação e os detalhes da análise.
- Recomendação: uma breve descrição da ação recomendada a ser aplicada.
- Impacto: uma breve descrição do possível impacto quando a recomendação não é aplicada.
- Categoria: o tipo de recomendação. As categorias são Eficiência de performance, Segurança, Confiabilidade, Otimização de custos, Excelência operacional e Sustentabilidade.
- Status: o status atual da recomendação. Os status possíveis são Todos, Ativo, Dispensado, Resolvido e Pendente.
- Horário de início: a hora em que o problema começou. Por exemplo, 18 horas atrás.
- Última modificação: a hora em que a recomendação foi atualizada pela última vez pelo sistema devido a uma alteração na Gravidade ou a hora em que você respondeu à recomendação. Por exemplo, 10 horas atrás.
- Horário de término: a hora em que o problema terminou. A hora não será exibida para nenhum problema contínuo.
- Identificador do recurso: o nome de um ou mais recursos.

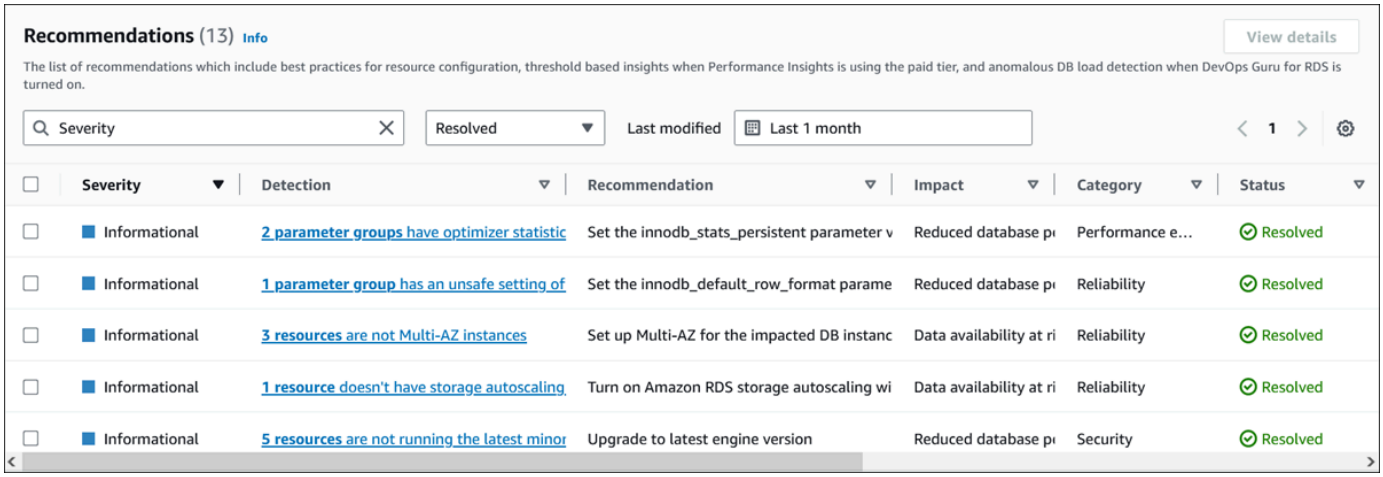
3. (Opcional) Selecione os operadores de Gravidade ou Categoria no campo para filtrar a lista de recomendações.



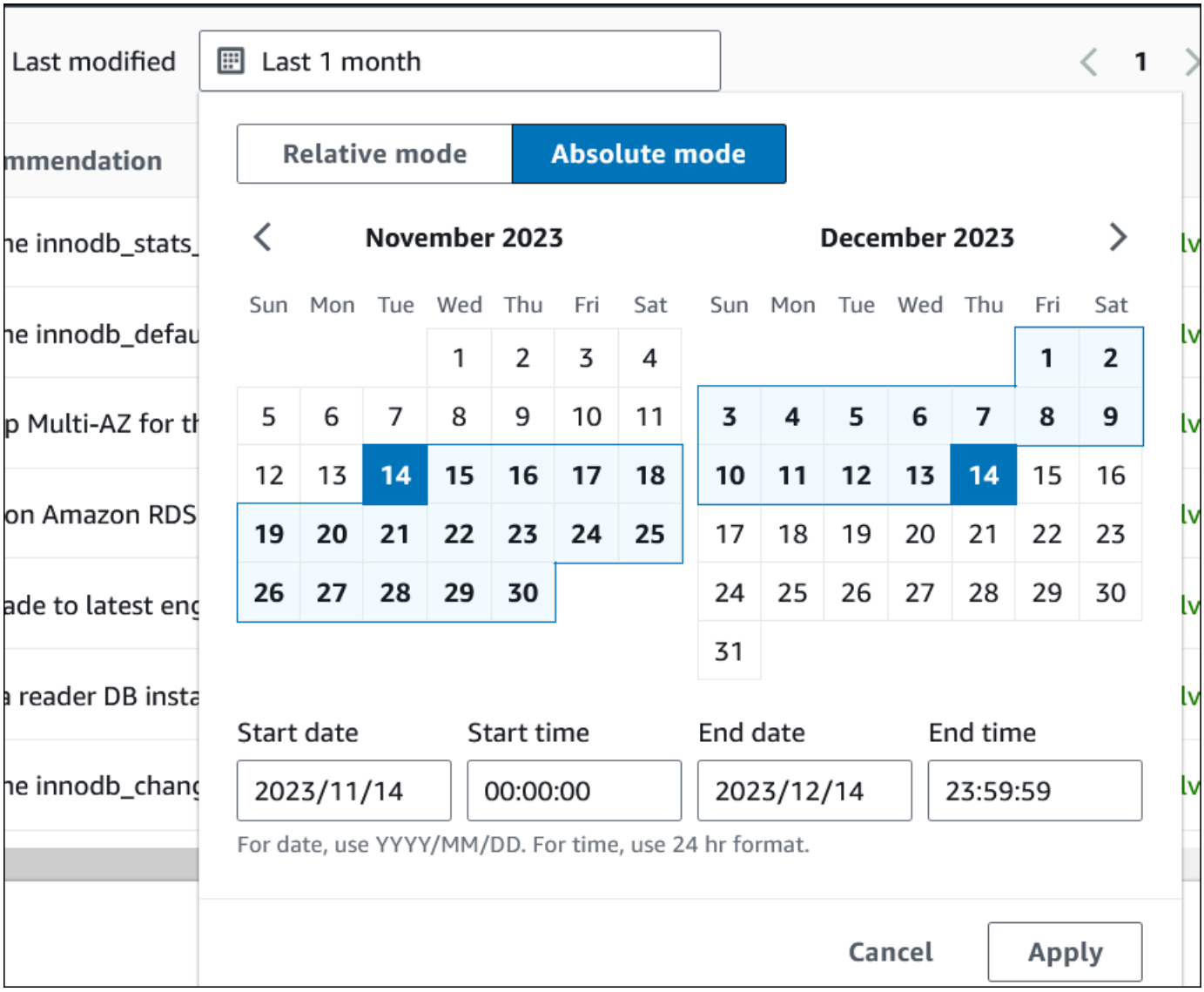
As recomendações para a operação selecionada são exibidas.

4. (Opcional) Selecione qualquer um dos seguintes status de recomendação:

   - Ativo (padrão): mostra as recomendações atuais que você pode aplicar, programar para a próxima janela de manutenção ou dispensar.
   - Todos: mostra todas as recomendações com o status atual.
   - Dispensado: mostra as recomendações dispensadas.
   - Resolvido: mostra as recomendações resolvidas.
   - Pendente: mostra as recomendações cujas ações recomendadas estão em andamento ou programadas para a próxima janela de manutenção.



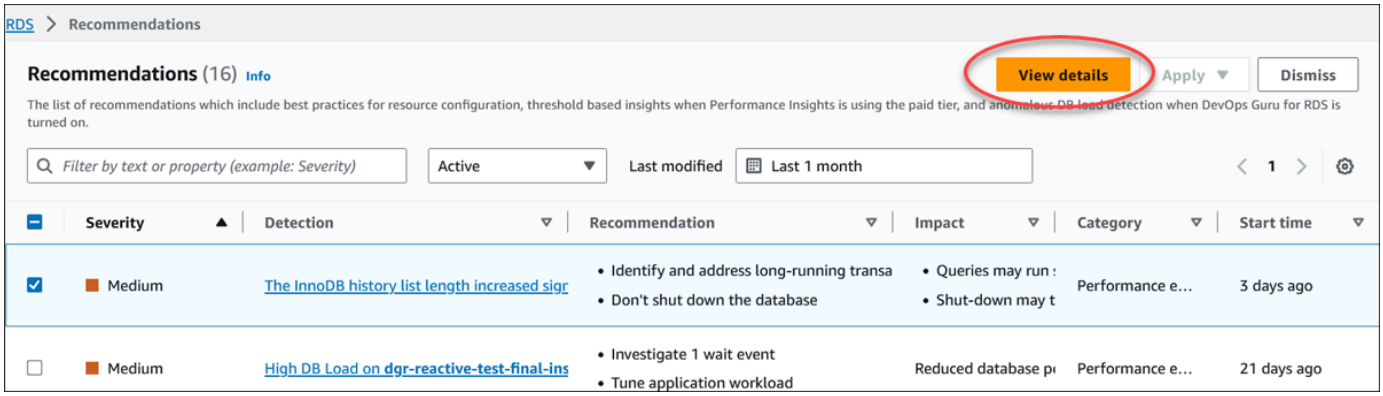
5. (Opcional) Selecione Modo relativo ou Modo absoluto em Última modificação para modificar o período. A página Recomendações exibe as recomendações geradas no período. O período padrão é o último mês. No Modo absoluto, é possível escolher o período ou inserir a hora nos campos Data de início e Data de término.



As recomendações para o período definido são exibidas.

Observe que é possível ver todas as recomendações de recursos na conta definindo o intervalo como Todos.

6. (Opcional) Selecione Preferências à direita para personalizar os detalhes a serem exibidos. É possível escolher um tamanho de página, quebrar as linhas do texto e exibir ou ocultar as colunas.
7. (Opcional) Selecione uma recomendação e, depois, escolha Visualizar detalhes.



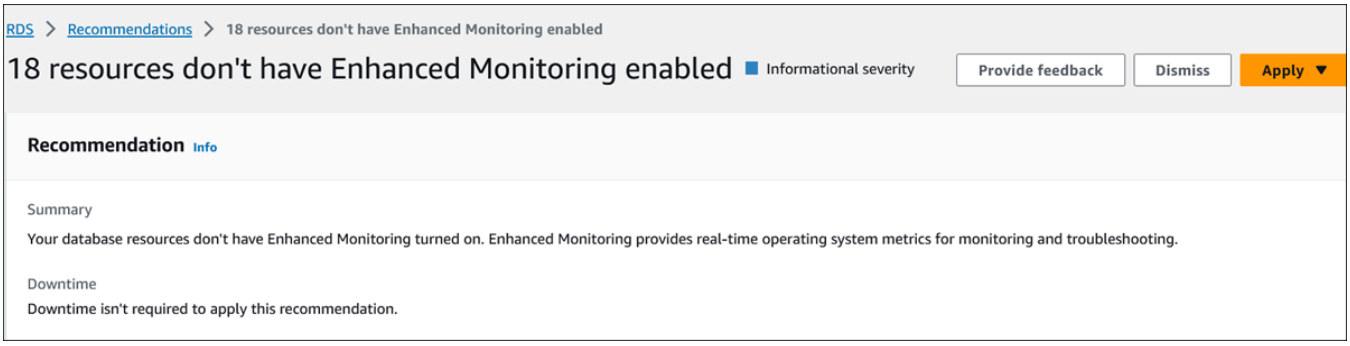
A página de detalhes da recomendação é exibida. O título fornece a contagem total dos recursos com o problema detectado e a gravidade.

Para ter informações sobre os componentes na página de detalhes de uma recomendação reativa baseada em anomalias, consulte Viewing reactive anomalies no Guia do usuário do Amazon DevOps Guru.

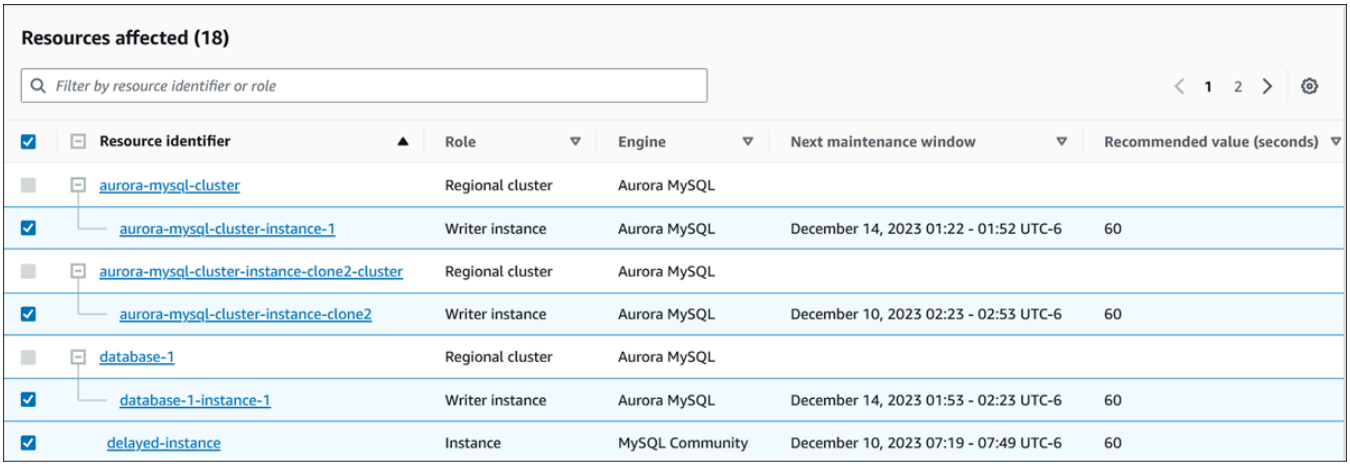
Para ter informações sobre os componentes na página de detalhes de uma recomendação proativa baseada em limites, consulte Visualizar as recomendações proativas do Performance Insights.

As outras recomendações automatizadas exibem os seguintes componentes na página de detalhes da recomendação:

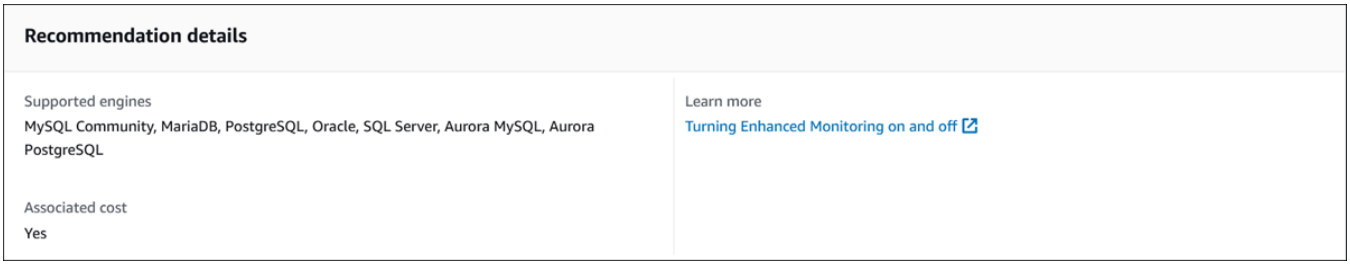
- Recomendação: um resumo da recomendação e se é necessário tempo de inatividade para aplicar a recomendação.



- Recursos afetados: detalhes dos recursos afetados.



- Detalhes da recomendação: informações do mecanismo compatível, qualquer custo associado necessário para aplicar a recomendação e link da documentação para saber mais.



CLI

Para visualizar as recomendações do Amazon RDS das instâncias de banco de dados ou clusters de banco de dados, use o comando a seguir em AWS CLI.

```
aws rds describe-db-recommendations
```

API do RDS

Para visualizar as recomendações do Amazon RDS usando a API do Amazon RDS, use a operação DescribeDBRecommendations.

Aplicar as recomendações do Amazon Aurora

Para aplicar as recomendações do Amazon Aurora usando o console do Amazon RDS, selecione uma recomendação baseada na configuração ou um recurso afetado na página de detalhes. Depois, opte por aplicar a recomendação imediatamente ou programá-la para a próxima janela de manutenção. Talvez seja necessário que o recurso seja reiniciado para que a alteração tenha efeito.

Para algumas recomendações de grupos de parâmetros de banco de dados, talvez seja necessário reiniciar os recursos.

As recomendações proativas baseadas em limites ou reativas baseadas em anomalias não terão a opção de aplicação e poderão precisar de análise adicional.

Console

Como aplicar uma recomendação baseada em configuração

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, execute qualquer uma das seguintes opções:

   - Selecione Recomendações.

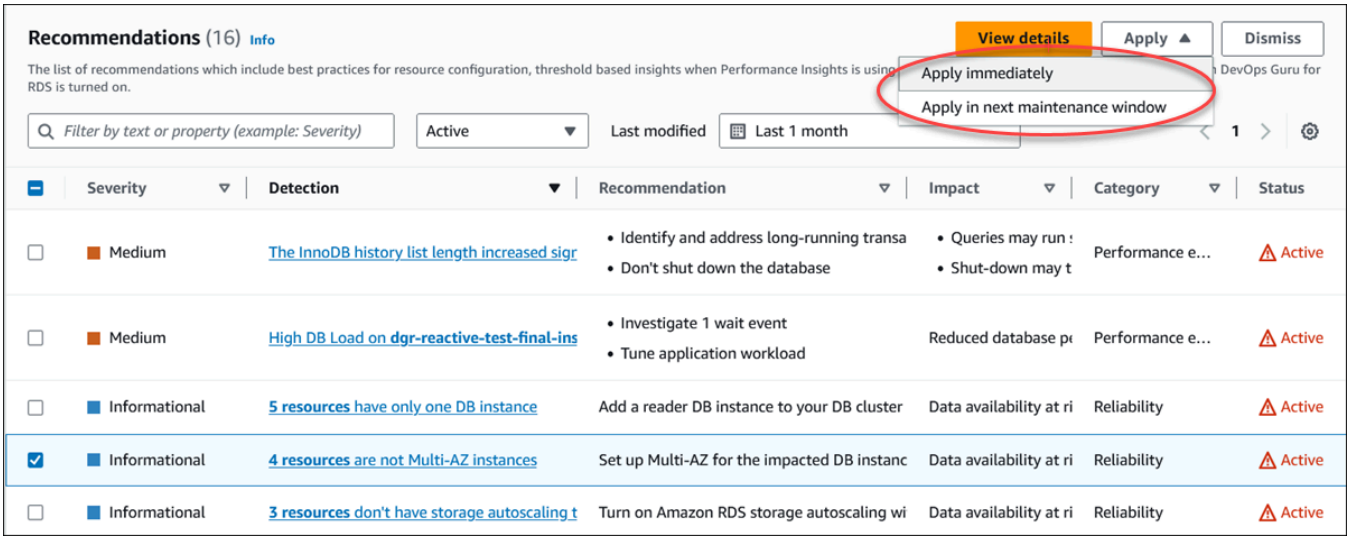
     A página Recomendações aparece com a lista de todas as recomendações.
   - Selecione Bancos de dados e, depois, escolha Recomendações para um recurso na página de bancos de dados.

     Os detalhes são exibidos na guia Recomendações da recomendação selecionada.
   - Selecione Detecção para receber uma recomendação ativa na página Recomendações ou na guia Recomendações na página Bancos de dados.

     A página de detalhes da recomendação é exibida.
3. Selecione uma recomendação ou um ou mais recursos afetados na página de detalhes da recomendação e faça o seguinte:

   - Selecione Aplicar e, depois, escolha Aplicar imediatamente para aplicar a recomendação imediatamente.
   - Selecione Aplicar e, depois, escolha Aplicar durante a próxima janela de manutenção programada para programar na próxima janela de manutenção.

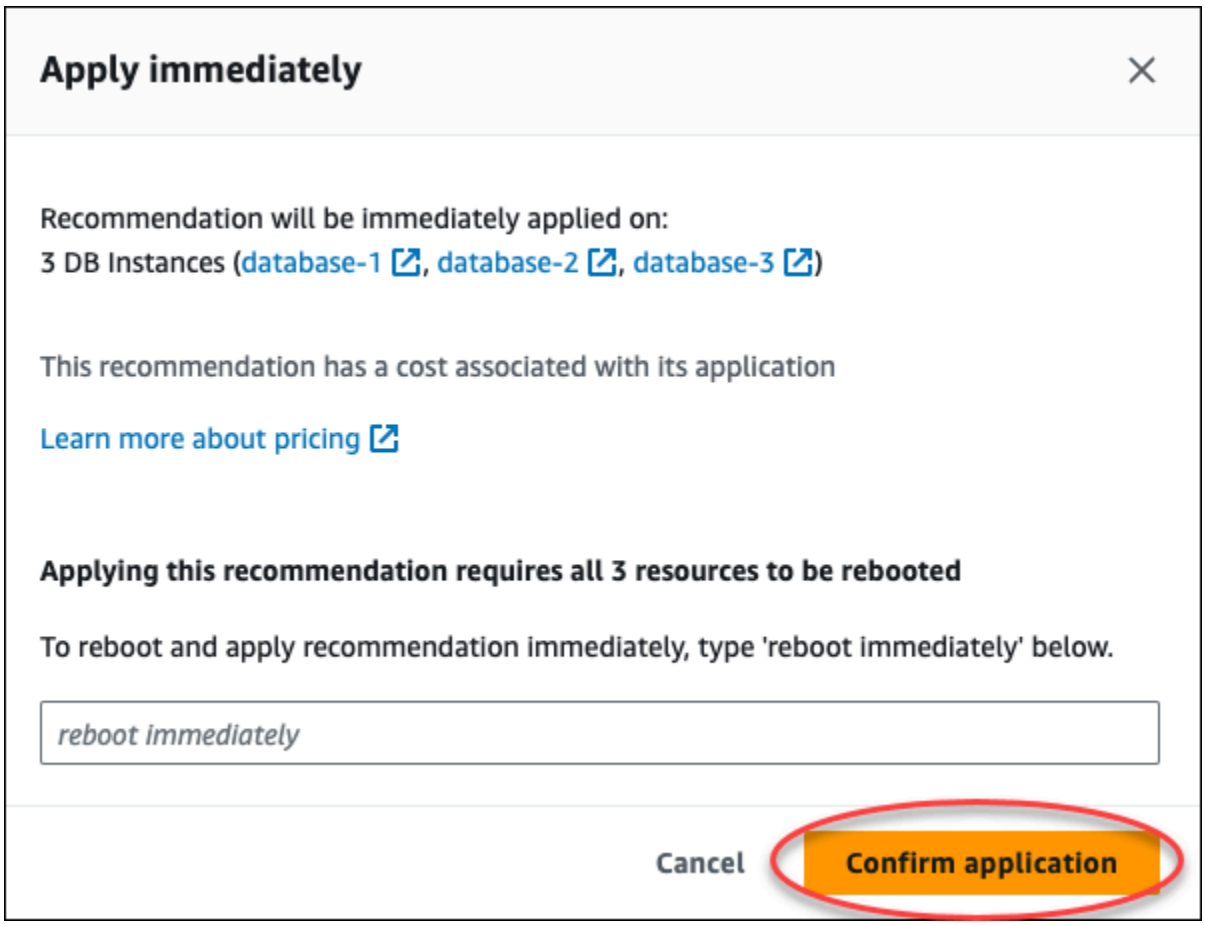
     O status da recomendação selecionada é atualizado para pendente até a próxima janela de manutenção.



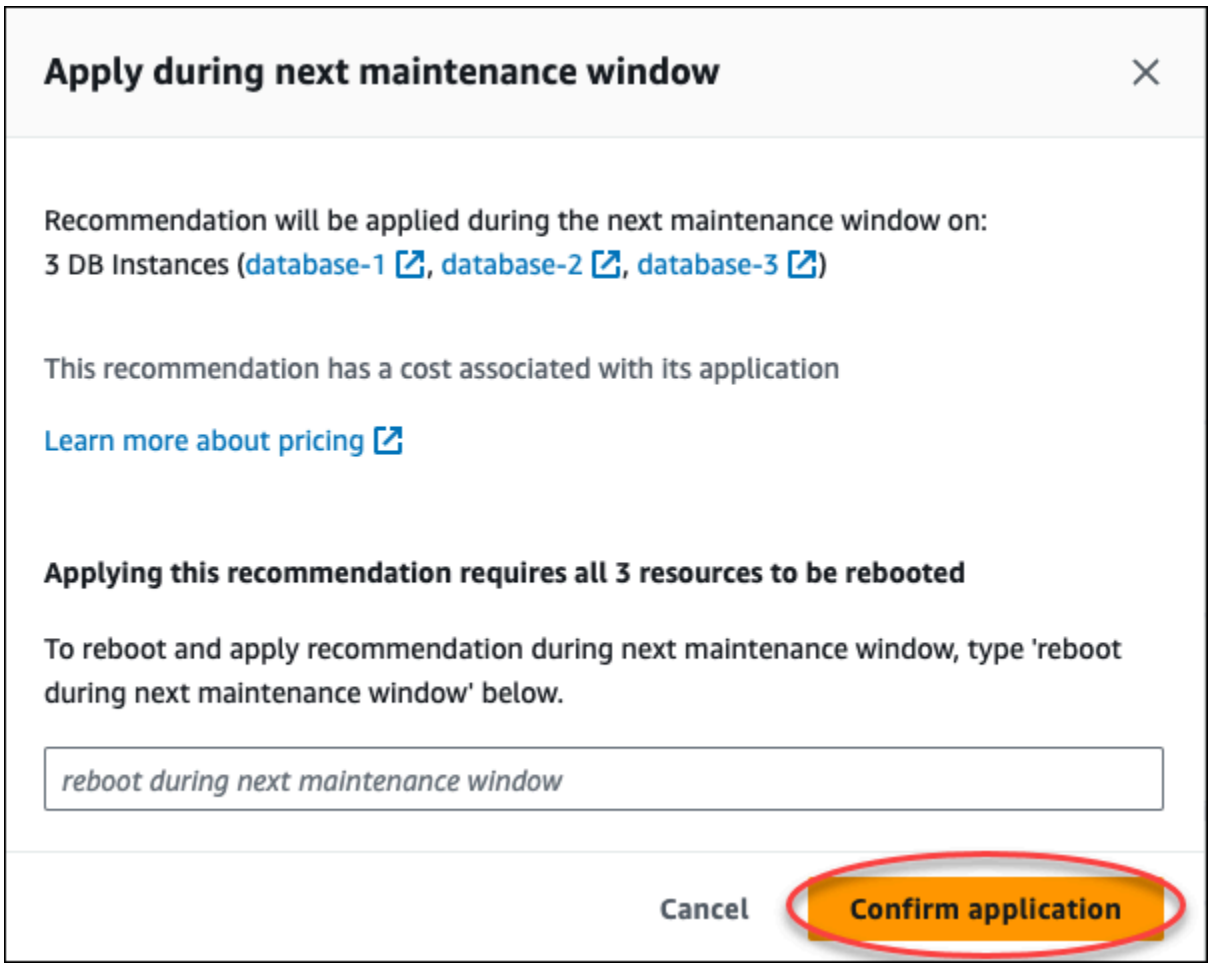
Uma janela de confirmação é exibida.

4. Selecione Confirmar aplicação para aplicar a recomendação. Essa janela confirma se os recursos precisam de uma reinicialização automática ou manual para que as alterações tenham efeito.

   O exemplo a seguir mostra a janela de confirmação para aplicar a recomendação imediatamente.

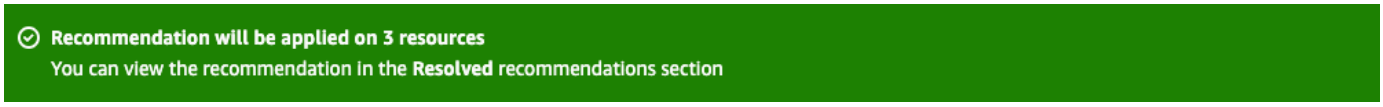


O exemplo a seguir mostra a janela de confirmação para programar a aplicação da recomendação na próxima janela de manutenção.



Um banner exibe uma mensagem quando a recomendação aplicada é bem-sucedida ou falhou.

O exemplo a seguir mostra o banner com a mensagem de êxito.



O exemplo a seguir mostra o banner com a mensagem de falha.

API do RDS

Para aplicar uma recomendação do Aurora baseada em configuração usando a API do Amazon RDS

1. Use a operação DescribeDBRecommendations. `RecommendedActions` na saída pode ter uma ou mais ações recomendadas.
2. Use o objeto RecommendedAction para cada ação recomendada da etapa 1. A saída contém `Operation` e `Parameters`.

   O exemplo a seguir mostra a saída com uma ação recomendada.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. Use `operation` para cada ação recomendada da saída na etapa 2 e insira os valores `Parameters`.
4. Depois que a operação na etapa 2 for bem-sucedida, use a operação ModifyDBRecommendation para modificar o status da recomendação.

Ignorar as recomendações do Amazon Aurora

É possível ignorar uma ou mais recomendações do Amazon Aurora usando o console do Amazon RDS, a AWS CLI ou a API do Amazon RDS.

Console

Como dispensar uma ou mais recomendações

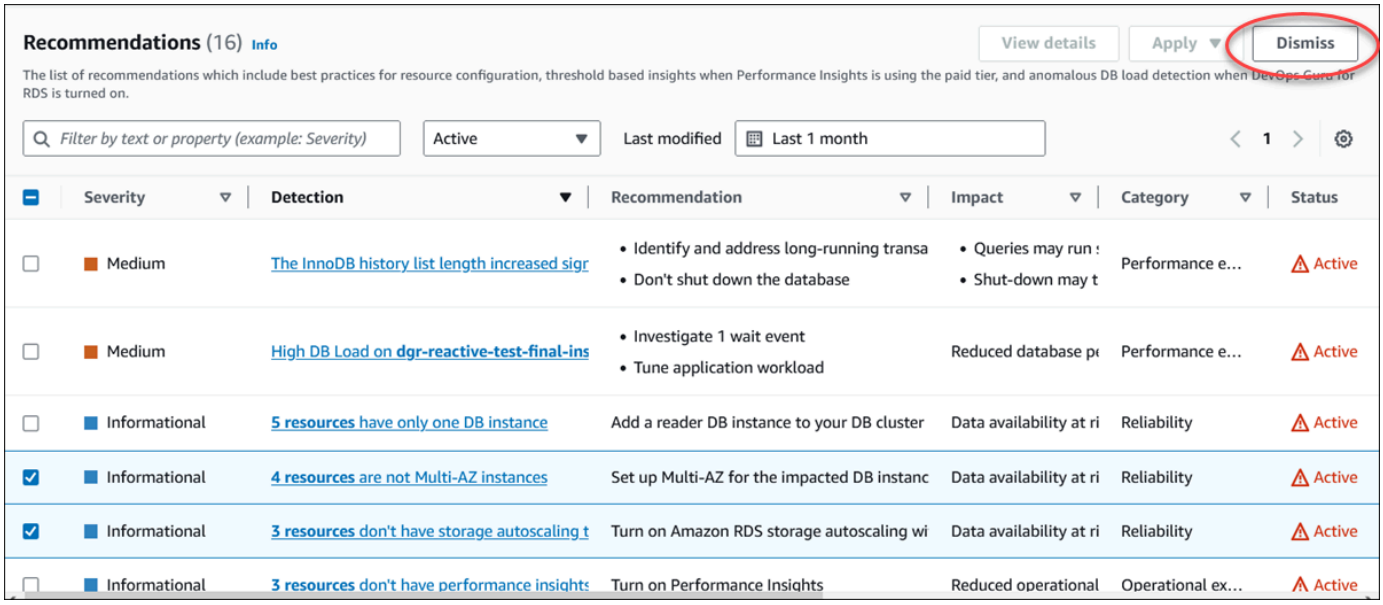
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, execute qualquer uma das seguintes opções:
   - Selecione Recomendações.

     A página Recomendações aparece com a lista de todas as recomendações.
   - Selecione Bancos de dados e, depois, escolha Recomendações para um recurso na página de bancos de dados.

     Os detalhes são exibidos na guia Recomendações da recomendação selecionada.
   - Selecione Detecção para receber uma recomendação ativa na página Recomendações ou na guia Recomendações na página Bancos de dados.

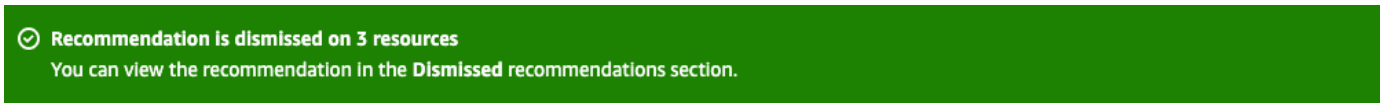
     A página de detalhes da recomendação exibe a lista dos recursos afetados.
3. Escolha uma ou mais recomendações ou um ou mais recursos afetados na página de detalhes da recomendação e selecione Dispensar.

   O exemplo a seguir mostra a página Recomendações com várias recomendações ativas selecionadas para serem dispensadas.



Um banner exibe uma mensagem quando uma ou mais recomendações selecionadas são dispensadas.

O exemplo a seguir mostra o banner com a mensagem de êxito.



O exemplo a seguir mostra o banner com a mensagem de falha.

CLI

Como dispensar uma recomendação do Aurora usando a AWS CLI

1. Execute o comando `aws rds describe-db-recommendations --filters "Name=status,Values=active"`.

   A saída fornece uma lista de recomendações no status `active`.
2. Encontre o `recommendationId` para a recomendação que você deseja dispensar na etapa 1.
3. Execute o comando `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` com o `recommendationId` da etapa 2 para dispensar a recomendação.

API do RDS

Para dispensar uma recomendação do Aurora usando a API do Amazon RDS, use a operação ModifyDBRecommendation.

Modificar as recomendações ignoradas do Amazon Aurora para recomendações ativas

É possível modificar uma ou mais recomendações ignoradas do Amazon Aurora para recomendações ativas usando o console do Amazon RDS, a AWS CLI ou a API do Amazon RDS.

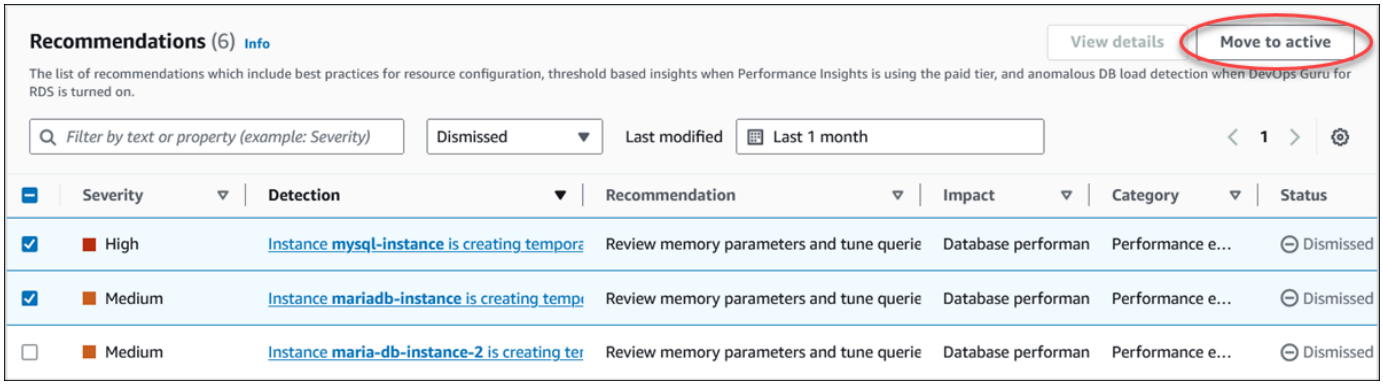
Console

Como mover uma ou mais recomendações dispensadas para recomendações ativas

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, execute qualquer uma das seguintes opções:
   - Selecione Recomendações.

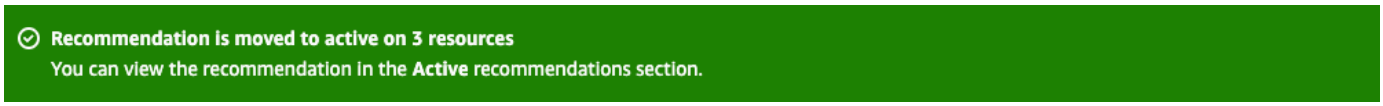
     A página Recomendações exibe uma lista de recomendações classificadas pela gravidade de todos os recursos da conta.
   - Selecione Bancos de dados e, depois, escolha Recomendações para um recurso na página de bancos de dados.

     A guia Recomendações exibe as recomendações e os detalhes do recurso selecionado.
3. Escolha uma ou mais recomendações dispensadas na lista e selecione Mover para ativo.



Um banner exibe uma mensagem de êxito ou falha ao mover as recomendações selecionadas do status dispensado para ativo.

O exemplo a seguir mostra o banner com a mensagem de êxito.



O exemplo a seguir mostra o banner com a mensagem de falha.

CLI

Como alterar uma recomendação dispensada do Aurora para uma recomendação ativa usando a AWS CLI

1. Execute o comando `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`.

   A saída fornece uma lista de recomendações no status `dismissed`.
2. Encontre `recommendationId` para a recomendação cujo status você deseja alterar da etapa 1.
3. Execute o comando `>aws rds modify-db-recommendation --status active --recommendationId <ID>` com o `recommendationId` da etapa 2 para alterar para a recomendação ativa.

API do RDS

Para alterar uma recomendação do Aurora para uma recomendação ativa usando a API do Amazon RDS, use a operação ModifyDBRecommendation.

Recomendações da referência do Amazon Aurora

O Amazon Aurora gera recomendações para um recurso quando esse recurso é criado ou modificado. É possível encontrar exemplos de recomendações do Amazon Aurora na tabela a seguir.

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Os backups automatizados do recurso estão desativados	Os backups automatizados não estão ativados para as instâncias de banco de dados. Backups automatizados são recomendados porque permitem a recuperação para um ponto no tempo de instâncias de banco de dados.	Ative os backups automatizados com um período de retenção de até 14 dias.	Sim	Visão geral do backup e da restauração de um cluster de banco de dados do Aurora Desmistificar os custos de armazenamento de backup do Amazon RDS no blog de bancos de dados da AWS
A atualização da versão secundária do mecanismo é necessária.	Os recursos de banco de dados não estão executando a versão secundária mais recente do mecanismo de banco de dados. A versão secundária mais recente contém as correções de segurança mais recentes e outras melhorias.	Atualize para a versão mais recente do mecanismo.	Sim	Manutenção de um cluster de banco de dados do Amazon Aurora
O monitoramento aprimorad	Os recursos de banco de dados não têm o monitoramento	Ative o monitoramento aprimorado.	Não	Monitorar métricas do SO com o monitoramento avançado

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
o está desativado.	aprimorado ativado. O monitoramento avançado fornece métricas do sistema operacional em tempo real para monitoramento e solução de problemas.			

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
A criptografia de armazenamento está desativada.	O Amazon RDS é compatível com a criptografia em repouso para todos os mecanismos de banco de dados usando as chaves gerenciadas no AWS Key Management Service (AWS KMS). Em uma instância de banco de dados ativa com criptografia do Amazon RDS, os dados armazenados em repouso no armazenamento são criptografados, de forma semelhante aos backups automatizados, réplicas de leitura e snapshots. Se a criptografia não estiver ativada durante a criação de um cluster de banco de dados do Aurora, será necessário restaurar um	Ative a criptografia de dados em repouso para o cluster de banco de dados.	Sim	Segurança no Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
	snapshot descriptografado em um cluster de banco de dados criptografado.			
Os clusters de banco de dados com todas as instâncias na mesma zona de disponibilidade.	No momento, os clusters de banco de dados estão em uma única zona de disponibilidade. Use várias zonas de disponibilidade para melhorar a disponibilidade.	Adicione as instâncias de banco de dados a várias zonas de disponibilidade no cluster de banco de dados.	Não	Alta disponibilidade do Amazon Aurora
Instâncias de banco de dados nos clusters com tamanhos de instância heterogêneos	Recomendamos usar a mesma classe e tamanho da instância de banco de dados para todas as instâncias no cluster de banco de dados.	Use a mesma classe e tamanho de instância para todas as instâncias no cluster de banco de dados.	Sim	Replicação com o Amazon Aurora
Instâncias de banco de dados nos clusters com classes de instância heterogêneas	Recomendamos usar a mesma classe e tamanho da instância de banco de dados para todas as instâncias no cluster de banco de dados.	Use a mesma classe e tamanho de instância para todas as instâncias no cluster de banco de dados.	Sim	Replicação com o Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Instâncias de banco de dados nos clusters com grupos de parâmetros heterogêneos	Recomendamos que todas as instâncias de banco de dados no cluster de banco de dados usem o mesmo grupo de parâmetros de banco de dados.	Associe a instância de banco de dados ao grupo de parâmetros de banco de dados associado à instância de gravador no cluster de banco de dados.	Não	Grupos de parâmetros para Amazon Aurora
Os clusters de banco de dados do Amazon RDS têm uma instância de banco de dados.	Adicione pelo menos mais uma instância de banco de dados ao cluster de banco de dados para melhorar a disponibilidade e a performance.	Adicione uma instância de banco de dados de leitor ao cluster de banco de dados.	Não	Alta disponibilidade do Amazon Aurora
O Performance Insights está desativado	O Performance Insights monitora a carga da instância de banco de dados para ajudar a analisar e solucionar problemas de performance do banco de dados. Recomendamos ativar o Performance Insights.	Ative o Performance Insights.	Não	Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
A atualização das versões principais dos recursos do RDS é necessária.	Bancos de dados com a versão principal atual do mecanismo de banco de dados não serão aceitos. Recomendamos atualizar para a versão principal mais recente, que inclui novas funcionalidades e aprimoramentos.	Atualize para a versão principal mais recente do mecanismo de banco de dados.	Sim	Atualizações do Amazon Aurora Criar uma implantação azul/verde no Amazon Aurora
Tamanho máximo do volume do cluster de banco de dados	As versões mais recentes do mecanismo comportam volumes de armazenamento maiores para seu cluster de banco de dados.	Recomendamos atualizar a versão do mecanismo do cluster de banco de dados para a versão mais recente a fim de se beneficiar do aumento da capacidade de armazenamento.	Sim	Limites de tamanho do Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Os clusters de banco de dados com todas as instâncias de leitor na mesma zona de disponibilidade.	As zonas de disponibilidade (AZs) são locais distintos entre si para fornecer isolamento em caso de interrupções em cada região da AWS. Recomendamos distribuir a instância primária e as instâncias de leitor no cluster de banco de dados em várias AZs para melhorar a disponibilidade do cluster de banco de dados. É possível criar um cluster multi-AZ usando o Console de Gerenciamento da AWS, a CLI da AWS ou a API do Amazon RDS ao criar o cluster. É possível modificar o cluster existente do Aurora em um cluster multi-AZ adicionando uma nova instância de	O cluster de banco de dados tem todas as instâncias de leitura na mesma zona de disponibilidade. Recomendamos distribuir as instâncias de leitor entre várias zonas de disponibilidade. A distribuição aumenta a disponibilidade e melhora o tempo de resposta reduzindo a latência da rede entre clientes e o banco de dados.	Não	Alta disponibilidade do Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
	leitor e especificando uma AZ diferente.			
Os parâmetros de memória do banco de dados estão divergindo do padrão.	Os parâmetros de memória das instâncias de banco de dados são significativamente diferentes dos valores padrão. Essas configurações podem afetar a performance e causar erros. Recomendamos definir os parâmetros de memória personalizados da instância de banco de dados como os valores padrão no grupo de parâmetros de banco de dados.	Redefina os parâmetros de memória para os valores padrão.	Não	Grupos de parâmetros para Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro de cache de consulta está ativado.	Quando as alterações exigirem que o cache de consultas seja limpo, a instância de banco de dados parecerá paralisada. A maioria das workloads não se beneficia de um cache de consultas. O cache de consultas foi removido do MySQL versão 8.0 e posterior. Recomendamos definir o parâmetro query_cache_type como 0.	Defina o valor do parâmetro `query_cache_type` como `0` nos grupos de parâmetros do banco de dados.	Sim	Grupos de parâmetros para Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `log_output` está definido como tabela.	Quando `log_output` está definido como `TABLE`, mais armazenamento é usado do que quando `log_output` está definido como `FILE`. Recomendamos definir o parâmetro como `FILE` para não atingir o limite de tamanho do armazenamento. Definido como `FILE` por padrão no MySQL 8.4 e versões posteriores.	Defina o valor do parâmetro `log_output` como `FILE` nos grupos de parâmetros do banco de dados.	Não	Arquivos de log do banco de dados AuroraMySQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `autovacuum` está desativado.	O parâmetro autovacuum está desativado para clusters de banco de dados. Desativar o autovacuum aumenta o inchaço da tabela e do índice e afeta a performance. Recomendamos que você ative o autovacuum nos grupos de parâmetros de banco de dados.	Ative o parâmetro autovacuum nos grupos de parâmetros do cluster de banco de dados.	Não	Understanding autovacuum in Amazon RDS for PostgreSQL environments no Blog de bancos de dados da AWS

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `synchronous_commit` está desativado.	Quando o `synchronous_commit` parâmetro é desativado, os dados podem ser perdidos em uma falha no banco de dados. A durabilidade do banco de dados está em risco. Recomendamos que você ative o parâmetro `synchronous_commit` .	Ative o parâmetro `synchronous_commit` nos grupos de parâmetros do banco de dados.	Sim	Parâmetros do Amazon Aurora PostgreSQL: replicação, segurança e registro em log no blog de banco de dados da AWS

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `track_counts` está desativado.	Quando o parâmetro `track_counts` estiver desativado, o banco de dados não coletará as estatísticas de atividade do banco de dados. O autovacuum exige que essas estatísticas funcionem corretamente. Recomendamos que você defina o parâmetro `track_counts` como 1.	Defina o parâmetro `track_counts` como 1.	Não	Estatísticas de tempo de execução do PostgreSQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `enable_indexonlyscan` está desativado.	O planejador ou o otimizador de consultas não pode usar o plano de analisar somente o índice quando ele está desativado. Recomendamos definir o valor do parâmetro `enable_indexonlyscan` como 1.	Defina o valor do parâmetro `enable_indexonlyscan` como 1.	Não	Configuração do método Planner para PostgreSQL
O parâmetro `enable_indexscan` está desativado.	O planejador ou o otimizador de consultas não pode usar o plano de analisar o índice quando ele está desativado. Recomendamos que você defina o valor `enable_indexscan` como 1.	Defina o valor do parâmetro `enable_indexscan` como 1.	Não	Configuração do método Planner para PostgreSQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `innodb_flush_log_at_trx` está desativado.	O valor do parâmetro `innodb_flush_log_at_trx` da instância de banco de dados não é um valor seguro. Esse parâmetro controla a persistência das operações de confirmação no disco. Recomendamos que você defina o parâmetro `innodb_flush_log_at_trx` como 1.	Defina o valor do parâmetro `innodb_flush_log_at_trx` como 1.	Não	Configurar a frequência com que o buffer de log é liberado

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `innodb_stats_persistent` está desativado.	Sua instância de banco de dados não está configurada para manter as estatísticas do InnoDB no disco. Quando as estatísticas não são armazenadas, elas são recalculadas sempre que a instância é reiniciada e a tabela é acessada. Isso causa variações no plano de execução da consulta. Você pode modificar o valor desse parâmetro global no nível da tabela. Recomendamos definir o valor do parâmetro `innodb_stats_persistent` como `ON`.	Defina o valor do parâmetro `innodb_stats_persistent` como `ON`.	Não	Grupos de parâmetros para Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `innodb_open_files` é baixo.	O parâmetro `innodb_open_files` controla o número de arquivos que o InnoDB pode abrir ao mesmo tempo. O InnoDB abre todos os arquivos de log e tablespace do sistema quando o mysqld está em execução. Sua instância de banco de dados tem um valor baixo para o número máximo de arquivos que o InnoDB pode abrir ao mesmo tempo. Recomendamos que você defina o parâmetro `innodb_open_files` com um valor mínimo de 65.	Defina o parâmetro `innodb_open_files` como um valor mínimo de 65.	Sim	Arquivos abertos do InnoDB para MySQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `max_user_connections` é baixo.	Sua instância de banco de dados tem um valor baixo para o número máximo de conexões simultâneas para cada conta de banco de dados. Recomendamos definir o parâmetro `max_user_connections` como um número maior que 5.	Aumente o valor do parâmetro `max_user_connections` para um número maior que 5.	Sim	Definir limites de recursos da conta para MySQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
As réplicas de leitura são abertas no modo gravável.	A instância de banco de dados tem uma réplica de leitura no modo de gravação, que permite que os clientes realizem atualizações. Recomendamos que você defina o parâmetro `read_only` como `TrueIfReplica` para que as réplicas de leitura não estejam no modo gravável.	Defina o valor do parâmetro `read_only` como `TrueIfReplica` .	Não	Grupos de parâmetros para Amazon Aurora

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
A configuração do parâmetro `innodb_default_row_format` não é segura.	A instância de banco de dados encontra um problema conhecido: uma tabela criada em uma versão do MySQL inferior à 8.0.26 com o valor `row_format` definido como `COMPACT` ou `REDUNDANT` ficará inacessível e irrecuperável se o índice exceder 767 bytes. Recomendamos definir o valor do parâmetro `innodb_default_row_format` como `DYNAMIC`.	Defina o valor do parâmetro `innodb_default_row_format` como `DYNAMIC`.	Não	Alterações feitas no MySQL 8.0.26

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
O parâmetro `general_logging` está ativado.	O registro em log geral é ativado para a instância de banco de dados. Essa configuração é útil para solucionar problemas no banco de dados. No entanto, ativar o registro em log geral aumenta a quantidade de operações de E/S e o espaço de armazenamento alocado, o que pode causar contenção e degradação da performance. Confira os requisitos para uso do registro em log geral. Recomendamos definir o valor do parâmetro `general_logging` como `0`.	Confira os requisitos para uso do registro em log geral. Se não for obrigatório, recomendamos definir o valor do parâmetro `general_logging` como `0`.	Não	Visão geral dos logs de banco de dados do Aurora MySQL

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Cluster de banco de dados subprovisionado para workload de leitura	Recomendamos adicionar uma instância de banco de dados de leitor ao cluster de banco de dados com a mesma classe e tamanho que a instância de banco de dados de gravador no cluster. A configuração atual tem uma instância de banco de dados com uma carga de banco de dados continuamente alta causada principalmente por operações de leitura. Distribua essas operações adicionando outra instância de banco de dados ao cluster e direcionando a workload de leitura para o endpoint somente leitura do cluster de banco de dados.	Adicione uma instância de banco de dados de leitor ao cluster.	Não	Adicionar réplicas do Aurora a um cluster de banco de dados Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora Preços do Amazon RDS

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Instância do RDS subprovisionada para a capacidade de memória do sistema	Recomendamos ajustar as consultas para usar menos memória ou usar um tipo de instância de banco de dados com maior memória alocada. Quando a instância está com pouca memória, a performance do banco de dados é afetada.	Usar uma instância de banco de dados com maior capacidade de memória	Sim	Escalar a instância do Amazon RDS vertical e horizontalmente no blog de bancos de dados da AWS Tipos de instância do Amazon RDS Preços do Amazon RDS

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Instância do RDS subprovisionada para a capacidade de CPU do sistema	Recomendamos ajustar as consultas para utilizar menos CPU ou modificar a instância de banco de dados para utilizar uma classe de instância de banco de dados com mais vCPUs alocadas. A performance do banco de dados pode diminuir quando uma instância de banco de dados está com pouca CPU.	Usar uma instância de banco de dados com maior capacidade de CPU	Sim	Escalar a instância do Amazon RDS vertical e horizontalmente no blog de bancos de dados da AWS Tipos de instância do Amazon RDS Preços do Amazon RDS

Tipo	Descrição	Recomendação	Tempo de inatividade necessário	Mais informações
Os recursos do RDS não estão utilizando o grupo de conexões corretamente	Recomendamos habilitar o Amazon RDS Proxy para agrupar e compartilhar com eficiência as conexões de banco de dados existentes. Se você já estiver usando um proxy para o banco de dados, configure-o corretamente para melhorar o grupo de conexões e o balanceamento de carga em várias instâncias de banco de dados. O RDS Proxy pode ajudar a reduzir o risco de esgotamento da conexão e o tempo de inatividade, enquanto melhora a disponibilidade e a escalabilidade.	Habilitar o RDS Proxy ou modificar a configuração de proxy existente	Não	Escalar a instância do Amazon RDS vertical e horizontalmente no blog de bancos de dados da AWS Amazon RDS Proxypara Aurora Preços do Amazon RDS Proxy

Visualizar métricas no console do Amazon RDS

O Amazon RDS se integra ao Amazon CloudWatch para exibir várias métricas de cluster de banco de dados Aurora no console do RDS. Algumas métricas são aplicadas no nível do cluster, enquanto outras são aplicadas no nível da instância. As descrições das métricas no nível da instância e do cluster estão disponíveis em Referência de métricas do Amazon Aurora.

Para o cluster de banco de dados do Aurora, as seguintes categorias de métricas são monitoradas:

- CloudWatch: mostra as métricas do Amazon CloudWatch relacionadas ao Aurora disponíveis no console do RDS. Também é possível visualizá-las no console do CloudWatch. Cada métrica inclui um gráfico que mostra a métrica monitorada durante um período específico. Veja uma lista completa de métricas do CloudWatch em Métricas do Amazon CloudWatch para o Amazon Aurora.
- Monitoramento avançado: mostra um resumo das métricas do sistema operacional quando o monitoramento avançado de seu cluster de banco de dados Aurora está ativado. O RDS fornece as métricas do monitoramento avançado à sua conta do Amazon CloudWatch Logs. Cada métrica de sistema operacional inclui um gráfico que mostra a métrica monitorada durante um período específico. Para obter uma visão geral, consulte Monitorar métricas do SO com o monitoramento avançado. Veja uma lista das métricas do monitoramento avançado em Métricas do sistema operacional no monitoramento avançado.
- OS Process list (Lista de processos de SO): mostra os detalhes de cada processo em execução no cluster de banco de dados selecionado.
- Performance Insights: abre o painel do Amazon RDS Performance Insights relacionado a uma instância de banco de dados em seu cluster de banco de dados Aurora. O Performance Insights não tem suporte no nível do cluster. Para ter uma visão geral do Performance Insights, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora. Veja uma lista das métricas do Performance Insights em Métricas do Amazon CloudWatch para Insights de Performance do Amazon RDS.

O Amazon RDS agora fornece uma visão consolidada das métricas do Insights de Performance e do CloudWatch no painel do Insights de Performance. O Insights de Performance deve estar ativado para que o cluster de banco de dados use essa visualização. Você pode escolher a nova visualização de monitoramento na guia Monitoramento ou Insights de Performance no painel de navegação. Para ver as instruções para escolher essa visualização, consulte Visualizar métricas combinadas com o painel do Insights de Performance.

Visualizar métricas combinadas com o painel do Insights de Performance

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.

Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.

Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

O Amazon RDS fornece uma visão consolidada das métricas do Insights de Performance e do CloudWatch para sua instância de banco de dados no painel do Insights de Performance. É possível usar o painel pré-configurado ou criar um painel personalizado. O painel pré-configurado fornece as métricas mais usadas para ajudar a diagnosticar problemas de performance em um mecanismo de banco de dados. Como alternativa, é possível criar um painel do personalizado com as métricas de um mecanismo de banco de dados que atenda aos seus requisitos de análise. Depois, use esse

painel para todas as instâncias de banco de dados desse tipo de mecanismo de banco de dados em sua conta da AWS.

Você pode escolher a visualização de monitoramento na guia Monitoramento ou Insights de Performance no painel de navegação.

O Insights de Performance deve estar ativado para que o cluster de banco de dados visualize as métricas combinadas no painel do Insights de Performance. Para obter mais informações sobre como ativar o Insights de Performance, consulte Ativar e desativar o Insights de Performance para o Aurora.

Nas seções a seguir, é possível aprender a exibir as métricas do Insights de Performance e do CloudWatch.

Tópicos

- Escolher a nova visualização de monitoramento na guia Monitoramento
- Escolher a nova visualização de monitoramento na página do Insights de Performance
- Criar um painel personalizado com o Insights de Performance
- Escolher o painel predefinido com o Insights de Performance

Escolher a nova visualização de monitoramento na guia Monitoramento

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com

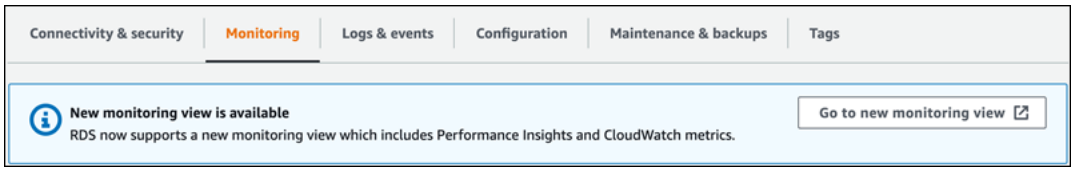
o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

No console do Amazon RDS, é possível escolher a nova visualização de monitoramento para visualizar as métricas do Insights de Performance e do CloudWatch para sua instância de banco de dados.

Como escolher a nova visualização de monitoramento na guia Monitoramento

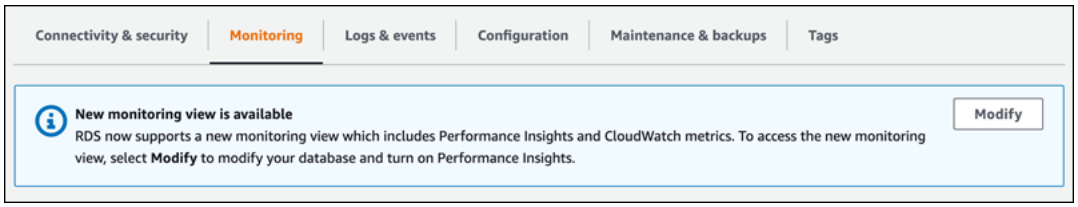
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação à esquerda, selecione Bancos de dados.
3. Escolha o cluster de banco de dados do Aurora que você deseja monitorar.
4. Role para baixo e escolha a guia Monitoramento.

   Um banner aparece com a opção de escolher a nova visualização de monitoramento. O exemplo a seguir mostra o banner para escolher a nova visualização de monitoramento.



5. Escolha Ir para a nova visualização de monitoramento para abrir o painel do Insights de Performance com métricas do Insights de Performance e do CloudWatch para o cluster de banco de dados.
6. (Opcional) Se o Insights de Performance estiver desativado para a instância de banco de dados, um banner aparecerá com a opção de modificar a instância de banco de dados e ativar o Insights de Performance.

O exemplo a seguir mostra o banner para modificar a instância de banco de dados na guia Monitoramento.



Escolha Modificar para modificar o cluster de banco de dados e ativar o Insights de Performance. Para obter mais informações sobre como ativar o Insights de Performance, consulte Ativar e desativar o Insights de Performance para o Aurora

Escolher a nova visualização de monitoramento na página do Insights de Performance

Important

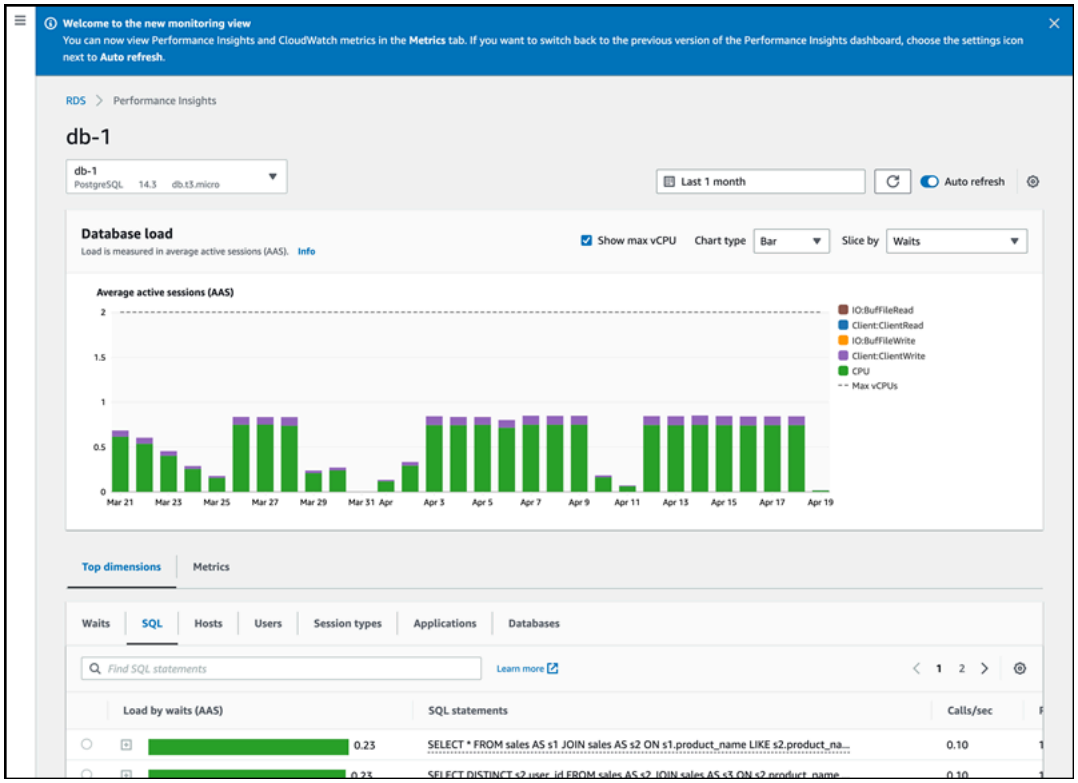
A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande

escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

No console do Amazon RDS, é possível escolher a nova visualização de monitoramento para visualizar as métricas do Insights de Performance e do CloudWatch para sua instância de banco de dados.

Como escolher a nova visualização de monitoramento com o Insights de Performance no painel de navegação

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados para visualizar o painel do Insights de Performance que mostra as métricas do Insights de Performance e do CloudWatch para a instância de banco de dados.



Criar um painel personalizado com o Insights de Performance

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

Na nova visualização de monitoramento, é possível criar um painel personalizado com as métricas necessárias para atender aos seus requisitos de análise.

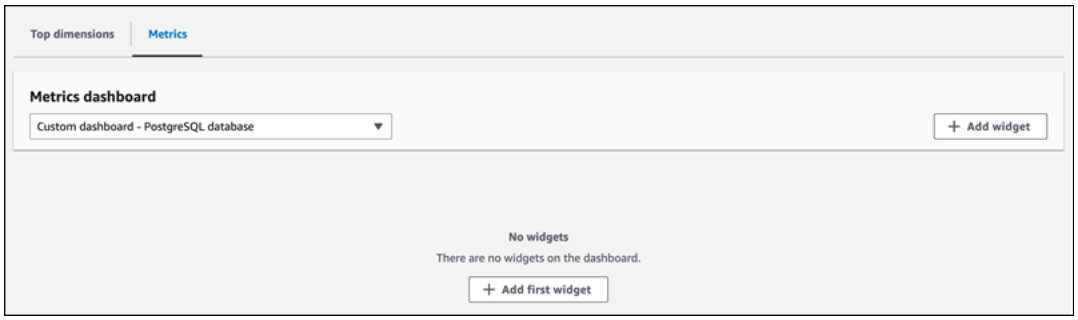
É possível criar um painel personalizado selecionando métricas do Insights de Performance e do CloudWatch para sua instância de banco de dados. É possível usar esse painel personalizado para outras instâncias de banco de dados do mesmo tipo de mecanismo de banco de dados em sua conta da AWS.

> **Note**
>
> O painel personalizado permite até 50 métricas.

Use o menu de configurações do widget para editar ou excluir o painel e mover ou redimensionar a janela do widget.

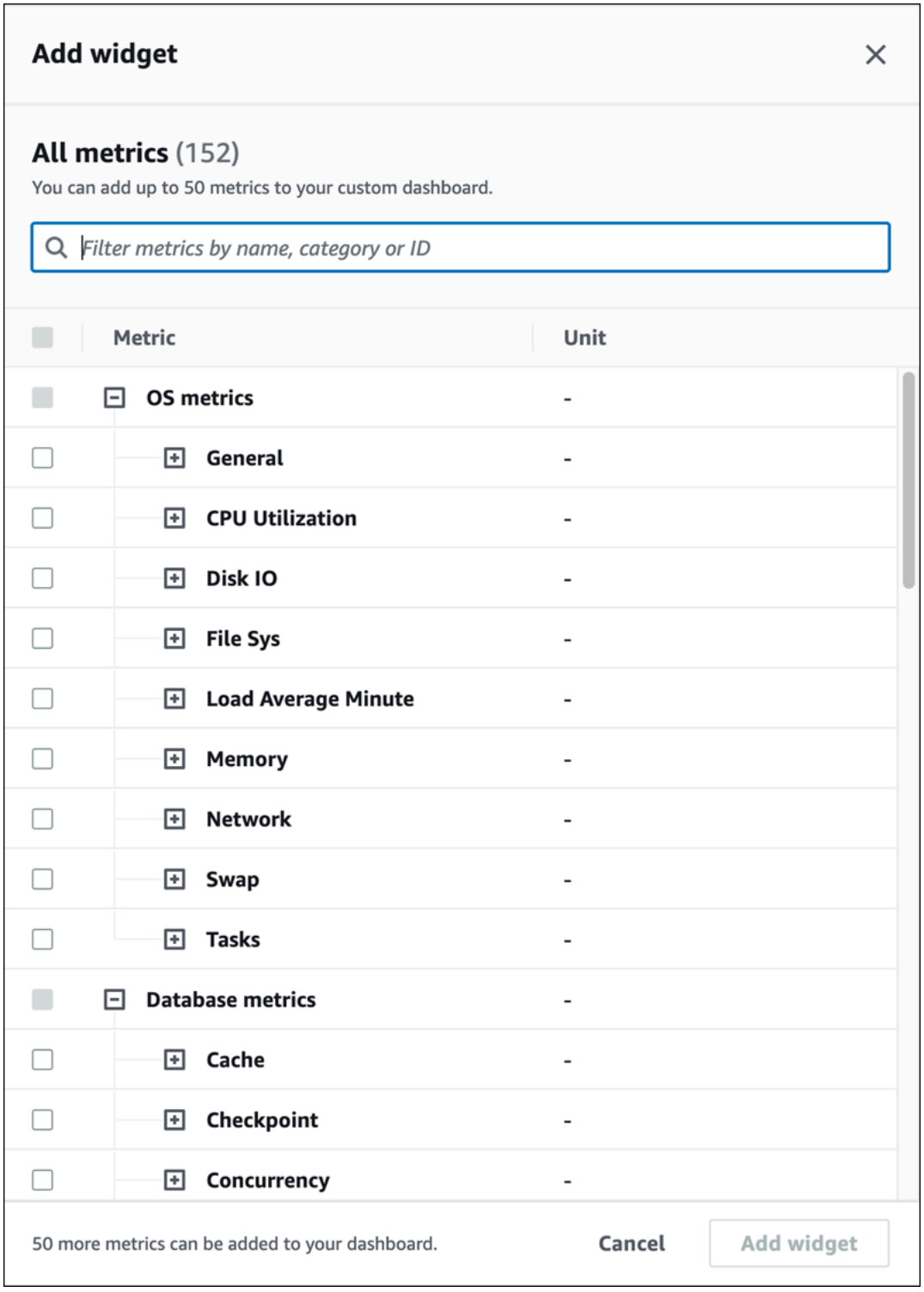
Como criar um painel personalizado com o Insights de Performance no painel de navegação

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.
4. Role para baixo até a guia Métricas na janela.
5. Selecione o painel personalizado na lista suspensa. O exemplo a seguir mostra a criação do painel personalizado.



6. Escolha Adicionar widget para abrir a janela Adicionar widget. É possível abrir e visualizar as métricas do sistema operacional (SO) disponíveis, as métricas do banco de dados e as métricas do CloudWatch na janela.

   O exemplo a seguir mostra a janela Adicionar widget com as métricas.



7. Selecione as métricas que você deseja visualizar no painel e escolha Adicionar widget. É possível usar o campo de pesquisa para encontrar uma métrica específica.

   As métricas selecionadas aparecem no painel.

8. (Opcional) Se você quiser modificar ou excluir o painel, escolha o ícone de configurações no canto superior direito do widget e selecione uma das ações a seguir no menu.

   - Editar modifique a lista de métricas na janela. Escolha Atualizar widget depois de selecionar as métricas para o painel.
   - Excluir: exclui o widget. Selecione Excluir na janela de confirmação.

Escolher o painel predefinido com o Insights de Performance

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
> Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
> Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

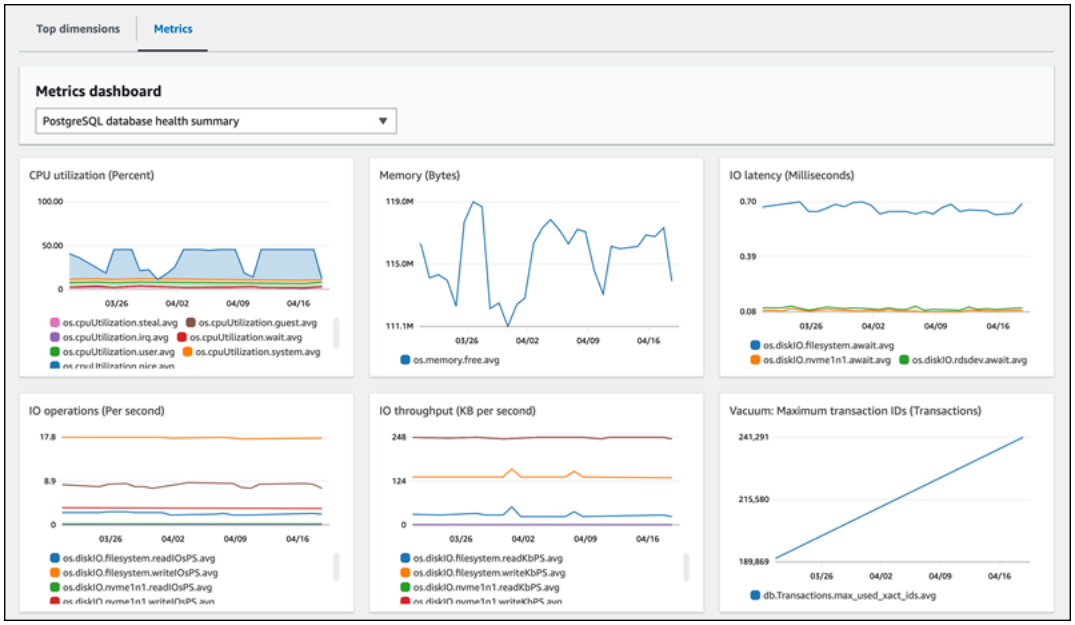
É possível visualizar as métricas mais usadas com o painel pré-configurado. Esse painel ajuda a diagnosticar problemas de performance com um mecanismo de banco de dados e a reduzir o tempo médio de recuperação de horas para minutos.

> **Note**
>
> Esse painel não pode ser editado.

Como escolher o painel predefinido com o Insights de Performance no painel de navegação

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.
4. Role para baixo até a guia Métricas na janela
5. Selecione um painel pré-configurado na lista suspensa.

   É possível visualizar as métricas da instância de banco de dados no painel. O exemplo a seguir mostra um painel de métricas pré-configurado.



Monitorar métricas do Amazon Aurora com o Amazon CloudWatch

O Amazon CloudWatch é um repositório de métricas. O repositório coleta e processa dados brutos do Amazon Aurora como métricas legíveis, quase em tempo real. Para obter uma lista completa de métricas do Amazon Aurora enviadas ao CloudWatch, consulte Referência de métricas para o Amazon Aurora.

Para analisar e solucionar problemas de performance de banco de dados em grande escala, use CloudWatch Database Insights.

Tópicos

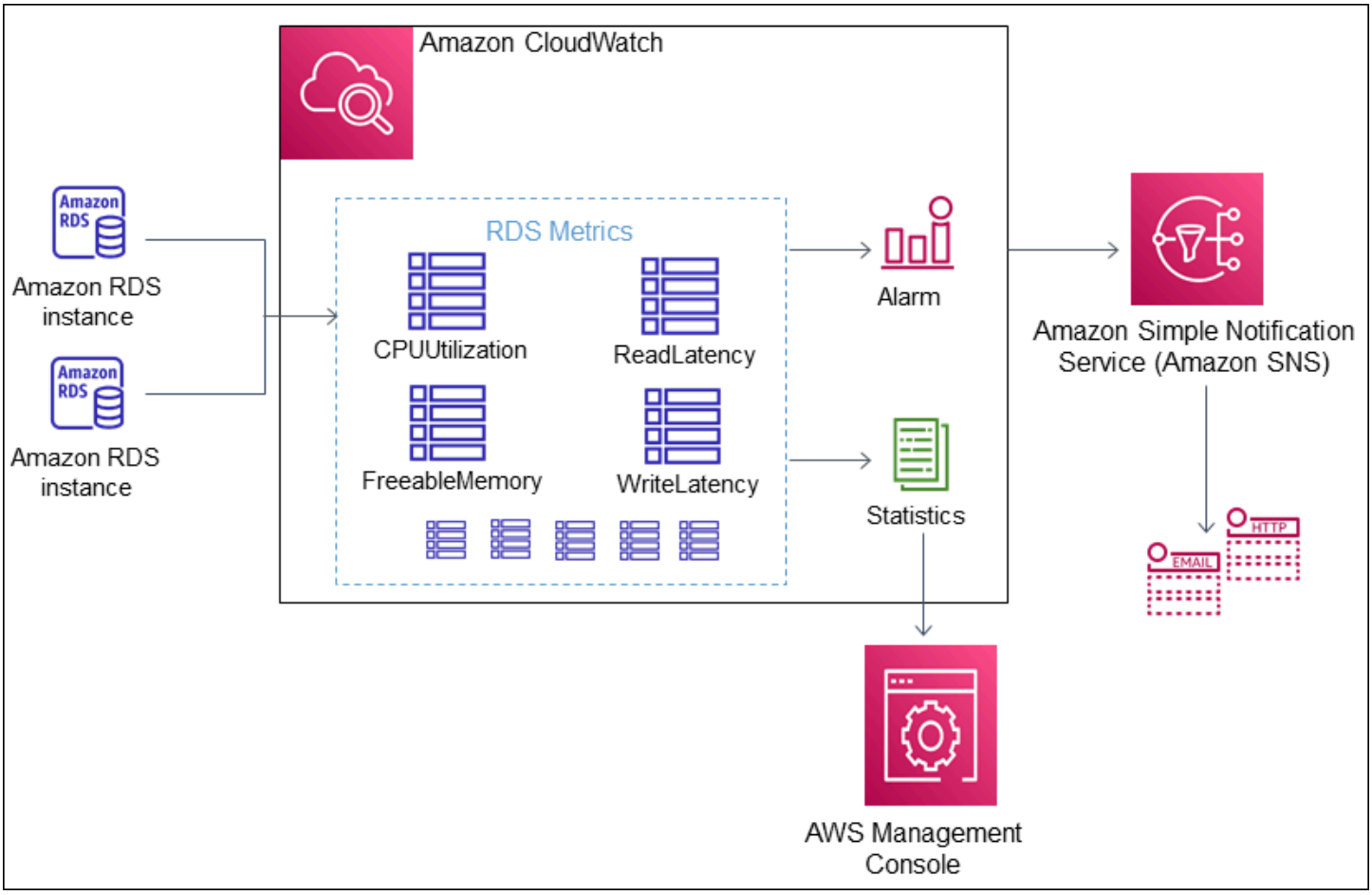
- Visão geral do Amazon Aurora e do Amazon CloudWatch
- Visualizar métricas de cluster de de banco de dados no console do CloudWatch e na AWS CLI.
- Exportar as métricas do Performance Insights para o CloudWatch
- Criação de alarmes CloudWatch para monitorar Amazon Aurora

Visão geral do Amazon Aurora e do Amazon CloudWatch

Por padrão, o Amazon Aurora envia dados de métrica automaticamente para o CloudWatch em períodos de um minuto. Por exemplo, a métrica `CPUUtilization` registra a porcentagem de utilização da CPU para uma instância de banco de dados ao longo do tempo. Pontos de dados com um período de 60 segundos (1 minuto) ficam disponíveis por 15 dias. Isso significa que você pode acessar informações históricas e ter uma perspectiva melhor sobre a performance da aplicação ou do serviço web.

Agora você pode exportar painéis de métricas do Performance Insights do Amazon RDS para o Amazon CloudWatch. Você pode exportar os painéis de métricas pré-configurados ou personalizados como um novo painel ou adicioná-los a um painel existente do CloudWatch. O painel exportado está disponível para visualização no console do CloudWatch. Para obter mais informações sobre como exportar os painéis de métricas do Performance Insights para o CloudWatch, consulte. Exportar as métricas do Performance Insights para o CloudWatch

Conforme mostrado no diagrama a seguir, você pode configurar alarmes para suas métricas do CloudWatch. Por exemplo, é possível criar um alarme que sinalize quando a utilização da CPU para uma instância é superior a 70%. Você pode configurar o Amazon Simple Notification Service para enviar um e-mail quando o limite for ultrapassado.



O Amazon RDS publica os seguintes tipos de métrica no Amazon CloudWatch:

- Métricas do Aurora em nível de cluster e instância

  Para obter uma tabela dessas métricas, consulte Métricas do Amazon CloudWatch para o Amazon Aurora.

- Métricas do Performance Insights

  Para obter uma tabela dessas métricas, consulte Métricas do Amazon CloudWatch para Insights de Performance do Amazon RDS e Métricas de contadores do Performance Insights.

- Métricas de monitoramento aprimoradas (publicadas no Amazon CloudWatch Logs)

  Para obter uma tabela dessas métricas, consulte Métricas do sistema operacional no monitoramento avançado.

- Métricas de uso para as cotas de serviço do Amazon RDS em sua Conta da AWS

Para obter uma tabela dessas métricas, consulte Métricas de uso do Amazon CloudWatch para Amazon Aurora. Para obter mais informações sobre cotas do Amazon RDS, consulte Cotas e restrições do Amazon Aurora.

Para obter mais informações sobre o CloudWatch, consulte O que é o Amazon CloudWatch? no Guia do usuário do Amazon CloudWatch. Para obter mais informações sobre a retenção de métricas do CloudWatch, consulte Retenção de métricas.

Visualizar métricas de cluster de de banco de dados no console do CloudWatch e na AWS CLI.

Veja a seguir detalhes sobre como visualizar métricas de sua instância de banco de dados usando o CloudWatch. Para obter informações sobre o monitoramento de métricas para o sistema operacional da sua instância de banco de dados em tempo real usando o CloudWatch Logs, consulte Monitorar métricas do SO com o monitoramento avançado.

Quando são usados recursos do Amazon Aurora, o Amazon Aurora envia métricas e dimensões ao Amazon CloudWatch a cada minuto.

Agora você pode exportar painéis de métricas do Performance Insights do Amazon RDS para o Amazon CloudWatch e visualizar essas métricas no console do CloudWatch. Para obter mais informações sobre como exportar os painéis de métricas do Performance Insights para o CloudWatch, consulte. Exportar as métricas do Performance Insights para o CloudWatch

É possível usar os procedimentos a seguir para visualizar as métricas do Amazon Aurora no console do CloudWatch e na CLI.

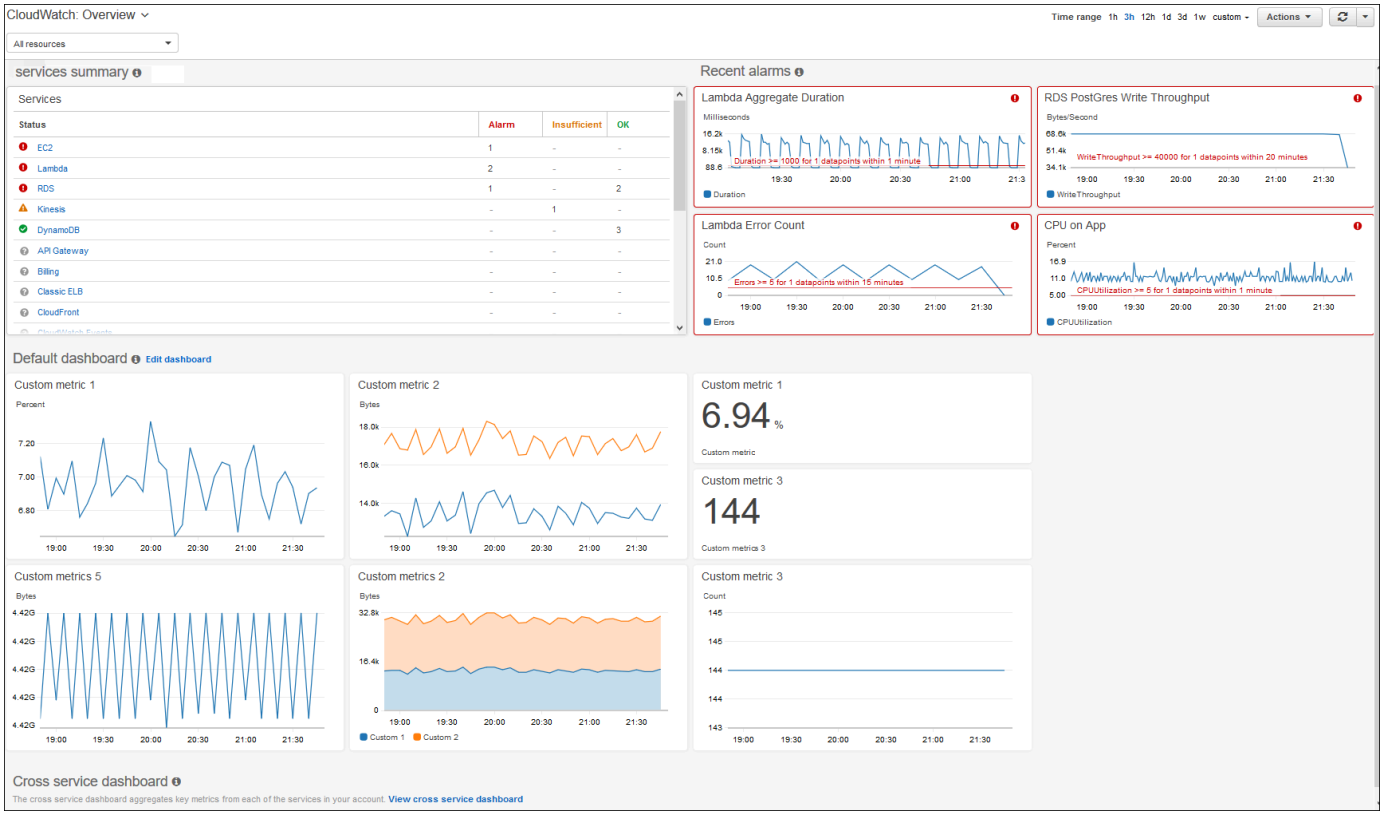
Console

Para visualizar as métricas usando o console do Amazon CloudWatch

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

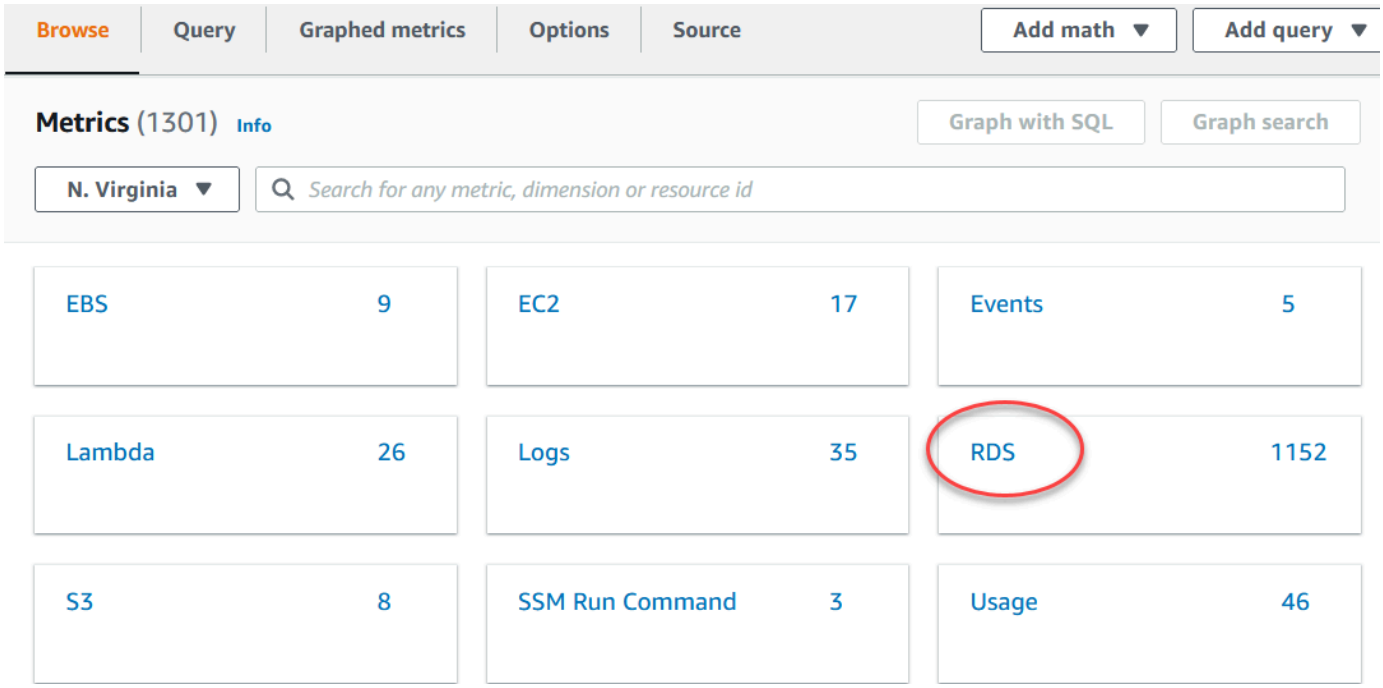
1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.

   A página inicial de visão geral do CloudWatch é exibida.



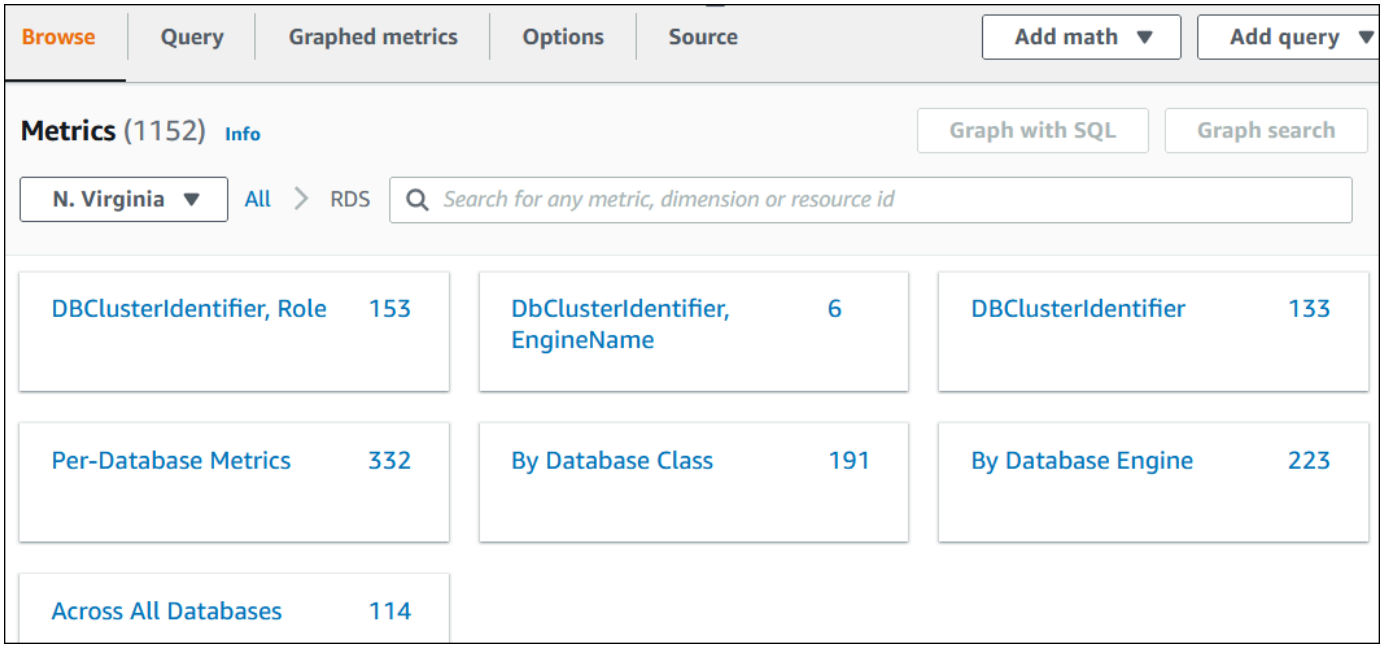
2. Se necessário, altere a Região da AWS. Na barra de navegação, escolha a Região da AWS na qual seus recursos da AWS estão localizados. Para obter mais informações, consulte Regiões e endpoints.

3. No painel de navegação, escolha Metrics (Métricas) e, em seguida, All metrics (Todas as métricas).

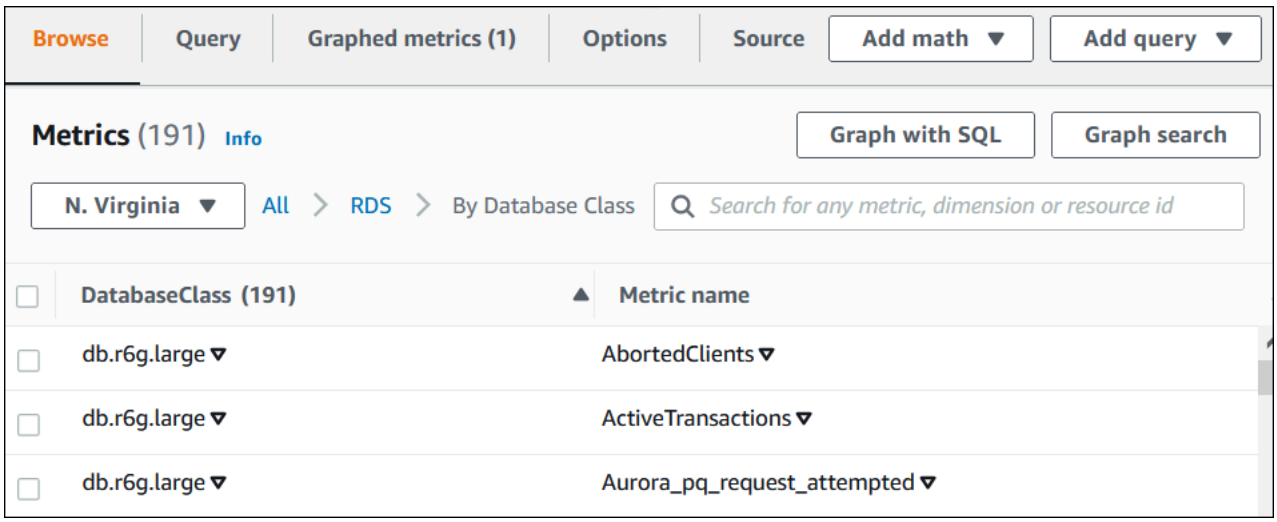


4. Role para baixo e escolha o namespace de métrica RDS.

   A página exibe as dimensões do Amazon Aurora. Para obter descrições dessas dimensões, consulte Dimensões do Amazon CloudWatch para o Aurora.



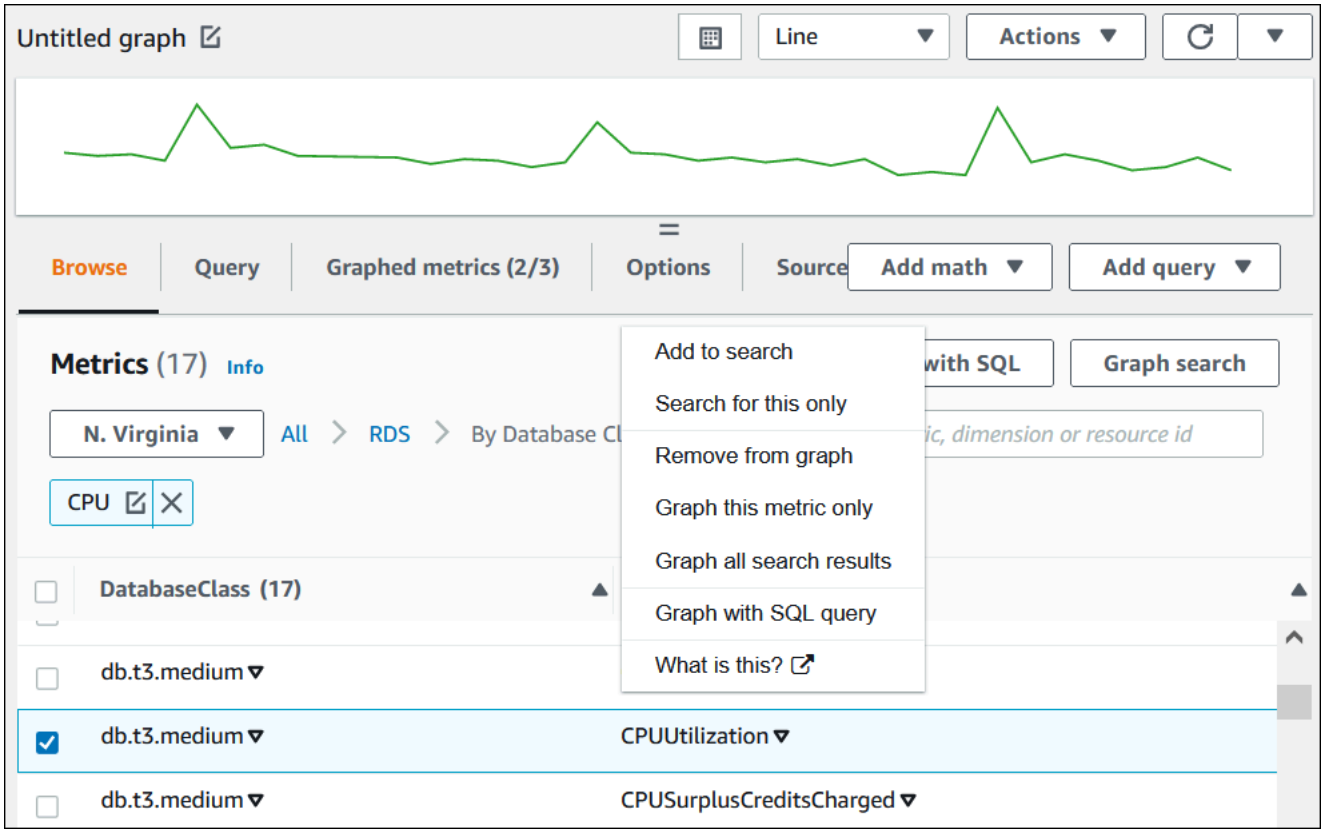
5. Escolha uma dimensão de métrica; por exemplo, By Database Class (Por classe de banco de dados).



6. Realize uma destas ações:

   - Para classificar a métrica, use o cabeçalho da coluna.
   - Para criar um gráfico de uma métrica, marque a caixa de seleção ao lado da métrica.
   - Para filtrar por recurso, escolha o ID do recurso e Add to search (Adicionar à pesquisa).
   - Para filtrar por métrica, escolha o nome da métrica e Add to search (Adicionar à pesquisa).

   O exemplo a seguir filtra com base na classe db.t3.medium e faz um gráfico da métrica CPUUtilization.



Veja detalhes sobre como analisar o uso de recursos do Aurora PostgreSQL usando métricas do CloudWatch. Para obter mais informações, consulte . Usar métricas do Amazon CloudWatch para analisar o uso de recursos do Aurora PostgreSQL

AWS CLI

Para obter informações sobre métricas usando a AWS CLI, use o comando `list-metrics` do CloudWatch. No exemplo a seguir, você lista todas as métricas no namespace `AWS/RDS`.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Para receber os dados de métricas, use o comando `get-metric-data`.

O exemplo a seguir obtém estatísticas `CPUUtilization` para a instância `my-instance` durante o período específico de 24 horas, com granularidade de 5 minutos.

Crie um arquivo JSON `CPU_metric.json` com o conteúdo apresentado a seguir.

```
{
```

```
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Para Linux, macOS ou Unix:

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Para Windows:

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

A saída da amostra é exibida da seguinte maneira:

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
```

```
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Para obter mais informações, consulte Obter as estatísticas de uma métrica no Guia do usuário do Amazon CloudWatch.

Exportar as métricas do Performance Insights para o CloudWatch

O Performance Insights permite que você exporte o painel de métricas pré-configurado ou personalizado da sua instância de banco de dados para o Amazon CloudWatch. Você pode exportar o painel de métricas como um novo painel ou adicioná-lo a um painel existente do CloudWatch. Ao optar por adicionar o painel a um painel existente do CloudWatch, você pode criar um rótulo de cabeçalho para que as métricas apareçam em uma seção separada no painel do CloudWatch.

Você pode visualizar as métricas exportadas no console do CloudWatch. Se você adicionar novas métricas a um painel de métricas do Performance Insights depois de exportá-lo, deverá exportá-lo novamente para visualizar as novas métricas no console do CloudWatch.

Você também pode selecionar um widget de métrica no painel do Performance Insights e visualizar os dados de métricas no console do CloudWatch.

Para obter mais informações sobre como exibir métricas de replicação no console do CloudWatch, consulte Visualizar métricas de cluster de de banco de dados no console do CloudWatch e na AWS CLI..

Nas seções a seguir, exporte as métricas do Insights de Performance para o CloudWatch como um painel novo ou existente e visualize as métricas do Insights de Performance no CloudWatch.

Tópicos

- Exportar métricas do Performance Insights como um novo painel para o CloudWatch

- Adicionar as métricas do Performance Insights a um painel existente do CloudWatch
- Visualizar um widget de métrica do Performance Insights no CloudWatch

Exportar métricas do Performance Insights como um novo painel para o CloudWatch

Escolha um painel de métricas pré-configurado ou personalizado no painel do Performance Insights e exporte-o como um novo painel no CloudWatch. Você pode visualizar o painel exportado no console do CloudWatch.

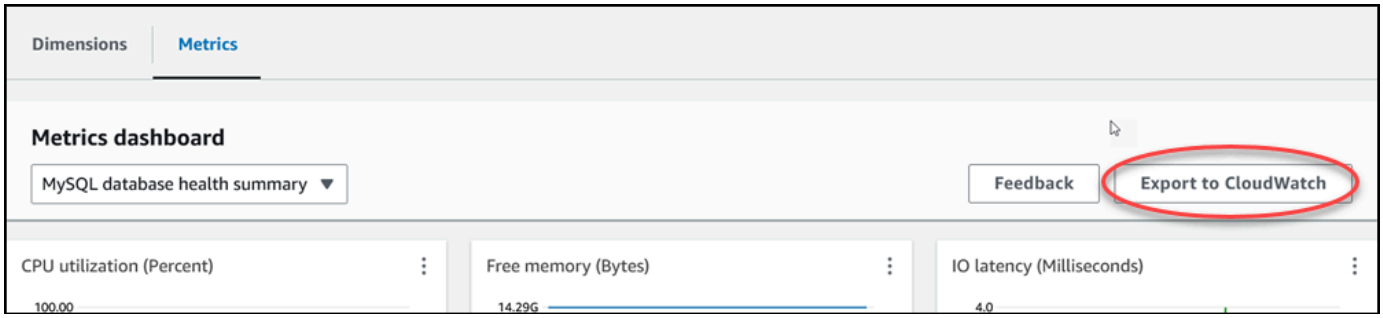
Para exportar métricas do Performance Insights como um novo painel no CloudWatch

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

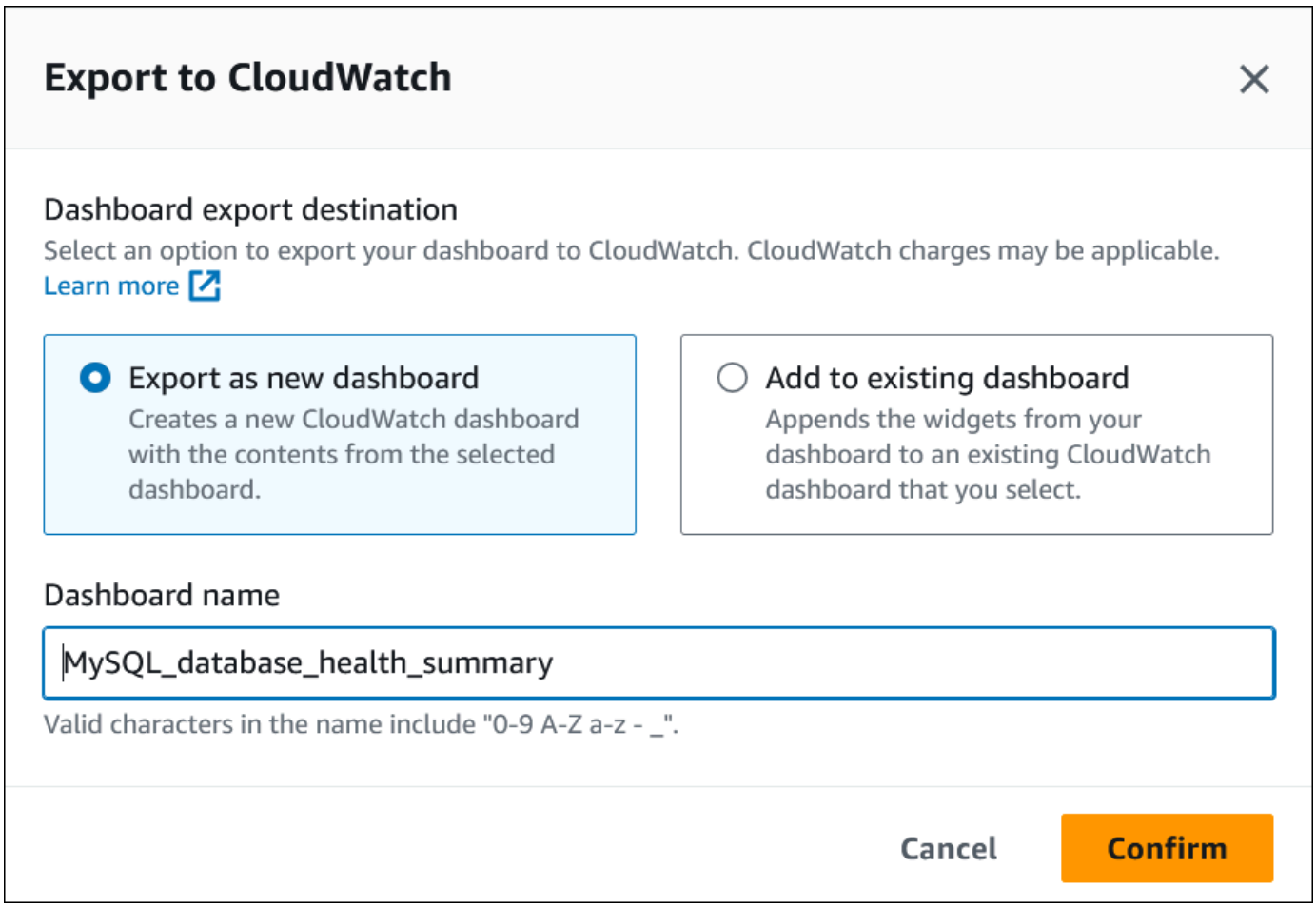
   O painel do Insights de Performance é exibido para a instância de banco de dados.
4. Role para baixo e escolha Métricas.

   Por padrão, é exibido o painel pré-configurado com as métricas do Performance Insights.
5. Escolha um painel pré-configurado ou personalizado e, em seguida, escolha Exportar para o CloudWatch.

   A janela Exportar para o CloudWatch é exibida.

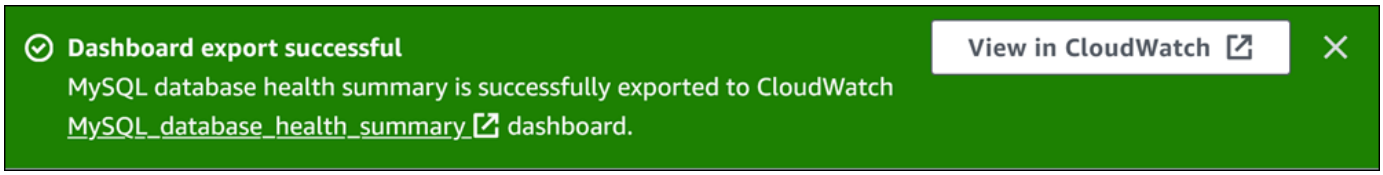


6. Escolha Exportar como novo painel.



7. Insira um nome para o novo painel no campo Nome do painel e escolha Confirmar.

   Um banner exibe uma mensagem após a exportação do painel ser bem-sucedida.



8. Escolha o link ou Exibir no CloudWatch no banner para visualizar o painel de métricas no console do CloudWatch.

Adicionar as métricas do Performance Insights a um painel existente do CloudWatch

Adicione um painel de métricas pré-configurado ou personalizado a um painel existente do CloudWatch. Você pode adicionar um rótulo ao painel de métricas para aparecer em uma seção separada no painel do CloudWatch.

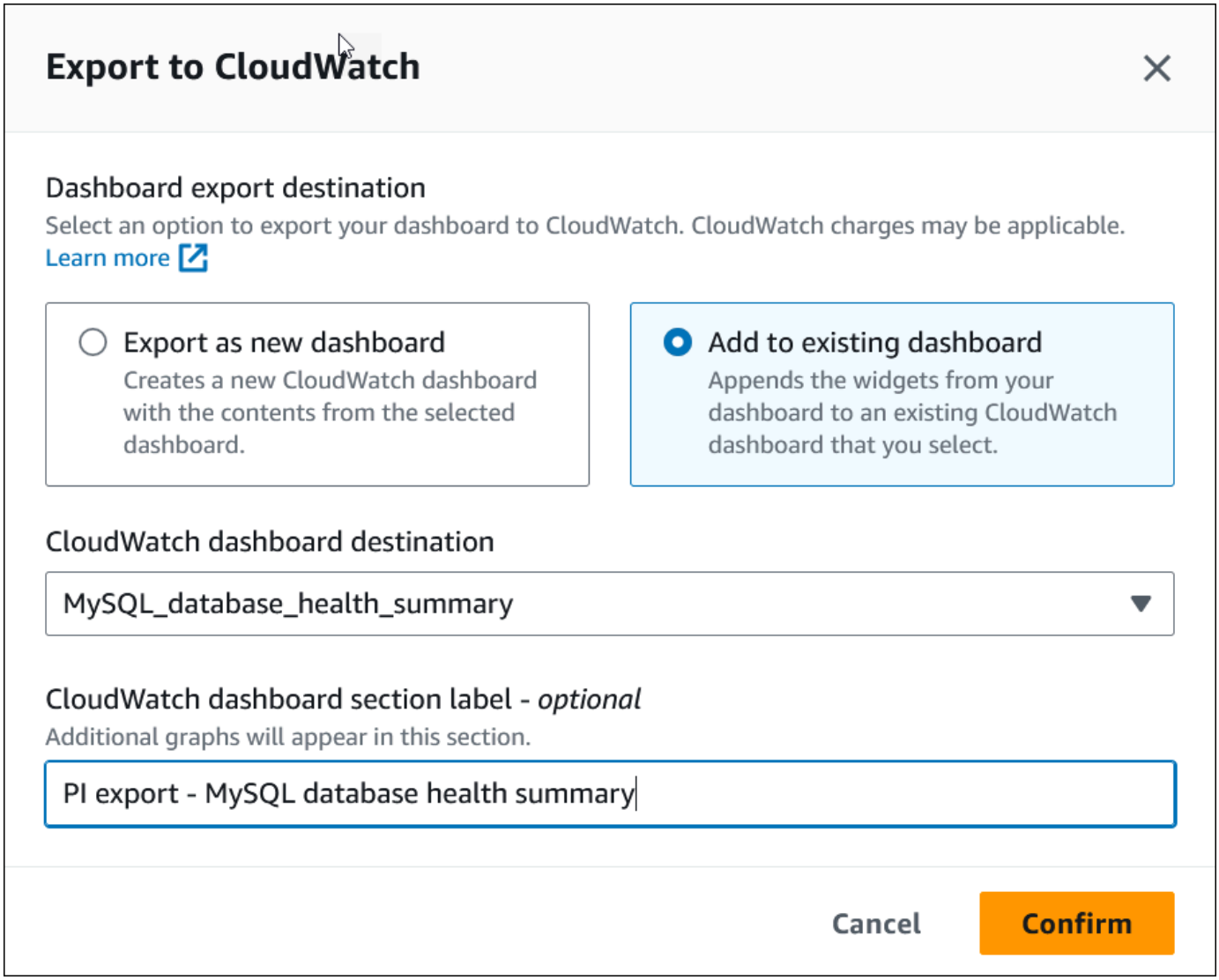
Para exportar as métricas para um painel existente do CloudWatch

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

   O painel do Insights de Performance é exibido para a instância de banco de dados.
4. Role para baixo e escolha Métricas.

   Por padrão, é exibido o painel pré-configurado com as métricas do Performance Insights.
5. Escolha o painel pré-configurado ou personalizado e, em seguida, escolha Exportar para o CloudWatch.

   A janela Exportar para o CloudWatch é exibida.
6. Escolha Adicionar ao painel existente.



7. Especifique o destino e o rótulo do painel e escolha Confirmar.

   - Destino do painel do CloudWatch: escolha um painel existente do CloudWatch.
   - Rótulo da seção do painel do CloudWatch - opcional: insira um nome para as métricas do Performance Insights que aparecerão nesta seção no painel do CloudWatch.

   Um banner exibe uma mensagem após a exportação do painel ser bem-sucedida.

8. Escolha o link ou Exibir no CloudWatch no banner para visualizar o painel de métricas no console do CloudWatch.

Visualizar um widget de métrica do Performance Insights no CloudWatch

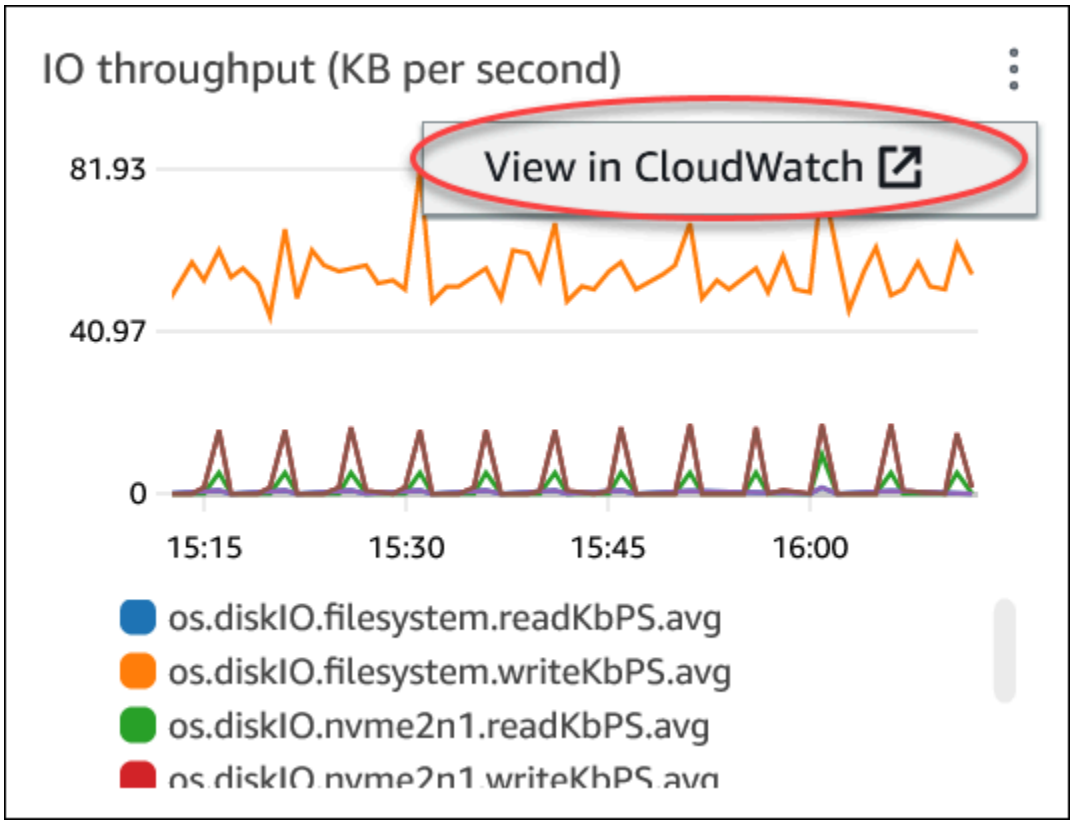
Selecione um widget de métrica no painel do Performance Insights do Amazon RDS e visualize os dados de métricas no console do CloudWatch.

Para exportar um widget de métrica e visualizar os dados das métricas no console do CloudWatch

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

   O painel do Insights de Performance é exibido para a instância de banco de dados.
4. Role para baixo até Métricas.

   Por padrão, é exibido o painel pré-configurado com as métricas do Performance Insights.
5. Escolha um widget de métrica e, em seguida, escolha Exibir no CloudWatch no menu.



   Os dados da métrica são exibidos no console do CloudWatch.

Criação de alarmes CloudWatch para monitorar Amazon Aurora

Você pode criar um alarme do CloudWatch que envia uma mensagem do Amazon SNS quando o alarme muda de estado. Um alarme observa uma única métrica por um período tempo que você especifica. O alarme também pode realizar uma ou mais ações com base no valor da métrica relativa a um limite especificado durante vários períodos. A ação é uma notificação enviada para um tópico do Amazon SNS ou por uma política do Amazon EC2 Auto Scaling.

Os alertas invocam ações apenas para alterações de estado mantidas. Os alarmes do CloudWatch não invocam ações somente porque estão em um determinado estado. O estado deve ter sido alterado e mantido por um período especificado.

> **Note**
>
> Para o Aurora, use métricas da função `WRITER` ou `READER` para configurar alarmes em vez de depender de métricas de instâncias de banco de dados específicas. As funções de instância de bancos de dados Aurora podem ser alteradas com o tempo. Você pode encontrar essas métricas baseadas em função no console do CloudWatch.
> O Auto Scaling do Aurora define automaticamente alarmes com base em métricas da função `READER`. Para obter mais informações sobre o Auto Scaling do Aurora, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.

É possível usar a função matemática métrica DB_PERF_INSIGHTS no console do CloudWatch para consultar métricas do contador do Amazon RDS do Insights de Performance. A função DB_PERF_INSIGHTS também inclui a métrica DBLoad em intervalos de menos de um minuto. Também é possível definir alarmes do CloudWatch para essas métricas.

Para obter mais detalhes sobre como criar um alarme, consulte Crie um alarme para as métricas do contador do Performance Insights a partir de um banco de dados da AWS.

Para definir um alarme usando a AWS CLI

- Chame `put-metric-alarm`. Para obter mais informações, consulte Referência de comandos da AWS CLI.

Para definir um alarme usando a API do CloudWatch

- Chame `PutMetricAlarm`. Para obter mais informações, consulte a Referência da API do Amazon CloudWatch.

Para obter mais informações sobre a definição de tópicos do Amazon SNS e a criação de alarmes, consulte Usar alarmes do Amazon CloudWatch.

Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights

Monitore a carga do banco de dados da sua frota de instâncias de banco de dados do Amazon Aurora com o Database Insights. A carga do banco de dados avalia o nível de atividade de sessão no banco de dados. Você pode usar o Database Insights para analisar e solucionar problemas de performance de seus bancos de dados do Amazon Aurora em grande escala.

Com o Database Insights, você pode visualizar a carga do banco de dados na sua frota e filtrá-la por esperas, instruções SQL, hosts ou usuários.

Por padrão, o RDS habilita o modo Padrão do Database Insights para os bancos de dados do Amazon Aurora. Quando você ativa o modo Avançado do Database Insights para um cluster de banco de dados, o RDS habilita o Database Insights para cada instância de banco de dados no cluster.

Para obter informações sobre o uso do Database Insights no console do Amazon CloudWatch, consulte CloudWatch Database Insights no Guia do usuário do Amazon CloudWatch.

Preços

Para obter informações sobre preços, consulte Amazon CloudWatch Pricing.

Tópicos

- Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com o Database Insights
- Ativação do modo Avançado do Database Insights para Amazon Aurora
- Ativação do modo Standard do Database Insights para Amazon Aurora
- Configurar o banco de dados para monitorar consultas SQL lentas com o Database Insights para Amazon Aurora
- Considerações sobre o Database Insights para Amazon Aurora

Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com o Database Insights

A tabela a seguir fornece mecanismos de banco de dados do Amazon Aurora que são compatíveis com o Database Insights.

Mecanismo de bancos de dados Amazon Aurora	Versões do mecanismo e regiões compatíveis	Limitações de classes de instâncias
Amazon Aurora Edição Compatível com MySQL	Para obter mais informaçõ es sobre a disponibilidade de versões e regiões do Database Insights com o Aurora MySQL, consulte Performance Insights com o Aurora MySQL.	O Database Insights tem as seguintes restrições de classe de mecanismo: • db.t2 – Não compatível • db.t3 – Não compatível • db.t4g.micro e db.t4g.small: incompatíveis
Amazon Aurora Edição Compatível com PostgreSQL	Para obter mais informaçõ es sobre a disponibilidade de versões e regiões do Database Insights com o Aurora PostgreSQ L, consulte Performance Insights com o Aurora PostgreSQL.	Não aplicável
Aurora PostgreSQL Limitless Database	Para ter mais informações sobre como usar o Database Insights com o Aurora PostgreSQL Limitless Database, consulte Monitorar o Aurora PostgreSQ L Limitless Database com o CloudWatch Database Insights.	Não aplicável

O Database Insights oferece suporte ao Amazon Aurora Serverless v2.

Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com os recursos do Database Insights

A tabela a seguir apresenta os mecanismos de banco de dados do Amazon Aurora que são compatíveis com recursos do Database Insights.

Recurso	Nível de preço	Regiões compatíveis	Mecanismos de banco de dados compatíveis	Classes de instância compatíveis
Estatísticas SQL para Performance Insights	Todos	Todos	Todos	Todos
Analisar a performance do banco de dados por um período	Somente nível pago	Todos	Todos	Todos, exceto db.serverless (Aurora Serverless v2)
Visualizar as recomendações proativas do Performance Insights	Somente nível pago	• Leste dos EUA (Ohio) • Leste dos EUA (Norte da Virgínia) • Oeste dos EUA (N. da Califórnia) • Oeste dos EUA (Oregon) • Ásia-Pacífico (Mumbai) • Ásia-Pacífico (Seul) • Ásia-Pacífico (Singapura) • Ásia-Pacífico (Sydney) • Ásia-Pacífico (Tóquio)	Todos	Todos, exceto db.serverless (Aurora Serverless v2)

Recurso	Nível de preço	Regiões compatíveis	Mecanismos de banco de dados compatíveis	Classes de instância compatíveis
		• Canadá (Central) • Europa (Frankfurt) • Europa (Irlanda) • Europa (Londres) • Europa (Paris) • Europa (Estocolmo) • América do Sul (São Paulo)		

Regiões em que é possível usar o Database Insights no Amazon Aurora

O Aurora oferece suporte ao Database Insights nas Regiões da AWS a seguir.

- Leste dos EUA (Norte da Virgínia)
- Leste dos EUA (Ohio)
- Oeste dos EUA (N. da Califórnia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Ásia-Pacífico (Hyderabad)
- Ásia-Pacífico (Jacarta)
- Ásia-Pacífico (Malásia)
- Ásia-Pacífico (Melbourne)

- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Oeste do Canadá (Calgary)
- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milão)
- Europe (Paris)
- Europa (Espanha)
- Europa (Estocolmo)
- Europa (Zurique)
- Israel (Tel Aviv)
- Oriente Médio (Bahrein)
- Oriente Médio (Emirados Árabes Unidos)
- América do Sul (São Paulo)
- AWS GovCloud (Leste dos EUA)
- AWS GovCloud (Oeste dos EUA)

Ativação do modo Avançado do Database Insights para Amazon Aurora

Para ativar o modo Avançado do Database Insights para Amazon Aurora, use os procedimentos a seguir.

Ativar ou desativar o modo Avançado do Database Insights ao criar uma cluster de banco de dados

Ative o modo Avançado do Database Insights ao criar um banco de dados para Amazon Aurora.

Console

No console, você pode ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo Avançado do Database Insights ao criar usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Selecione Criar banco de dados.
4. Na seção Database Insights, selecione Modo avançado. Em seguida, escolha as seguintes opções:
   - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de 15 meses para o modo avançado do Database Insights.
   - AWS KMS key – especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
5. Selecione Criar banco de dados.

AWS CLI

Para ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados, chame o comando create-db-cluster da AWS CLI e forneça os seguintes valores:

- `--database-insights-mode advanced` para ativar o modo Avançado do Database Insights.
- `--engine`: o mecanismo de banco de dados para o cluster de banco de dados.
- `--db-cluster-identifier`, o identificar do cluster de banco de dados.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.

- `--performance-insights-retention-period`: o período de retenção de dados do cluster de banco de dados. Para ativar o Database Insights, o período de retenção deve ser de pelo menos 465 dias.

O exemplo a seguir habilita o modo Avançado do Database Insights ao criar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --database-insights-mode advanced \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode advanced ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API CreateDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `advanced`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos 465 dias

Ativar ou desativar o modo Avançado do Database Insights ao modificar uma cluster de banco de dados

Ative o Database Insights ao modificar um banco de dados para o Amazon Aurora. A modificação de um cluster de banco de dados para habilitar o modo avançado do Database Insights não causa tempo de inatividade.

> **Note**
>
> Para habilitar o Database Insights, cada instância de banco de dados em um cluster de banco de dados deve ter as mesmas configurações do Insights de Performance e do monitoramento avançado.

Console

No console, você pode ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo Avançado do Database Insights ao modificar usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha uma instância de banco de dados e escolha Modify (Modificar).
4. Na seção Database Insights, selecione Modo avançado. Em seguida, escolha as seguintes opções:
   - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de 15 meses para o modo avançado do Database Insights.
   - AWS KMS key – especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
5. Escolha Continue.

6. Em Programação das modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção programada, seu banco de dados vai ignorar essa configuração e ativar imediatamente o modo Avançado do Database Insights.
7. Escolha Modificar cluster.

AWS CLI

Para ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados, chame o comando modify-db-cluster da AWS CLI e forneça os seguintes valores:

- `--database-insights-mode advanced` para ativar o modo Avançado do Database Insights.
- `--db-cluster-identifier`, o identificar do cluster de banco de dados.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.
- `--performance-insights-retention-period`: o período de retenção de dados do cluster de banco de dados. Para ativar o modo Avançado do Database Insights, o período de retenção deve ser de pelo menos 465 dias.

O exemplo a seguir habilita o modo Avançado do Database Insights ao modificar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API ModifyDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `advanced`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos 465 dias

Ativação do modo Standard do Database Insights para Amazon Aurora

Para ativar o modo Standard do Database Insights para Amazon Aurora, use os procedimentos a seguir.

Ativar ou desativar o modo Standard do Database Insights ao criar uma cluster de banco de dados

Ative o modo Standard do Database Insights ao criar um banco de dados para Amazon Aurora.

Console

No console, você pode ativar o modo Standard do Database Insights ao criar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo Standard do Database Insights ao criar usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Selecione Criar banco de dados.
4. Na seção Database Insights, selecione Modo Standard. Em seguida, escolha uma das opções a seguir para ativar ou desativar o Insights de Performance:
   - Para desativar o Insights de Performance, desmarque a opção Habilitar Insights de Performance.
   - Para ativar o Insights de Performance, selecione Habilitar Insights de Performance. Para configurar o Performance Insights, especifique as seguintes opções:

- Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de pelo menos sete dias.
- AWS KMS key – especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.

5. Selecione Criar banco de dados.

AWS CLI

Para ativar o modo Standard do Database Insights ao criar um cluster de banco de dados, chame o comando create-db-cluster da AWS CLI e forneça os seguintes valores:

- `--database-insights-mode standard` para ativar o modo Standard do Database Insights.
- `--engine`: o mecanismo de banco de dados para o cluster de banco de dados.
- `--db-cluster-identifier`, o identificar do cluster de banco de dados.
- `--enable-performance-insights` ou `--no-enable-performance-insights` para ativar ou desativar o Performance Insights. Se você especificar `--enable-performance-insights`, também deverá especificar o `--performance-insights-retention-period`, que é o período de retenção de dados do cluster de banco de dados. O período de retenção deve ser de pelo menos sete dias.

O exemplo a seguir habilita o modo Padrão do Database Insights e o Insights de Performance ao criar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

O exemplo a seguir habilita o modo Padrão do Database Insights e desabilita o Insights de Performance ao criar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Para ativar o modo Standard do Database Insights ao criar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API CreateDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `standard`
- `EnablePerformanceInsights` como `True` ou `False`. Se você definir `EnablePerformanceInsights` como `True`, deverá definir `PerformanceInsightsRetentionPeriod` como pelo menos sete dias.

Ativar ou desativar o modo Standard do Database Insights ao modificar uma cluster de banco de dados

Ative o modo Standard do Database Insights ao modificar um banco de dados para o Amazon Aurora. A modificação de um cluster de banco de dados para habilitar o modo padrão do Database Insights não causa tempo de inatividade.

> **Note**
>
> Para habilitar o Database Insights, cada instância de banco de dados em um cluster de banco de dados deve ter as mesmas configurações do Insights de Performance e do monitoramento avançado.

Console

No console, você pode ativar o modo Standard do Database Insights ao modificar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo Standard do Database Insights ao modificar usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha uma instância de banco de dados e escolha Modify (Modificar).
4. Na seção Database Insights, selecione Modo Standard. Depois, escolha uma das seguintes opções:
   - Para desativar o Insights de Performance, desmarque a opção Habilitar Insights de Performance.
   - Para ativar o Insights de Performance, selecione Habilitar Insights de Performance. Para configurar o Performance Insights, especifique as seguintes opções:
     - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de pelo menos sete dias.
     - AWS KMS key – especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são

criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.

5. Escolha Continue.
6. Em Programação das modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção programada, seu banco de dados vai ignorar essa configuração e ativar imediatamente o modo Padrão do Database Insights.
7. Escolha Modificar cluster.

AWS CLI

Para ativar o modo Standard do Database Insights ao modificar um cluster de banco de dados, chame o comando modify-db-cluster da AWS CLI e forneça os seguintes valores:

- `--database-insights-mode standard` para ativar o modo Standard do Database Insights.
- `--db-cluster-identifier`, o identificar do cluster de banco de dados.
- `--enable-performance-insights` ou `--no-enable-performance-insights` para ativar ou desativar o Performance Insights. Se você especificar `--enable-performance-insights`, também deverá especificar o `--performance-insights-retention-period`, que é o período de retenção de dados do cluster de banco de dados. O período de retenção deve ser de pelo menos sete dias.

O exemplo a seguir habilita o modo Padrão do Database Insights e o Insights de Performance ao modificar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Para Windows:

```
aws rds modify-db-cluster ^
```

```
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

O exemplo a seguir habilita o modo Padrão do Database Insights e desabilita o Insights de Performance ao modificar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Para ativar o modo Standard do Database Insights ao modificar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API ModifyDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `standard`
- `EnablePerformanceInsights` como `True` ou `False`. Se você definir `EnablePerformanceInsights` como `True`, deverá definir `PerformanceInsightsRetentionPeriod` como pelo menos sete dias.

Configurar o banco de dados para monitorar consultas SQL lentas com o Database Insights para Amazon Aurora

Para monitorar consultas SQL lentas em seu banco de dados, você pode usar a seção Consultas SQL lentas no painel do Database Insights. Antes de você configurar seu banco de dados para monitorar consultas SQL lentas, a seção Consultas SQL lentas se apresenta em branco.

Para ter mais informações sobre o monitoramento de consultas SQL lentas no painel do Database Insights, consulte Visualizar o painel de instância de banco de dados do CloudWatch Database Insights no Guia do usuário do Amazon CloudWatch.

Para configurar o banco de dados para monitorar consultas SQL lentas com o Database Insights, conclua as seguintes etapas:

1. Habilite as exportações de log para o CloudWatch Logs.
2. Crie ou modifique o grupo de parâmetros do cluster de banco de dados para seu cluster de banco de dados.

Para ter informações sobre como configurar exportações de log, consulte Publicação de logs de banco de dados no Amazon CloudWatch Logs no Guia do usuário do Amazon Aurora.

Para criar ou modificar seu grupo de parâmetros de cluster de banco de dados, consulte os tópicos a seguir.

- Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora
- Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora

Amazon Aurora MySQL

Para configurar seu cluster de banco de dados do Amazon Aurora MySQL para monitorar consultas SQL lentas, defina os seguintes parâmetros.

- `slow_query_log`: definir como `1`
- `long_query_time`: definir como `1.0`
- `log_output`: definir como `FILE`

Amazon Aurora PostgreSQL

Para configurar seu cluster de banco de dados do Amazon Aurora PostgreSQL para monitorar consultas SQL lentas, defina os seguintes parâmetros. Observe que definir esses parâmetros pode reduzir o desempenho do cluster de banco de dados.

- `log_min_duration_statement`: definir como `1000`
- `log_statement`: definir como `none`

- `log_destination`: definir como `stderr`

> **Note**
>
> Para o Aurora MySQL, você pode configurar o parâmetro `long_query_time` com granularidade de 1 microssegundo. Por exemplo, você pode definir esse parâmetro como `0.000001`. Dependendo da quantidade de consultas na instância de banco de dados, o valor do parâmetro `long_query_time` pode reduzir o desempenho. Comece com o valor `1.0` e ajuste-o com base na sua workload. Quando você define esse parâmetro como `0`, o Database Insights registra em log todas as consultas.

Para ter informações sobre os logs do Aurora MySQL e do Aurora PostgreSQL, consulte o seguinte:

- Arquivos de log do banco de dados AuroraMySQL
- Arquivos de log do banco de dados Aurora PostgreSQL

Considerações sobre o Database Insights para Amazon Aurora

Confira a seguir considerações sobre o Database Insights para Amazon Aurora.

- Não é possível gerenciar o Database Insights para uma instância de banco de dados em um cluster de banco de dados.
- Para ativar o modo Avançado do Database Insights, você deve ativar o Insights de Performance e definir o período de retenção do Insights de Performance para pelo menos 465 dias (15 meses). Não há custo adicional para definir o período de retenção do Insights de Performance como 15 meses, além do custo do Database Insights. Para ter informações sobre os preços do Database Insights, consulte Definição de preço do Amazon CloudWatch.
- Para habilitar o Database Insights, cada instância de banco de dados em um cluster de banco de dados deve ter as mesmas configurações do Insights de Performance e do monitoramento avançado.
- A modificação de um cluster de banco de dados para habilitar qualquer um dos modos do Database Insights não causa tempo de inatividade.

Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.

Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.

Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

O Performance Insights expande os recursos de monitoramento do Amazon Aurora existentes para ilustrar e ajudar você a analisar a performance do cluster. Com o painel do Performance Insights, você pode visualizar a carga do banco de dados em sua carga de clusters do Amazon Aurora e filtrá-la por esperas, instruções SQL, hosts ou usuários. Para obter informações sobre como usar o Performance Insights com Amazon DocumentDB, consulte o Guia do desenvolvedor do Amazon DocumentDB.

Tópicos

- Visão geral do Performance Insights no Amazon Aurora
- Ativar e desativar o Insights de Performance para o Aurora
- Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL
- Configurar políticas de acesso para o Performance Insights
- Análise de métricas usando o painel do Performance Insights
- Visualizar as recomendações proativas do Performance Insights
- Recuperar métricas com a API do Insights de Performance para o Aurora
- Registrar em log as chamadas do Performance Insights usando o AWS CloudTrail
- API do Performance Insights e endpoints da VPC de interface (AWS PrivateLink)

Visão geral do Performance Insights no Amazon Aurora

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.

Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.

Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande

escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

Por padrão, o RDS ativa o Insights de Performance no assistente de criação do console para mecanismos do Amazon RDS. Se você ativar o Insights de Performance em nível de cluster de banco de dados, o RDS habilitará o Insights de Performance para cada instância de banco de dados no cluster. Se você tiver mais de um banco de dados em uma instância de banco de dados, o Performance Insights agregará dados de performance.

É possível encontrar uma visão geral do Performance Insights para Amazon Aurora no vídeo a seguir.

Como usar o Performance Insights para analisar a performance do Amazon Aurora PostgreSQL

Tópicos

- Carga de banco de dados
- Máximo de CPU
- Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com o Insights de Performance
- Preços e retenção de dados para o Performance Insights

Carga de banco de dados

Carga do banco do dados mede o nível de atividade de sessão no banco de dados. `DBLoad` é a métrica principal no Insights de Performance, e o Insights de Performance coleta a carga do banco de dados a cada segundo.

Tópicos

- Sessões ativas
- Média de sessões ativas
- Média de execuções ativas
- Dimensões

Sessões ativas

Uma sessão de base de dados relacional representa o diálogo de uma aplicação com um banco de dados relacional. Uma sessão ativa é uma conexão que enviou trabalho para o mecanismo de banco de dados e está aguardando uma resposta.

Uma sessão fica ativa quando está em execução na CPU ou aguardando a disponibilidade de um recurso para que ela possa continuar. Por exemplo, uma sessão ativa pode esperar que uma página (ou um bloco) seja lida na memória e, depois, consumir CPU enquanto faz a leitura dos dados na página.

Média de sessões ativas

A média de sessões ativas (AAS) é a unidade da métrica `DBLoad` no Performance Insights. Ele mede quantas sessões estão ativas simultaneamente no banco de dados.

A cada segundo, o Insights de Performance faz uma amostra do número de sessões executando simultaneamente uma consulta. Para cada sessão ativa, o Insights de Performance coleta os seguintes dados:

- Declaração do SQL
- Estado da sessão (em execução na CPU ou em espera)
- Host
- Usuário executando o SQL

O Insights de Performance calcula a AAS dividindo o número total de sessões pelo número total de amostras por um período específico. Por exemplo, a tabela a seguir mostra cinco amostras consecutivas de uma consulta em execução em intervalos de um segundo.

Amostra	Número de sessões que executam a consulta	AAS	Cálculo
1	2	2	2 sessões no total/1 amostra
2	0	1	2 sessões no total/2 amostras

Amostra	Número de sessões que executam a consulta	AAS	Cálculo
3	4	2	6 sessões no total/3 amostras
4	0	1.5	6 sessões no total/4 amostras
5	4	2	10 sessões no total/5 amostras

No exemplo anterior, a carga do banco de dados para o intervalo de tempo foi de 2 AAS. Essa medida significa que, em média, duas sessões estavam ativas em determinado momento durante o intervalo em que as cinco amostras foram obtidas.

Média de execuções ativas

A média de execuções ativas (AAE) por segundo está relacionada ao AAS. Para calcular os AAE, o Performance Insights divide o tempo total de execução de uma consulta pelo intervalo de tempo. A tabela a seguir mostra o cálculo de AAE para a mesma consulta na tabela anterior.

Tempo decorrido (s)	Tempo de execução total (s)	AAE	Cálculo
60	120	2	120 segundos de execução/60 segundos decorridos
120	120	1	120 segundos de execução/120 segundos decorridos
180	380	2.11	380 segundos de execução/180 segundos decorridos

Tempo decorrido (s)	Tempo de execução total (s)	AAE	Cálculo
240	380	1,58	380 segundos de execução/240 segundos decorridos
300	600	2	600 segundos de execução/300 segundos decorridos

Na maioria dos casos, o AAS e o AAE de uma consulta são aproximadamente os mesmos. No entanto, como as entradas para os cálculos são diferentes fontes de dados, os cálculos geralmente variam ligeiramente.

Dimensões

A métrica `db.load` é diferente das outras métricas da série temporal, pois você pode fragmentá-la em subcomponentes chamados de dimensões. Você pode pensar em dimensões como “pedaços” de categorias para as diferentes características da métrica `DBLoad`.

Quando você está diagnosticando problemas de performance, as seguintes dimensões geralmente são as mais úteis:

Tópicos

- Eventos de espera
- SQL principal

Para obter uma lista completa de dimensões dos mecanismos Aurora, consulte Carga de banco de dados separada por dimensões.

Eventos de espera

Um evento de espera faz com que uma instrução SQL aguarde que um evento específico aconteça antes que ele possa continuar a execução. Eventos de espera são uma dimensão, ou categoria, importante para a carga do banco de dados, pois indicam onde o trabalho está impedido.

Todas as sessões ativas estão em um estado de espera ou de execução na CPU. Por exemplo, sessões consomem CPU quando procuram um buffer na memória, realizam um cálculo ou executam um código processual. Quando as sessões não estão consumindo CPU, elas podem estar aguardando a liberação de um buffer de memória, a leitura de um arquivo de dados ou a gravação em um log. Quanto mais tempo uma sessão aguardar recursos, menos tempo ela será executada na CPU.

Ao ajustar um banco de dados, muitas vezes você tenta descobrir os recursos que as sessões estão aguardando. Por exemplo, dois ou três eventos de espera podem representar 90% da carga do banco de dados. Essa medida significa que, em média, as sessões ativas estão passando a maior parte do tempo aguardando um pequeno número de recursos. Se você conseguir descobrir a causa dessas esperas, poderá tentar uma solução.

Os eventos de espera variam de acordo com o mecanismo de banco de dados:

- Para obter uma lista dos eventos de espera comuns do Aurora MySQL, consulte Eventos de espera do Aurora MySQL. Para aprender a ajustar utilizando esses eventos de espera, consulte Ajustar o Aurora MySQL.
- Para obter informações sobre todos os eventos de espera do MySQL, consulte Tabelas de resumo de eventos de espera na documentação do MySQL.
- Para obter uma lista de eventos de espera comuns do Aurora PostgreSQL, consulte Eventos de espera do Amazon Aurora PostgreSQL. Para aprender a ajustar utilizando esses eventos de espera, consulte Ajustar com eventos de espera do Aurora PostgreSQL.
- Para obter informações sobre todos os eventos de espera do PostgreSQL, consulte The Statistics Collector > Wait Event tables (Coletor de estatísticas > Tabelas de eventos de espera) na documentação do PostgreSQL.

SQL principal

Enquanto eventos de espera mostram gargalos, o gráfico Top SQL (SQL principal) mostra quais consultas estão contribuindo mais para a carga do banco de dados. Por exemplo, muitas consultas podem estar em execução no banco de dados, mas uma única consulta pode consumir 99% da carga do banco de dados. Nesse caso, a carga alta pode indicar um problema com a consulta.

Por padrão, o console do Performance Insights exibe as consultas de SQL principal que estão contribuindo para a carga do banco de dados. O console também mostra estatísticas relevantes para cada instrução. Para diagnosticar problemas de performance para uma instrução específica, você pode examinar seu plano de execução.

Máximo de CPU

No painel, o gráfico Carga de banco de dados coleta, agrega e exibe informações da sessão. Para ver se as sessões ativas estão excedendo o máximo de CPU, observe sua relação com a linha Máx. vCPU. O Insights Performance determina o valor de Máx. vCPU pelo número de núcleos de vCPU (CPUs virtuais) da instância de banco de dados. Para Aurora Serverless v2, Máx. vCPU representa o número estimado de vCPUs.

Um processo pode ser executado em uma vCPU por vez. Se o número de processos exceder o número de vCPUs, os processos começarão a ser colocados em fila. Quando a colocação em fila aumenta, a performance do banco de dados diminui. Se a carga de banco de dados estiver com frequência acima da linha Máx. vCPU e o estado de espera primário for CPU, isso indicará que a CPU está sobrecarregada. Nesse caso, convém limitar as conexões com a instância, ajustar todas as consultas SQL com uma alta carga de CPU ou considerar uma classe de instância maior. As instâncias altas e consistentes de qualquer estado de espera indicam que pode haver problemas de gargalos ou de contenção de recursos que você deve resolver. Isso pode ser válido mesmo quando a carga do banco de dados não ultrapassa a linha de Máx. vCPU.

Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com o Insights de Performance

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de

análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

A tabela a seguir fornece mecanismos de banco de dados do Amazon Aurora que são compatíveis com o Insights de Performance.

Mecanismo de bancos de dados Amazon Aurora	Versões do mecanismo e regiões compatíveis	Limitações de classes de instâncias
Amazon Aurora Edição Compatível com MySQL	Para obter mais informaçõ es sobre a disponibilidade de versões e regiões do Performan ce Insights com o Aurora MySQL, consulte Performance Insights com o Aurora MySQL.	O Performance Insights tem as seguintes restrições de classe de mecanismo: • db.t2 – Não compatível • db.t3 – Não compatível • db.t4g.micro e db.t4g.small: incompatíveis
Amazon Aurora Edição Compatível com PostgreSQL	Para obter mais informaçõ es sobre a disponibilidade de versões e regiões do Performance Insights com o Aurora PostgreSQ L, consulte Performance Insights com o Aurora PostgreSQL.	N/D

Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance

A tabela a seguir fornece mecanismos de banco de dados do Amazon Aurora que são compatíveis com atributos do Insights de Performance.

Recurso	Nível de preço	Regiões compatíveis	Mecanismos de banco de dados compatíveis	Classes de instância compatíveis
Estatísticas SQL para Performance Insights	Todos	Todos	Todos	Todos
Analisar a performance do banco de dados por um período	Somente nível pago	Todos	Todos	Todos, exceto db.serverless (Aurora Serverless v2)
Visualizar as recomendações proativas do Performance Insights	Somente nível pago	• Leste dos EUA (Ohio) • Leste dos EUA (Norte da Virgínia) • Oeste dos EUA (N. da Califórnia) • Oeste dos EUA (Oregon) • Ásia-Pacífico (Mumbai) • Ásia-Pacífico (Seul) • Ásia-Pacífico (Singapura)	Todos	Todos, exceto db.serverless (Aurora Serverless v2)

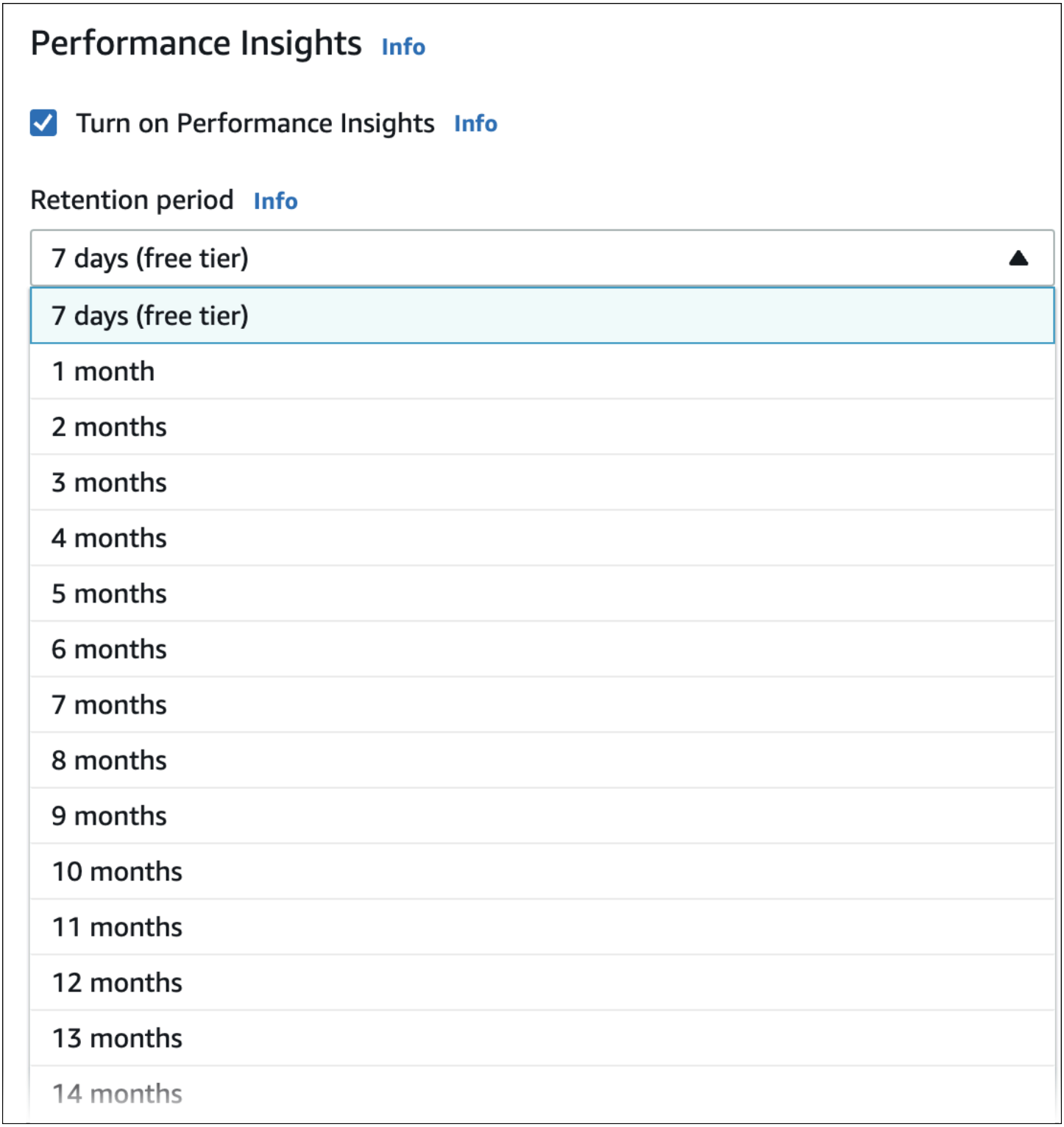
Recurso	Nível de preço	Regiões compatíveis	Mecanismos de banco de dados compatíveis	Classes de instância compatíveis
		• Ásia-Pacífico (Sydney) • Ásia-Pacífico (Tóquio) • Canadá (Central) • Europa (Frankfurt) • Europa (Irlanda) • Europa (Londres) • Europa (Paris) • Europa (Estocolmo) • América do Sul (São Paulo)		

Preços e retenção de dados para o Performance Insights

Por padrão, o Performance Insights oferece um nível gratuito que inclui 7 dias de histórico de dados de performance e 1 milhão de solicitações de API por mês. Você também pode comprar períodos de retenção mais longos. Para obter informações completas sobre custos, consulte Definição de preço do Performance Insights.

No console do RDS, você pode escolher qualquer um dos seguintes períodos de retenção para seus dados do Performance Insights:

- Default (7 days) (Padrão (7 dias))
- ***n*** meses, em que ***n*** é um número entre 1 e 24



Para saber como definir um período de retenção usando a AWS CLI, consulte ???.

Ativar e desativar o Insights de Performance para o Aurora

Important

A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.

Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.

Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

É possível ativar o Insights de Performance para o cluster de banco de dados durante a criação. Se necessário, você pode desativá-lo posteriormente modificando o cluster de banco de dados no console. A ativação e a desativação do Performance Insights não causa tempo de inatividade, reinicialização ou failover.

Note

O Performance Schema é uma ferramenta de performance opcional usada pelo Aurora MySQL. Se você ativar ou desativar o Performance Schema, será necessário reinicializar. No

entanto, se você ativar ou desativar o Performance Insights, não será necessário reinicializar. Para obter mais informações, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Se você usar o Insights de Performance com os bancos de dados globais do Aurora, ative o Insights de Performance individualmente para as instâncias de bancos de dado em cada Região da AWS. Para obter detalhes, consulte Monitorando um banco de dados Amazon Aurora global com Performance Insights Amazon RDS.

O agente do Performance Insights consome CPU e memória limitadas no host do banco de dados. Quando a carga do banco de dados é alta, o agente limita o impacto sobre a performance coletando dados com menos frequência.

Console

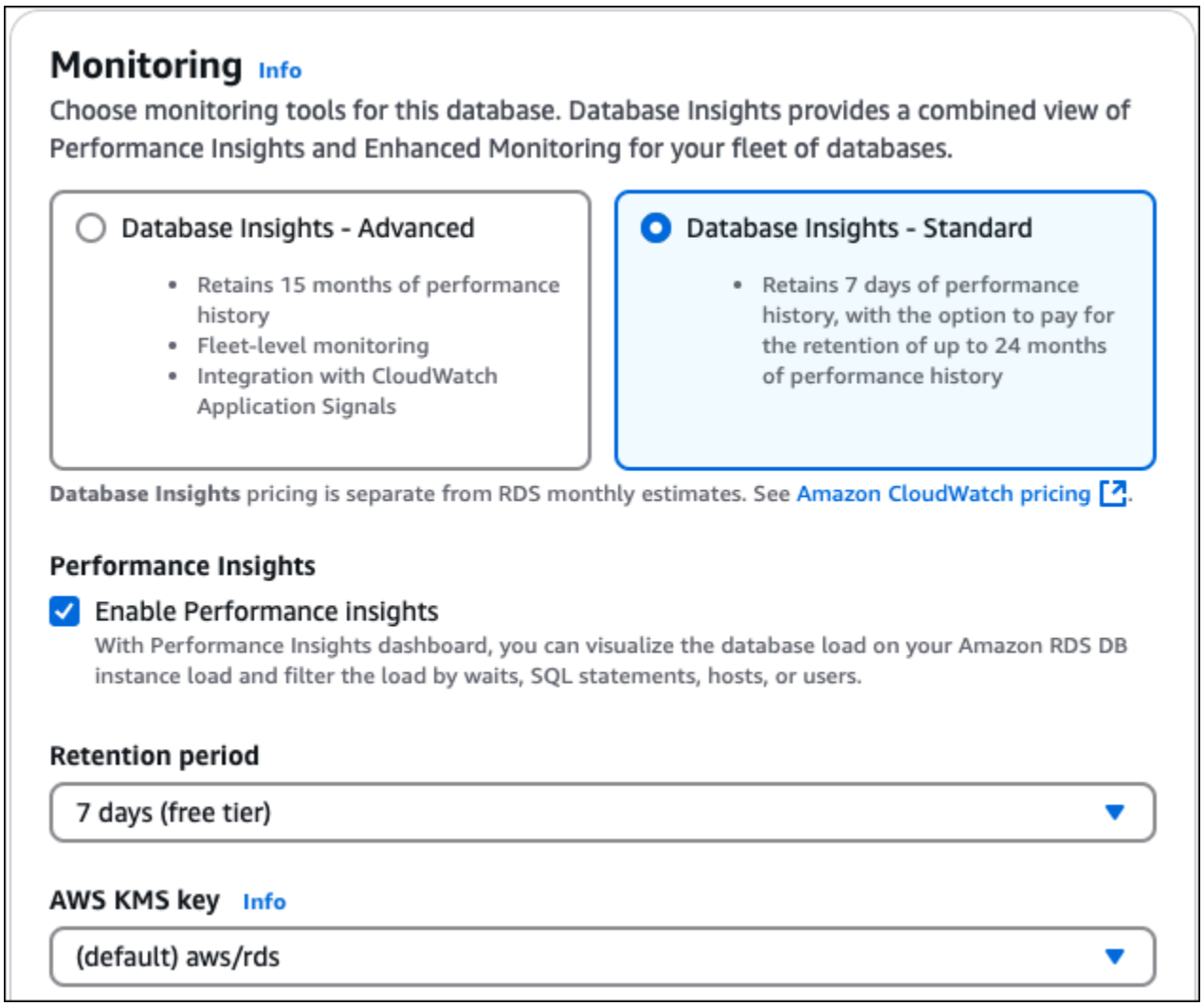
No console, ative ou desative o recurso Insights de Performance ao criar um cluster de banco de dados. Habilitar o Insights de Performance possibilita gerenciar as configurações e opções do Insights de Performance para o cluster de banco de dados. As configurações no nível do cluster se aplicam a todas as instâncias de banco de dados no cluster.

Ativar ou desativar o Insights de Performance ao criar uma cluster de banco de dados

Depois de criar um cluster de banco de dados, o Amazon RDS habilita o Insights de Performance por padrão. Para desativar o Insights de Performance para o cluster de banco de dados, escolha a opção Database Insights: padrão e desmarque a opção Habilitar Insights de Performance.

Para criar um cluster de banco de dados, siga as instruções relacionadas ao mecanismo de banco de dados em Criar um cluster de bancos de dados do Amazon Aurora.

A captura de tela a seguir mostra a seção Performance Insights.



Se você escolher Enable Performance Insights (Habilitar o Performance Insights) terá as seguintes opções:

- Retenção (apenas para o modo padrão do Database Insights): o intervalo de tempo para reter os dados do Insights de Performance. A configuração de retenção é Padrão (7 dias). Para reter seus dados de performance por mais tempo, especifique entre 1 e 24 meses. Para obter mais informações sobre os períodos de retenção, consulte Preços e retenção de dados para o Performance Insights.
- AWS KMS key – especifica a sua AWS KMS key. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Alterar uma política do AWS KMS para o Insights de Performance.

Ativar ou desativar o Insights de Performance ao modificar um cluster de banco de dados

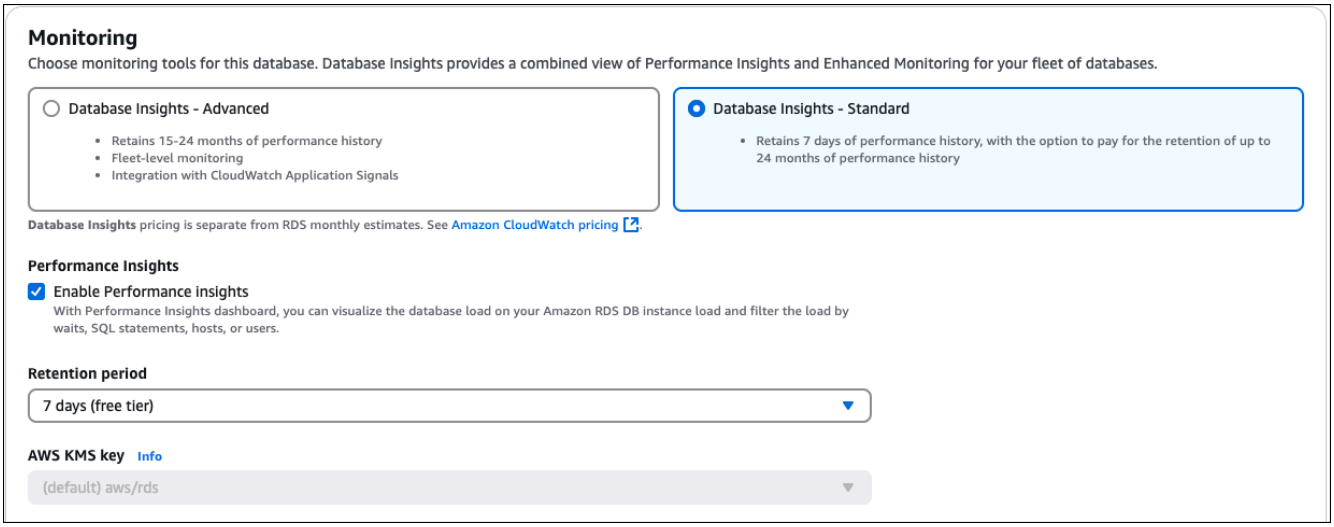
No console, é possível modificar um cluster de banco de dados para gerenciar o Insights de Performance.

Como gerenciar o Insights de Performance para um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha um cluster de banco de dados e selecione Modificar.
4. Para ativar o Insights de Performance, selecione Habilitar Insights de Performance. Para desativar o Insights de Performance para o cluster de banco de dados, escolha a opção Database Insights: padrão e desmarque a opção Habilitar Insights de Performance.

   Se você escolher Enable Performance Insights (Habilitar o Performance Insights) terá as seguintes opções:

   - Retenção (apenas para o modo padrão do Database Insights): o intervalo de tempo para reter os dados do Insights de Performance. A configuração de retenção é Padrão (7 dias). Para reter seus dados de performance por mais tempo, especifique entre 1 e 24 meses. Para obter mais informações sobre os períodos de retenção, consulte Preços e retenção de dados para o Performance Insights.
   - AWS KMS key – especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.



5. Escolha Continue.
6. Em Scheduling of Modifications (Programação de modificações), escolha Apply immediately (Aplicar imediatamente). Se você escolher Apply during the next scheduled maintenance window (Aplicar durante a próxima janela de manutenção agendada), sua instância ignorará essa configuração e ativará o Performance Insights imediatamente.
7. Escolha Modify instance (Modificar instância).

AWS CLI

Ao usar o comando create-db-instance da AWS CLI, ative o Insights de Performance especificando `--enable-performance-insights` e definindo `--database-insights-mode` como `advanced` ou `standard`. Para desativar o Insights de Performance, especifique `--no-enable-performance-insights` e `database-insights-mode` como `standard`.

Você também pode especificar esses valores usando os seguintes comandos da AWS CLI:

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

Como gerenciar o Insights de Performance para um cluster de banco de dados usando a AWS CLI

- Chame o comando modify-db-cluster AWS CLI e forneça os seguintes valores:
  - `--db-cluster-identifier`: o nome da instância de banco de dados no cluster de banco de dados.
  - `--enable-performance-insights` para ativar ou `--no-enable-performance-insights` para desativar
  - `database-insights-mode`: o modo do Database Insights para o cluster de banco de dados. Para desativar o Insights de Performance, defina esse valor como `standard`.

  O exemplo a seguir ativa o Performance Insights para a `sample-db-cluster`.

  Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights
```

  Para Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights
```

Quando você ativa o Performance Insights na CLI, é possível especificar o número de dias para retenção dos dados do Performance Insights com a opção `--performance-insights-retention-period`. Você pode especificar 7, *mês* * 31 (em que *mês* é um número de 1 a 23) ou 731. Por exemplo, se você quiser reter seus dados de desempenho por 3 meses, especifique 93, que é 3 * 31. O padrão são 7 dias. Para obter mais informações sobre os períodos de retenção, consulte Preços e retenção de dados para o Performance Insights.

O exemplo a seguir ativa o Performance Insights para `sample-db-cluster` e especifica que os dados do Performance Insights serão retidos por 93 dias (3 meses).

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Para Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

Se você especificar um período de retenção como 94 dias, que não é um valor válido, o RDS emitirá um erro.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> É possível ativar o Insights de Performance para uma instância em um cluster de banco de dados no qual o Insights de Performance não é gerenciado no nível do cluster.

RDS API

Ao criar uma instância de banco de dados em seu cluster de banco de dados usando a operação CreateDBInstance da API do Amazon RDS, ative o Performance Insights definindo `EnablePerformanceInsights` como `True`. Para desativar o Insights de Performance, defina `EnablePerformanceInsights` como `False` e `DatabaseInsightsMode` como `standard`.

Também é possível especificar o valor `EnablePerformanceInsights` usando as seguintes operações da API:

- CreateDBCluster
- ModifyDBCluster
- ModifyDBInstance
- CreateDBInstanceReadReplica
- RestoreDBInstanceFromS3

Quando você ativar o Performance Insights, é possível especificar a quantidade de tempo, em dias, para a retenção de dados do Performance Insights com o parâmetro `PerformanceInsightsRetentionPeriod`. Você pode especificar 7, *`mês`* * 31 (em que *`mês`* é um número de 1 a 23) ou `731`. Por exemplo, se você quiser reter seus dados de desempenho por 3 meses, especifique `93`, que é 3 * 31. O padrão são 7 dias. Para obter mais informações sobre os períodos de retenção, consulte Preços e retenção de dados para o Performance Insights.

Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL

O Performance Schema é um recurso opcional para monitorar a performance do tempo de execução do Aurora MySQL em um nível inferior de detalhes. O Performance Schema foi projetado para causar impacto mínimo na performance do banco de dados. O Performance Insights é um recurso separado que você pode usar com ou sem o Performance Schema.

Tópicos

- Visão Geral do Performance Schema
- Performance Insights e Performance Schema
- Gerenciamento automático do Performance Schema pelo Performance Insights
- Efeito de uma reinicialização no Performance Schema
- Determinar se o Performance Insights está gerenciando o Performance Schema
- Ativar o Performance Schema para o Aurora MySQL

Visão Geral do Performance Schema

O Performance Schema monitora os eventos nos bancos de dados Aurora MySQL. Um evento é uma ação do servidor de banco de dados que consome tempo e foi instrumentada para que as informações de tempo possam ser coletadas. São exemplos de evento:

- Chamadas de função
- Aguarda o sistema operacional
- Estágios da execução SQL
- Grupos de instruções SQL

O mecanismo de armazenamento `PERFORMANCE_SCHEMA` é um mecanismo para implementar o recurso Performance Schema. Esse mecanismo coleta dados de eventos usando instrumentação no código-fonte do banco de dados. O mecanismo armazena eventos em tabelas somente na memória no banco de dados `performance_schema`. É possível consultar `performance_schema` assim como quaisquer outras tabelas. Para obter mais informações, consulte MySQL Performance Schema no Guia de referência do MySQL.

Performance Insights e Performance Schema

O Performance Insights e o Performance Schema são recursos separados, mas estão conectados. O comportamento do Performance Insights para Aurora MySQL depende se o Performance Schema está ativado e, em caso afirmativo, se o Performance Insights gerencia o Performance Schema automaticamente. A tabela a seguir descreve o comportamento.

Performance Schema ativado	Modo de gerenciamento do Performance Insights	Comportamento do Performance Insights
Sim	Automatico	• Coleta informações de monitoramento detalhadas e de nível inferior. • Coleta métricas de sessão ativas a cada segundo. • Exibe carga de banco de dados categorizada por eventos de espera detalhados, que você pode usar para identificar gargalos.
Sim	Manual	• Coleta eventos de espera e métricas por SQL • Coleta métricas de sessão ativas a cada segundo.

Performance Schema ativado	Modo de gerenciamento do Performance Insights	Comportamento do Performance Insights
		• Relata estados de usuário, como inserção e envio, que não ajudam a identificar gargalos.
Não	N/D	• Não coleta eventos de espera, métricas por SQL nem outras informações detalhadas de monitoramento de nível inferior. • Coleta métricas de sessão ativas a cada cinco segundos em vez de cada segundo. • Relata estados de usuário, como inserção e envio, que não ajudam a identificar gargalos.

Gerenciamento automático do Performance Schema pelo Performance Insights

Quando você cria uma instância de banco de dados do Aurora MySQL com o Performance Insights ativado, o Performance Schema também é ativado. Nesse caso, o Performance Insights gerencia automaticamente seus parâmetros do Performance Schema. Essa é a configuração recomendada.

Quando o Insights de Performance gerencia o Performance Schema automaticamente, a Origem de `performance_schema` é `System default`.

> **Note**
>
> O gerenciamento automático do esquema de performance não é compatível com a classe de instância t4g.medium.

Você também pode gerenciar o Performance Schema manualmente. Se você escolher essa opção, defina os parâmetros de acordo com os valores na tabela a seguir.

Nome do parâmetro	Valor do parâmetro
`performance_schema`	1 (a coluna Source (Fonte) tem o valor `Modified`)
`performance-schema-consumer-events-waits-current`	`ON`
`performance-schema-instrument`	`wait/%=ON`
`performance_schema_consumer_global_instrumentation`	`1`
`performance_schema_consumer_thread_instrumentation`	`1`

Se você alterar o valor do parâmetro `performance_schema` manualmente e, posteriormente, quiser reverter para o gerenciamento automático, consulte Ativar o Performance Schema para o Aurora MySQL.

> **Important**
>
> Quando o Performance Insights ativa o Performance Schema, ele não altera os valores do grupo de parâmetros. No entanto, os valores são alterados nas instâncias de banco de dados que estão em execução. A única forma de ver os valores alterados é executar o comando `SHOW GLOBAL VARIABLES`.

Efeito de uma reinicialização no Performance Schema

O Performance Insights e o Performance Schema diferem em seus requisitos para reinicializações de instâncias de banco de dados:

Performance Schema

Para ativar ou desativar esse recurso, você deve reinicializar a instância de banco de dados.

Performance Insights

Para ativar ou desativar esse recurso, não é necessário reinicializar a instância de banco de dados.

Se o Performance Schema não estiver ativado no momento e você ativar o Performance Insights sem reinicializar a instância de banco de dados, o Performance Schema não será ativado.

Determinar se o Performance Insights está gerenciando o Performance Schema

Para descobrir se, no momento, o Insights de Performance está gerenciando o Performance Schema nas versões principais compatíveis do mecanismo, consulte a tabela a seguir.

Configuração do parâmetro performance_schema	Configuração da coluna Source (Fonte)	O Performance Insights está gerenciando o Performance Schema?
`0`	`System default`	Sim
`0` ou `1`	`Modified`	Não

No procedimento a seguir, você determinará se o Performance Insights está gerenciando o Performance Schema automaticamente.

Como saber se o Performance Insights está gerenciando automaticamente o Performance Schema

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Parameter groups (Grupos de parâmetros).
3. Selecione o nome do grupo de parâmetros para a instância de banco de dados.
4. Insira **`performance_schema`** na barra de pesquisa.
5. Confira se o valor de Origem é o padrão do sistema e Valor é 0. Nesse caso, o Performance Insights está gerenciando o Performance Schema automaticamente.

   No exemplo mostrado aqui, o Performance Insights não está gerenciando o Performance Schema automaticamente.

Ativar o Performance Schema para o Aurora MySQL

Suponha que o Performance Insights esteja ativado para sua instância de banco de dados, mas no momento, não está gerenciando o Performance Schema. Se você quiser permitir que o Performance Insights gerencie o Performance Schema automaticamente, conclua as etapas a seguir.

Como configurar o Performance Schema para gerenciamento automático

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Parameter groups (Grupos de parâmetros).
3. Selecione o nome do grupo de parâmetros de sua instância de banco de dados.
4. Escolha Editar.
5. Insira **`performance_schema`** na barra de pesquisa.
6. Selecione o parâmetro `performance_schema`.
7. Selecione Definir como valor padrão.
8. Confirme escolhendo Definir valores como padrão.
9. Escolha Salvar alterações.
10. Reinicialize a instância de banco de dados.

> **Important**
>
> Sempre que habilitar ou desabilitar o Performance Schema, você deverá reinicializar a instância de banco de dados.

Para obter mais informações sobre como modificar os parâmetros da instância, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora. Para obter mais informações sobre o painel, consulte Análise de métricas usando o painel do Performance Insights. Para obter mais informações sobre o esquema de desempenho do MySQL, consulte MySQL

Performance Schema (para 8.0) na MySQL Performance Schema (para 8.4) na documentação do MySQL.

Configurar políticas de acesso para o Performance Insights

Para acessar o Performance Insights, é necessário que uma entidade principal tenha as permissões apropriadas do AWS Identity and Access Management (IAM).

> **Note**
>
> Para usar o Insights de Performance com uma chave gerenciada pelo cliente, conceda aos usuários as permissões `kms:Decrypt` e `kms:GenerateDataKey` para sua chave do AWS AWS KMS.

Acesse o Insights de Performance usando estes métodos:

- Anexe a política gerenciada `AmazonRDSPerformanceInsightsReadOnly` para acesso somente leitura
- Anexar a política gerenciada `AmazonRDSPerformanceInsightsFullAccess` para acessar todas as operações da API do Insights de Performance
- Crie uma política personalizada do IAM com permissões específicas.
- Configurar permissões do AWS KMS para dados criptografados do Insights de Performance
- Configurar o acesso refinado usando permissões em nível de recurso
- Usar o controle de acesso baseado em tags para gerenciar permissões por meio de tags de recursos

Anexar a política AmazonRDSPerformanceInsightsReadOnly a uma entidade principal do IAM

A `AmazonRDSPerformanceInsightsReadOnly` é uma política gerenciada pela AWS que concede acesso a todas as operações somente leitura da API do Insights de Performance do Amazon RDS.

Se você anexar `AmazonRDSPerformanceInsightsReadOnly` a um conjunto de permissões ou perfil, também deverá anexar as seguintes permissões do CloudWatch:

- `GetMetricStatistics`

- `ListMetrics`
- `GetMetricData`

Com essas permissões, o destinatário pode usar o Insights de Performance junto com outros recursos do console.

Para ter mais informações sobre as permissões do CloudWatch, consulte Referência de permissões do Amazon CloudWatch.

Para obter mais informações sobre o `AmazonRDSPerformanceInsightsReadOnly`, consulte AWSPolítica gerenciada pela : AmazonRDSPerformanceInsightsReadOnly.

Anexar a política AmazonRDSPerformanceInsightsFullAccess a uma entidade principal do IAM

A `AmazonRDSPerformanceInsightsFullAccess` é uma política gerenciada pela AWS que concede acesso a todas as operações da API do Insights de Performance do Amazon RDS.

Se você anexar `AmazonRDSPerformanceInsightsFullAccess` a um conjunto de permissões ou perfil, também deverá anexar as seguintes permissões do CloudWatch:

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Com essas permissões, o destinatário pode usar o Insights de Performance junto com outros recursos do console.

Para ter mais informações sobre as permissões do CloudWatch, consulte Referência de permissões do Amazon CloudWatch.

Para obter mais informações sobre o `AmazonRDSPerformanceInsightsFullAccess`, consulte Política gerenciada pela AWS: AmazonRDSPerformanceInsightsFullAccess.

Criação de uma política de IAM personalizada para o Performance Insights

Para usuários que não têm a política `AmazonRDSPerformanceInsightsReadOnly` ou `AmazonRDSPerformanceInsightsFullAcce` é possível conceder acesso ao Insights de Performance criando ou modificando uma política do IAM

gerenciada pelo usuário. Quando você anexa a política a um conjunto de permissões ou perfil do IAM, o destinatário pode usar o Performance Insights.

Para criar uma política personalizada

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Policies.
3. Selecione Criar política.
4. Na página Criar política, selecione a guia JSON.
5. Copie e cole o texto fornecido na seção do documento de política JSON no Guia de referência de políticas gerenciadas da AWS para AmazonRDSPerformanceInsightsReadOnly ou a política AmazonRDSPerformanceInsightsFullAccess.
6. Escolha Review policy (Revisar política).
7. Forneça um nome para a política e, se preferir, uma descrição. Em seguida, escolha Criar política.

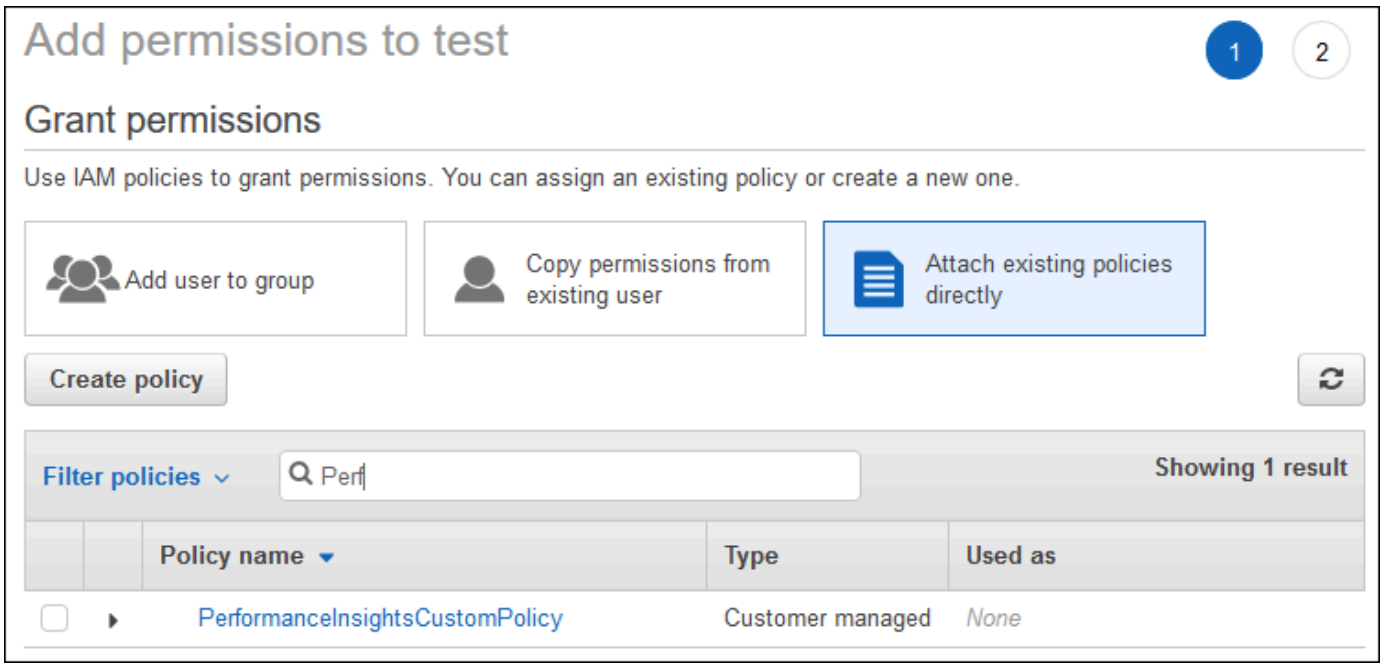
Agora você pode anexar a política a um conjunto de permissões ou perfil. O procedimento a seguir pressupõe que você já tem um usuário disponível para essa finalidade.

Como anexar a política a um usuário

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Usuários.
3. Escolha um usuário existente na lista.

> **Important**
>
> Para usar o Performance Insights, você deve ter acesso ao Amazon RDS e à política personalizada. Por exemplo, a política predefinida `AmazonRDSPerformanceInsightsReadOnly` concede acesso somente leitura ao Amazon RDS. Para obter mais informações, consulte Gerenciamento do acesso usando políticas.

4. Na página Summary (Resumo), escolha Add permissions (Adicionar permissões).
5. Escolha Anexar políticas existentes diretamente. Em Pesquisar, digite os primeiros caracteres do nome da política, conforme mostrado na imagem a seguir.



6. Escolha a política e, em seguida, escolha Próximo: revisar.
7. Escolha Adicionar permissões.

Alterar uma política do AWS KMS para o Insights de Performance

O Performance Insights usa uma AWS KMS key para criptografar dados sigilosos. Ao habilitar o Performance Insights por meio da API ou do console, você poderá seguir um destes procedimentos:

- Escolha o padrão Chave gerenciada pela AWS.

  O Amazon RDS usa a Chave gerenciada pela AWS para a sua nova instância de banco de dados. O Amazon RDS cria uma Chave gerenciada pela AWS para a sua Conta da AWS. A sua Conta da AWS tem uma Chave gerenciada pela AWS diferente para o Amazon RDS para cada Região da AWS.

- Escolha uma chave gerenciada pelo cliente.

  Se você especificar uma chave gerenciada pelo cliente, os usuários em sua conta que chamam a API do Performance Insights precisarão das permissões `kms:Decrypt` e `kms:GenerateDataKey` na chave do KMS. É possível configurar essas permissões por meio de políticas do IAM. No entanto, recomendamos que você gerencie essas permissões por meio da política de chaves do KMS. Para saber mais, consulte Políticas de chaves no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Example

O exemplo a seguir mostra como adicionar instruções à sua política da chaves do KMS. Essas instruções permitem acesso ao Performance Insights. Dependendo de como você usa a chave KMS, talvez você queira alterar algumas restrições. Antes de adicionar instruções à política, remova todos os comentários.

JSON

```
{
    "Version":"2012-10-17",
    "Id" : "your-policy",
    "Statement" : [
        {
            "Sid" : "AllowViewingRDSPerformanceInsights",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/Role1"
                ]
                },
             "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
                ],
            "Resource": "*",
            "Condition" : {
            "StringEquals" : {
                "kms:ViaService" : "rds.us-east-1.amazonaws.com"
                },
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContext:aws:pi:service": "rds",
                "kms:EncryptionContext:service": "pi",
                "kms:EncryptionContext:aws:rds:db-id": "db-
AAAAABBBBBCCCCDDDDDEEEEE"
                }
            }
        }
    ]
}
```

Como o Insights de Performance usa a chave do AWS KMS gerenciada pelo cliente

O Insights de Performance usa chaves gerenciadas pelo cliente para criptografar dados sigilosos. Ao ativar o Insights de Performance, você pode fornecer uma chave do AWS KMS por meio da API. O Insights de Performance cria permissões do AWS KMS nessa chave. Ele usa a chave e executa as operações necessárias para processar dados sigilosos. Os dados sigilosos incluem campos como usuário, banco de dados, aplicação e texto de consulta SQL. O Insights de Performance garante que os dados permaneçam criptografados tanto em repouso quanto em trânsito.

Como o Insights de Performance e o IAM funcionam com o AWS KMS

O IAM concede permissões para APIs específicas. O Insights de Performance tem as seguintes APIs públicas, que você pode restringir usando políticas do IAM:

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

Você pode usar as solicitações de API a seguir para obter dados sigilosos.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

Quando você usa a API para obter dados sigilosos, o Insights de Performance utiliza as credenciais do chamador. Essa verificação garante que o acesso a dados sigilosos seja limitado àqueles com acesso à chave do KMS.

Ao chamar essas APIs, você precisa de permissões para chamar a API por meio da política do IAM e de permissões para invocar a ação `kms:decrypt` por meio da política de chave AWS KMS.

A API `GetResourceMetrics` pode retornar dados sigilosos e não sigilosos. Os parâmetros da solicitação determinam se a resposta deve incluir dados sigilosos. A API retorna dados sigilosos quando a solicitação inclui uma dimensão confidencial nos parâmetros filtrar ou agrupar por.

Para obter mais informações sobre as dimensões que você pode usar com a API `GetResourceMetrics`, consulte DimensionGroup.

Example Exemplos

O seguinte exemplo solicita dados sigilosos para o grupo `db.user`:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

O seguinte exemplo solicita dados não sigilosos para o grupo `db.load.avg`:

```
POST / HTTP/1.1
Host: <Hostname>
```

```
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Atribuir acesso refinado para o Insights de Performance

O controle de acesso refinado oferece formas adicionais de controlar o acesso para o Insights de Performance. Esse controle de acesso pode permitir ou negar acesso a dimensões individuais para as ações `GetResourceMetrics`, `DescribeDimensionKeys` e `GetDimensionKeyDetails` do Insights de Performance. Para usar o acesso refinado, especifique as dimensões na política do IAM usando chaves de condição. A avaliação do acesso segue a lógica de avaliação da política do IAM. Para acessar mais informações, consulte Lógica da avaliação de política no Guia do usuário do IAM. Se a declaração de política do IAM não especificar nenhuma dimensão, a declaração controlará o acesso a todas as dimensões da ação especificada. Para ver a lista de dimensões disponíveis, consulte DimensionGroup.

Para descobrir as dimensões que as credenciais estão autorizadas a acessar, use o parâmetro `AuthorizedActions` em `ListAvailableResourceDimensions` e especifique a ação. Os valores permitidos para `AuthorizedActions` são os seguintes:

- `GetResourceMetrics`
- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`

Por exemplo, se você especificar `GetResourceMetrics` como o parâmetro `AuthorizedActions`, `ListAvailableResourceDimensions` exibirá a lista de dimensões que a ação `GetResourceMetrics` está autorizada a acessar. Se você especificar várias ações no parâmetro `AuthorizedActions`, `ListAvailableResourceDimensions` exibirá uma interseção de dimensões que essas ações estão autorizadas a acessar.

Example

O exemplo a seguir concede acesso às dimensões especificadas para as ações `GetResourceMetrics` e `DescribeDimensionKeys`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
```

```
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

Veja abaixo a resposta para a dimensão solicitada:

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

O exemplo a seguir especifica uma permissão e duas negações de acesso às dimensões.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
```

```
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
}
```

Veja abaixo a resposta para as dimensões solicitadas:

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
```

```
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
```

```
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

Usar controle de acesso baseado em tags para o Insights de Performance

É possível controlar o acesso às métricas do Insights de Performance usando tags herdadas da instância de banco de dados principal. Para controlar o acesso às operações do Insights de Performance, use as políticas do IAM. Essas políticas podem conferir as tags na sua instância de banco de dados para determinar as permissões.

Como as tags funcionam com o Insights de Performance

O Insights de Performance aplica automaticamente suas tags de instância de banco de dados para autorizar as respectivas métricas. Ao adicionar tags à sua instância de banco de dados, você pode usá-las imediatamente para controlar o acesso aos dados do Insights de Performance.

- Para adicionar ou atualizar tags para métricas do Insights de Performance, modifique as tags na sua instância de banco de dados.
- Para ver as tags das métricas do Insights de Performance, chame `ListTagsForResource` no recurso de métricas da solução. Ele exibirá as tags da instância de banco de dados associadas à métrica.

> **Note**
>
> As operações `TagResource` e `UntagResource` exibirão um erro se você tentar usá-las diretamente nas métricas do Insights de Performance.

Criar políticas do IAM baseadas em tag

Para controlar o acesso às operações do Insights de Performance, use a chave de condição `aws:ResourceTag` nas políticas do IAM. Essas políticas conferem as tags na sua instância de banco de dados.

Example

Essa política impede o acesso às métricas do Insights de Performance para bancos de dados de produção. A política nega a operação `pi:GetResourceMetrics` no Insights de Performance para qualquer recurso de banco de dados marcado com `env:prod`.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Deny",
           "Action": "pi:GetResourceMetrics",
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:ResourceTag/env": "prod"
               }
           }
       }
   ]
}
```

Análise de métricas usando o painel do Performance Insights

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de junho de 2026. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos

períodos de retenção flexíveis (de 1 a 24 meses) e aos preços associados do Insights de Performance. A API do Insights de Performance continuará existindo sem alterações de preço. Os custos da API do Insights de Performance aparecerão em sua fatura da com o custo do CloudWatch Database Insights.
Recomendamos que você atualize todos os clusters de banco de dados usando o nível pago do Insights de Performance para o modo avançado do Database Insights antes de 30 de junho de 2026. Para ter informações sobre como atualizar para o modo avançado do Database Insights, consulte Ativação do modo Avançado do Database Insights para Amazon Aurora.
Se você não tomar nenhuma medida, os clusters de banco de dados que estão usando o Insights de Performance usarão como padrão o modo padrão do Database Insights. Com o modo padrão do Database Insights, você pode perder o acesso ao histórico de dados de desempenho após sete dias e talvez não consiga usar planos de execução e recursos de análise sob demanda no console do Amazon RDS. Após 30 de junho de 2026, somente o modo avançado do Database Insights aceitará planos de execução e análise sob demanda. Com o CloudWatch Database Insights, você pode monitorar a carga do banco de dados para sua frota de bancos de dados e analisar e solucionar problemas de desempenho em grande escala. Para ter mais informações sobre o Database Insights, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights. Para obter informações de definição de preço, consulte Definição de preço do Amazon CloudWatch.

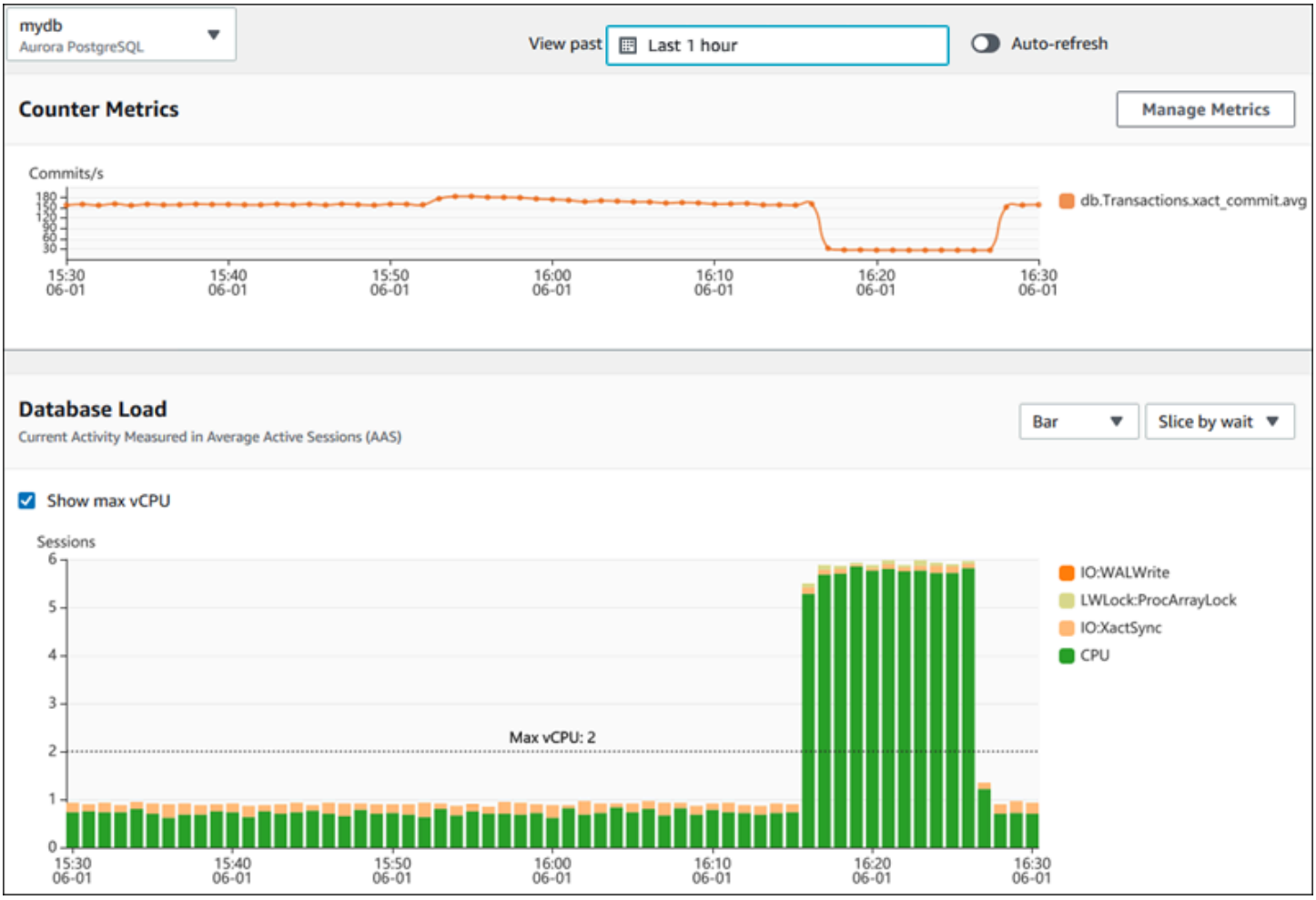
O painel do Performance Insights contém informações de performance do banco de dados para ajudar você a analisar e solucionar problemas de performance. Na página principal do painel, você pode visualizar informações sobre a carga do banco de dados. Você pode separar a carga de banco de dados por dimensões como eventos de espera ou SQL.

Painel do Performance Insights

- Visão geral do painel do Performance Insights
- Acessar o painel do Performance Insights.
- Analisar a carga do banco de dados por eventos de espera
- Analisar a performance do banco de dados por um período
- Analisar consultas com a guia SQL principal no Insights de Performance

Visão geral do painel do Performance Insights

O painel é a maneira mais fácil de interagir com o Performance Insights. O exemplo a seguir mostra o painel de uma instância de banco de dados do PostgreSQL.



Tópicos

- Filtro de intervalo de tempo
- Gráfico de métricas de contador
- Gráfico de carga do banco de dados
- Tabela Top dimensions (Principais dimensões)

Filtro de intervalo de tempo

Por padrão, o painel do Performance Insights exibe a carga de banco de dados da última hora. Você pode ajustar esse intervalo para cinco minutos ou dois anos. Também é possível selecionar um intervalo relativo personalizado.

Você pode selecionar um intervalo absoluto com data e hora de início e término. O exemplo a seguir mostra o intervalo de tempo que começa à meia-noite de 25/9/24 e termina às 23h59 de 28/9/24.

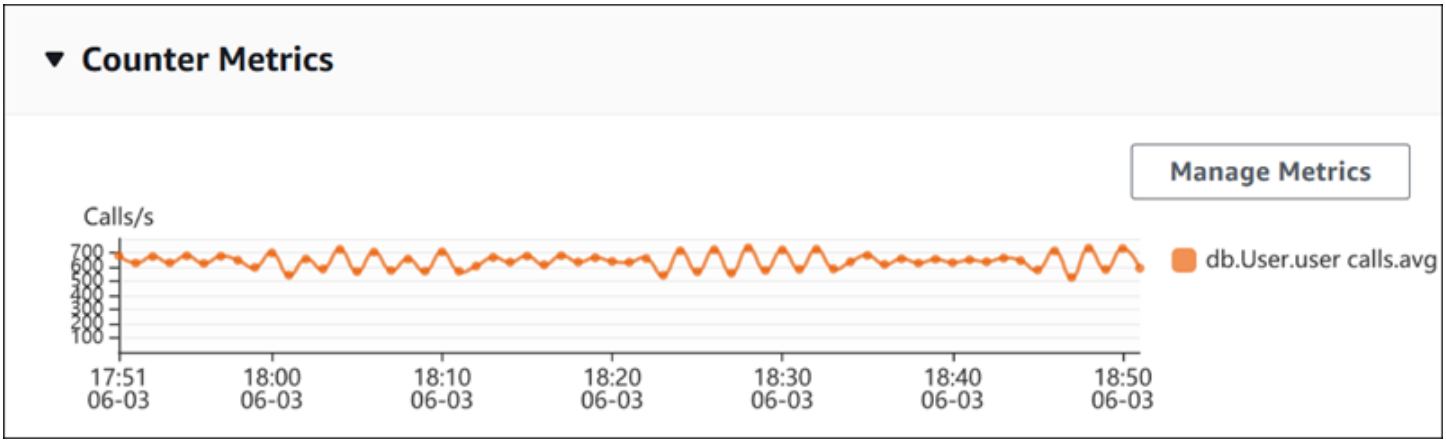
Por padrão, o fuso horário do painel do Insights de Performance é o Tempo Universal Coordenado (UTC). Também é possível escolher o fuso horário local.

Gráfico de métricas de contador

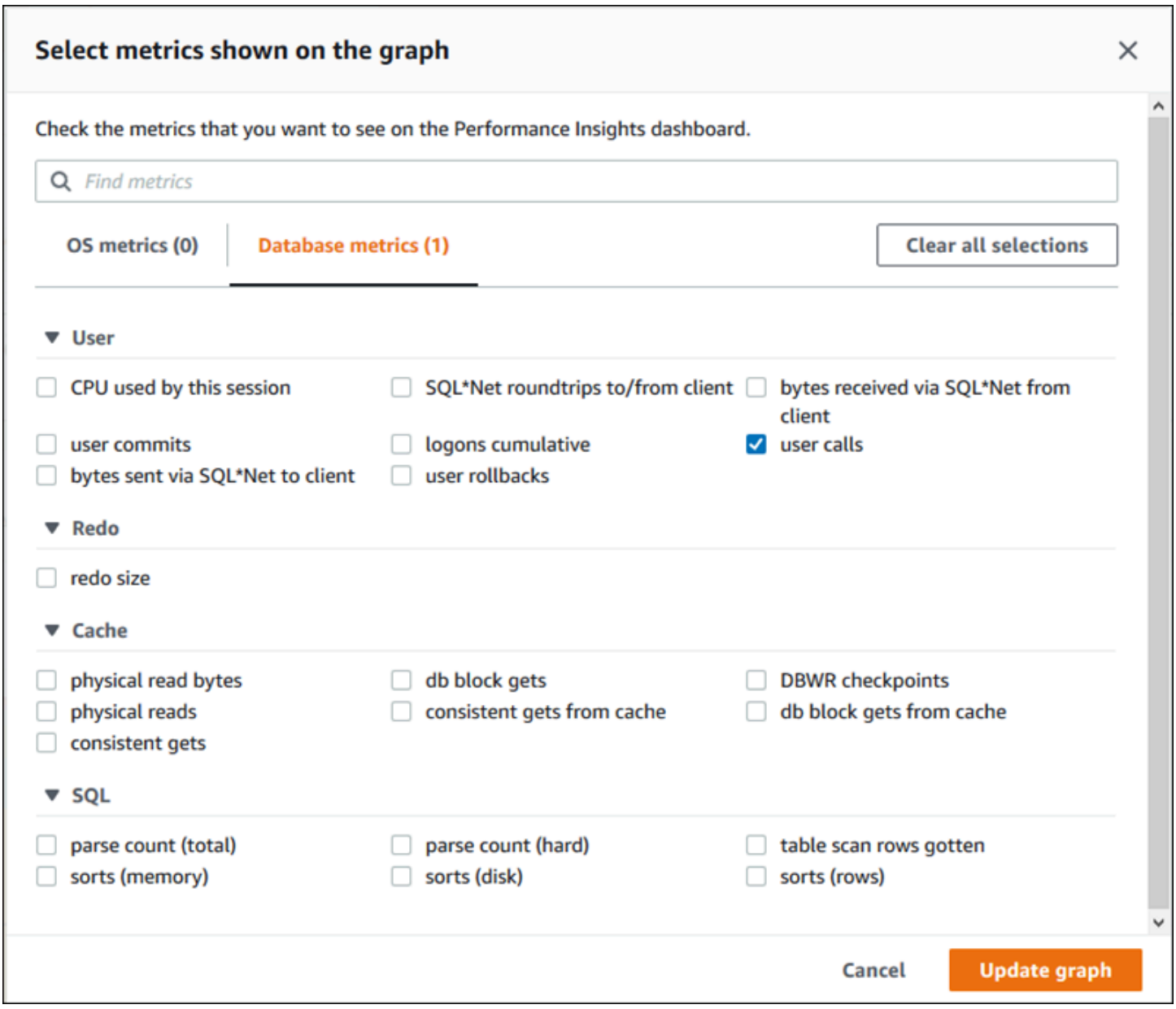
Com métricas de contador, você pode personalizar o painel do Performance Insights para incluir até 10 gráficos adicionais. Esses gráficos mostram uma seleção de dezenas de métricas de performance do sistema operacional e do banco de dados. É possível correlacionar essas informações à carga do banco de dados para ajudar a identificar e analisar problemas de performance.

O gráfico Counter Metrics (Métricas de contador) exibe dados dos contadores de performance. As métricas padrão dependem do mecanismo de banco de dados:

- Aurora MySQL– `db.SQL.Innodb_rows_read.avg`
- Aurora PostgreSQL – `db.Transactions.xact_commit.avg`



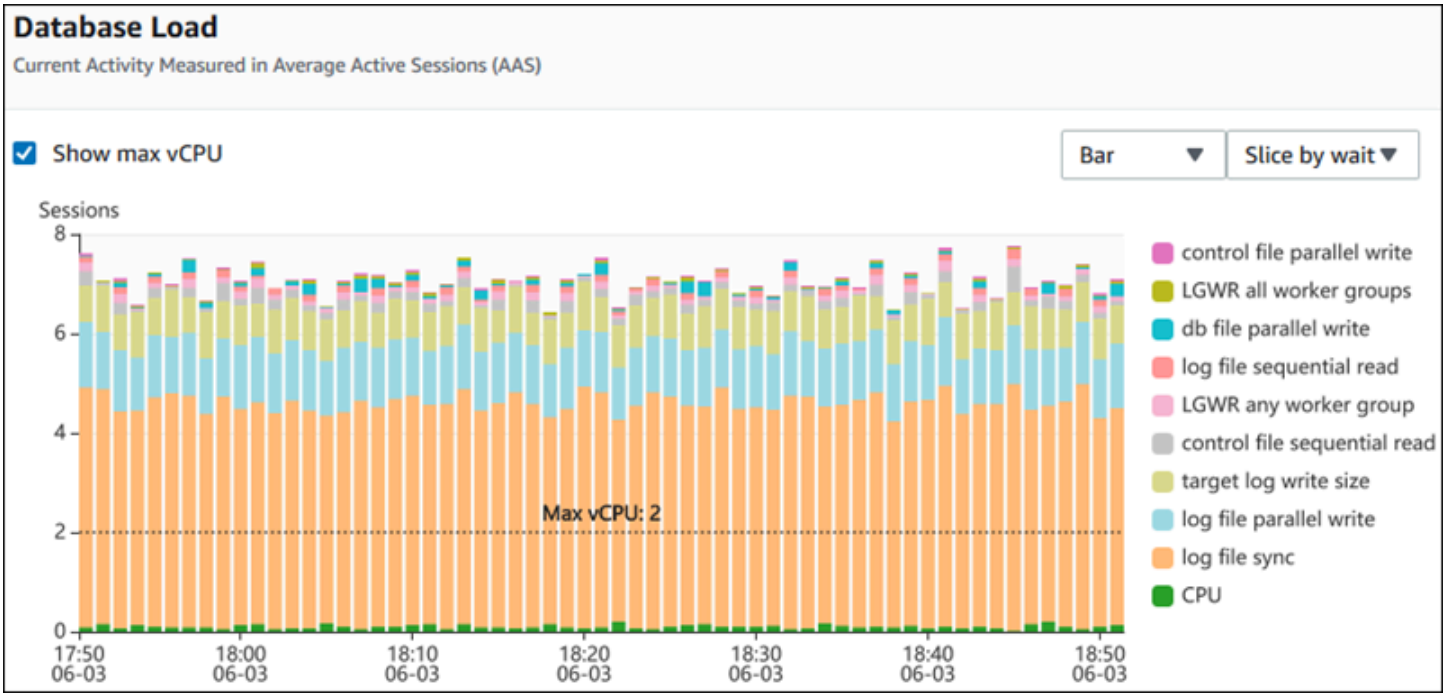
Para alterar os contadores de performance, escolha Manage Metrics (Gerenciar métricas). É possível selecionar várias Métricas de SO ou Métricas de banco de dados, conforme mostrado na captura de tela a seguir. Para ver detalhes de qualquer métrica, passe o mouse sobre o nome da métrica.



Para obter descrições das métricas de contador que você pode adicionar a cada mecanismo de banco de dados, consulte Métricas de contadores do Performance Insights.

Gráfico de carga do banco de dados

O gráfico Database load (Carga do banco de dados) mostra como a atividade do banco de dados se compara à capacidade da instância de banco de dados representada pela linha Max vCPU (Máximo de vCPU). Por padrão, o gráfico de linhas empilhadas representa a carga do banco de dados como sessões ativas médias por unidade de tempo. A carga do banco de dados é separada (agrupada) por estados de espera.

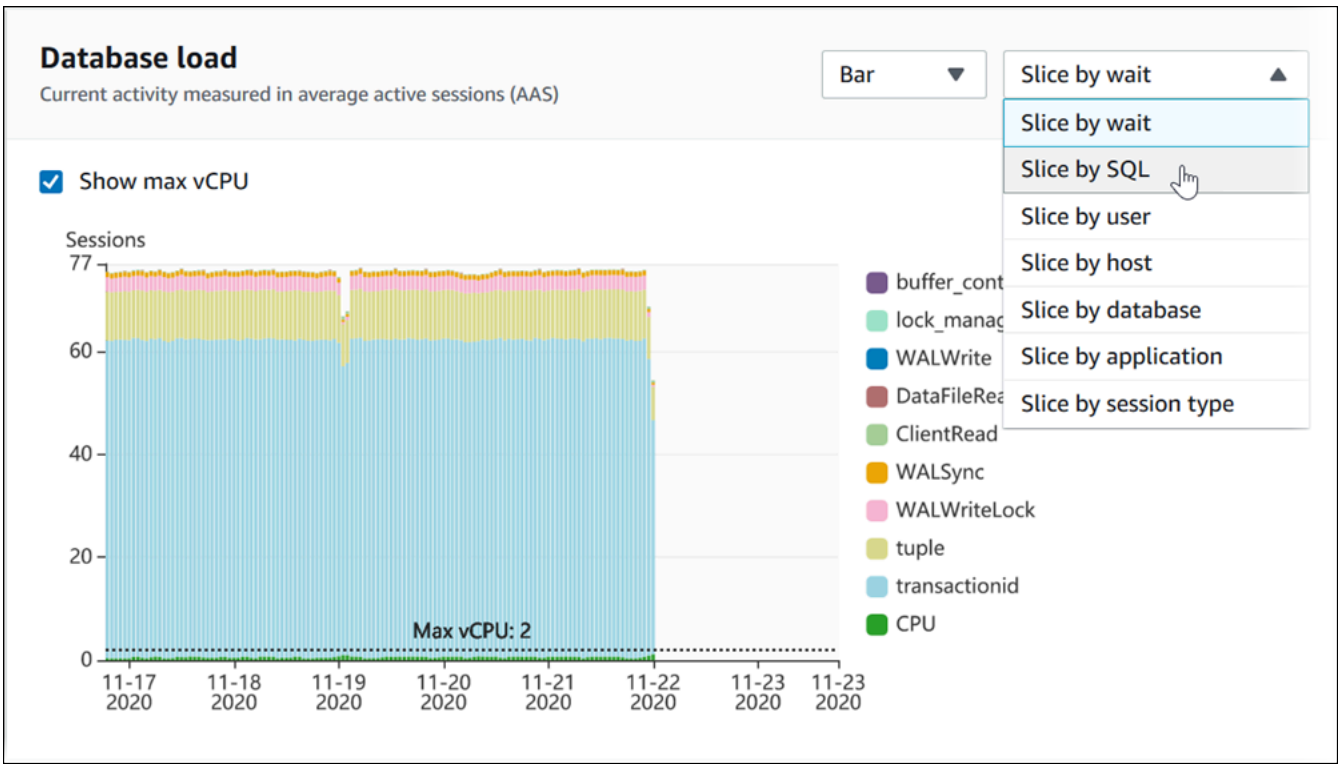


Carga de banco de dados separada por dimensões

Você pode optar por exibir a carga como sessões ativas agrupadas por quaisquer dimensões aceitas. A tabela a seguir mostra quais dimensões são aceitas pelos diferentes mecanismos.

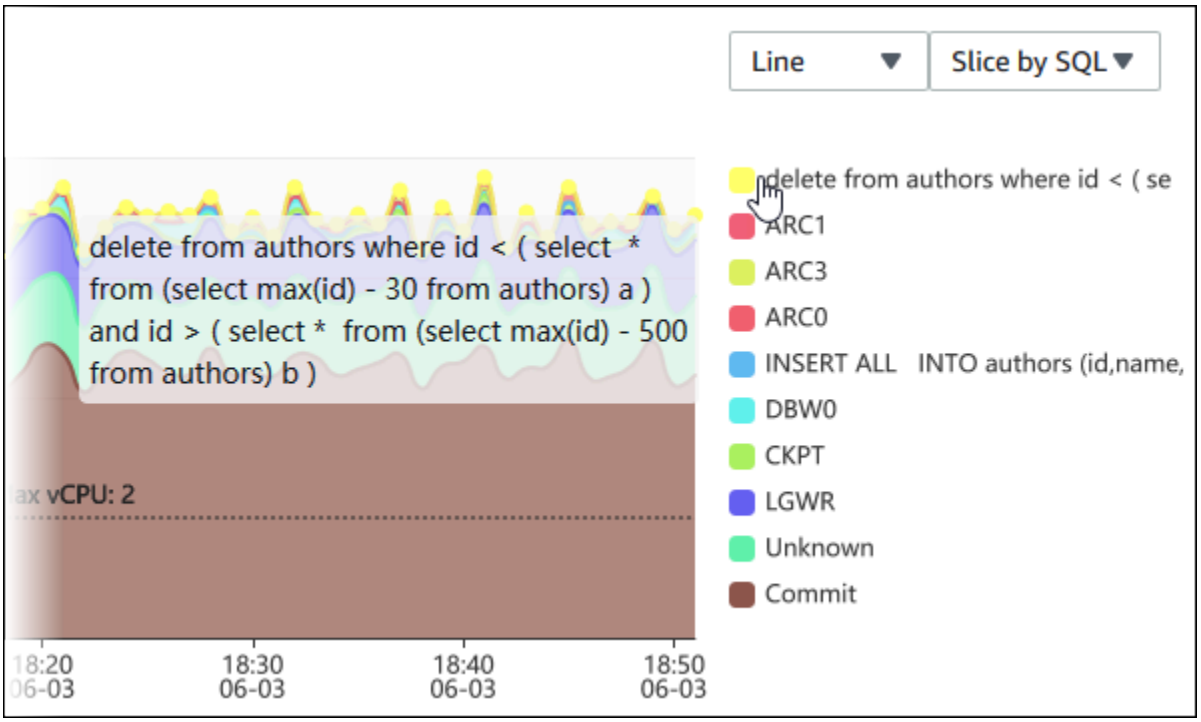
Dimensão	Aurora PostgreSQL	Aurora MySQL
Host	Sim	Sim
SQL	Sim	Sim
Usuário	Sim	Sim
Espera	Sim	Sim
Aplicação	Sim	Não
Banco de dados	Sim	Sim
Tipo de sessão	Sim	Não

A imagem a seguir mostra as dimensões de uma instância de banco de dados do PostgreSQL.



Detalhes de carga de banco de dados para um item de dimensão

Para ver detalhes sobre um item de carga de banco de dados dentro de uma dimensão, passe o mouse sobre o nome do item. A imagem a seguir mostra detalhes de uma instrução SQL.



Para ver detalhes de qualquer item do período selecionado na legenda, passe o mouse sobre esse item.

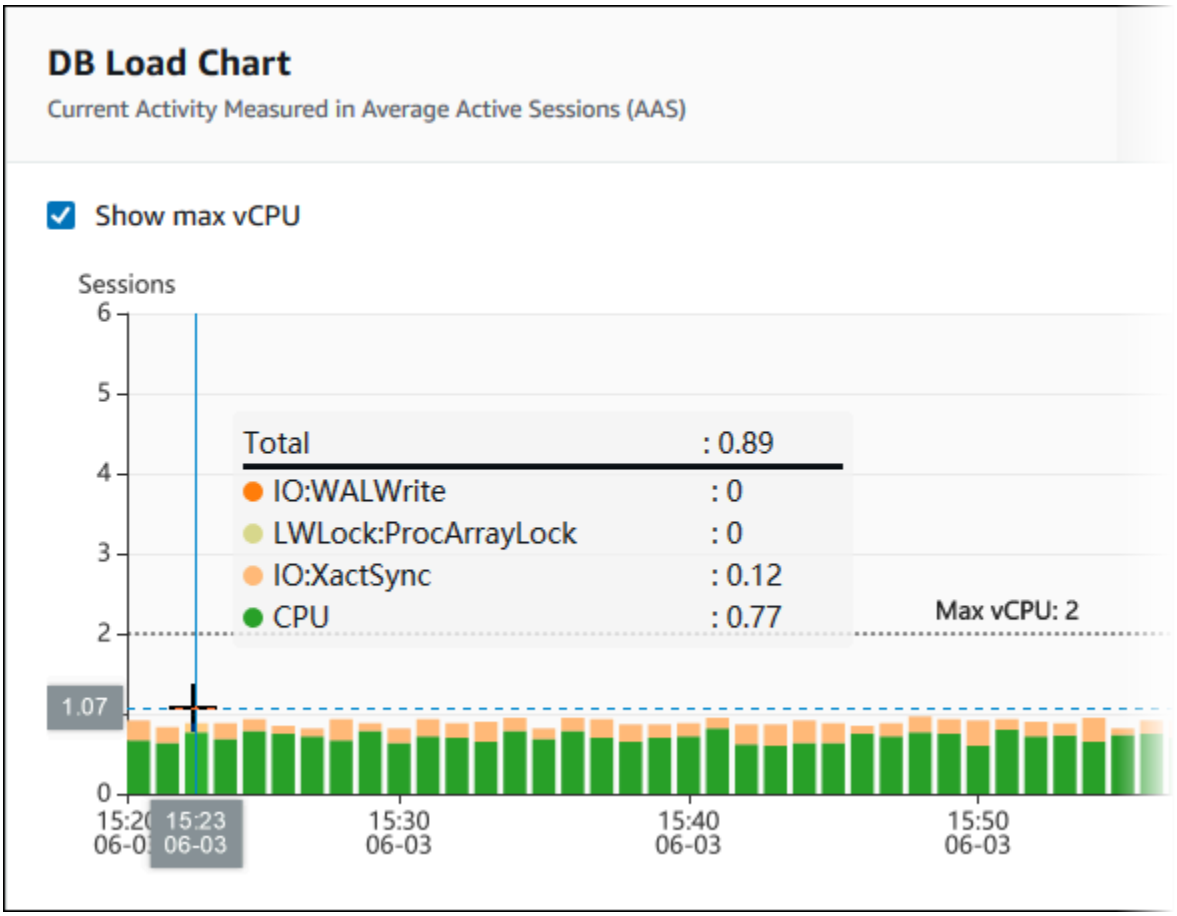
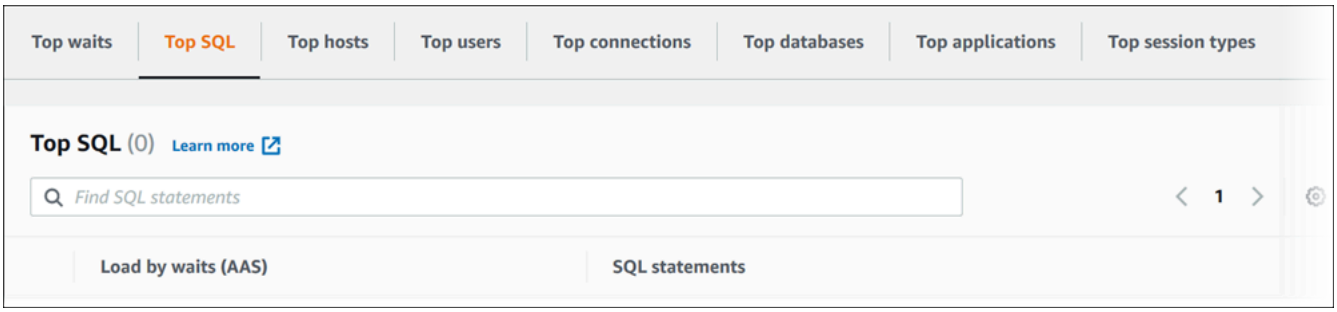


Tabela Top dimensions (Principais dimensões)

A tabela Top dimensions (Principais dimensões) separa a carga do banco de dados com base em diferentes dimensões. Uma dimensão é uma categoria ou “pedaços” de diferentes características de uma carga de banco de dados. Se a dimensão for SQL, Top SQL (SQL principal) mostrará as instruções SQL que mais contribuem para a carga do banco de dados.



Escolha qualquer uma das guias de dimensão a seguir.

Tab	Descrição	Mecanismos com suporte
SQL principal	As instruções SQL que estão sendo executadas no momento	Todos
Esperas principais	O evento para o qual o backend do banco de dados está aguardando	Todos
Hosts principais	O nome do host do cliente conectado	Todos
Principais usuários	O usuário conectado ao banco de dados	Todos
Principais aplicações	O nome da aplicação que está conectada ao banco de dados	Somente Aurora PostgreSQL
Principais tipos de sessão	O tipo da sessão atual	Apenas Aurora PostgreSQL

Para aprender a analisar consultas utilizando a guia Top SQL (SQL principal), consulte Visão geral da guia Top SQL (SQL principal).

Acessar o painel do Performance Insights.

O Amazon RDS fornece uma visão consolidada das métricas do Insights de Performance e do CloudWatch no painel do Insights de Performance.

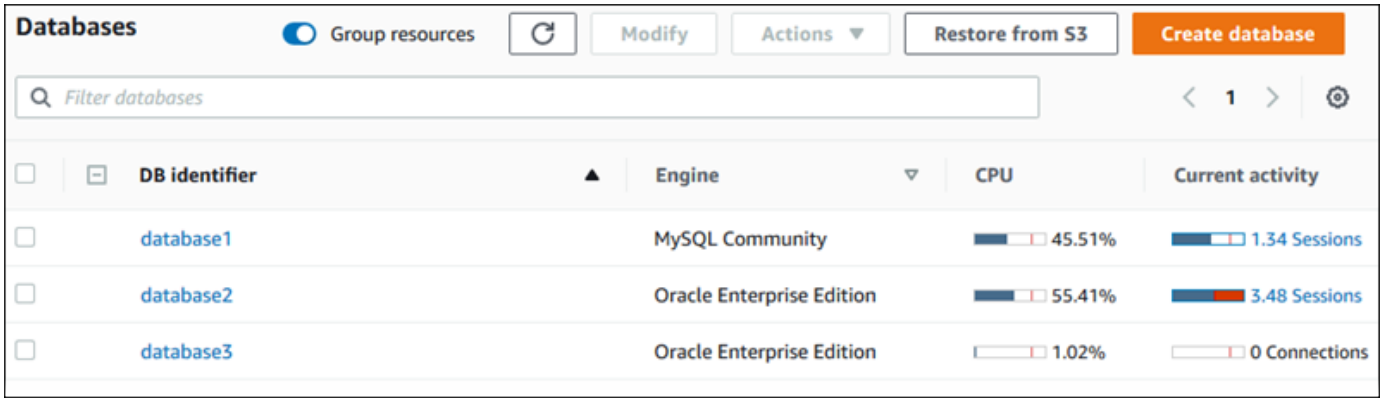
Para acessar o painel do Performance Insights, use o procedimento a seguir.

Para visualizar o painel do Performance Insights no Console de gerenciamento da AWS

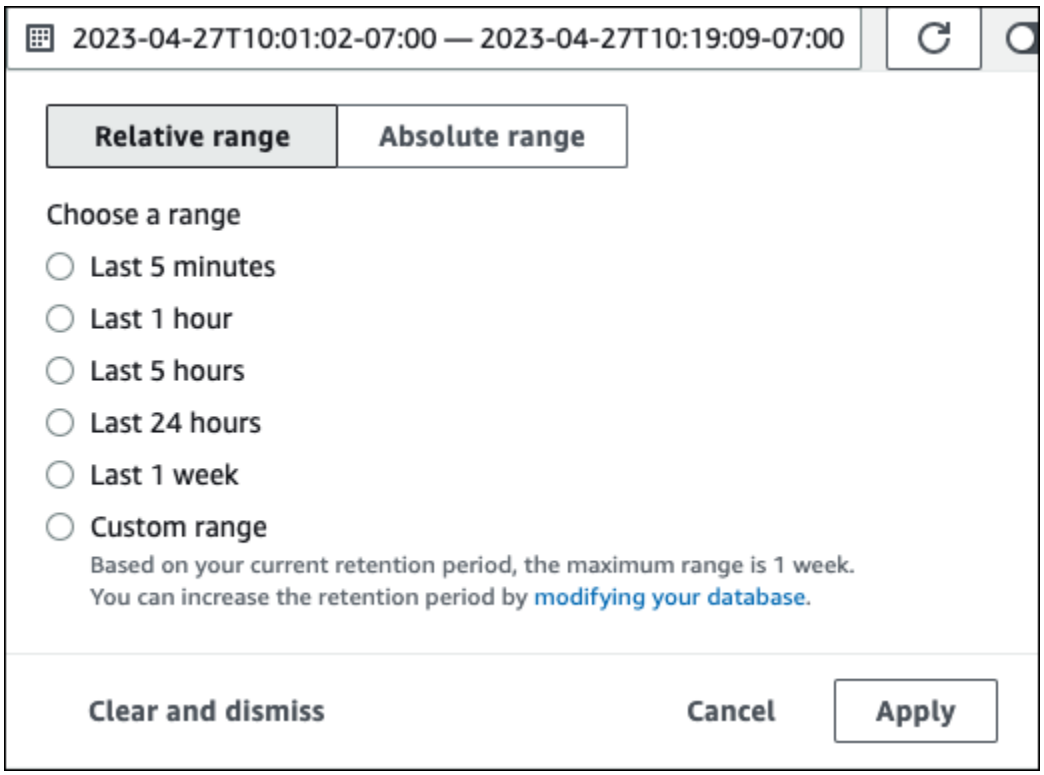
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

   Para as instâncias de banco de dados com o Insights de Performance ativado, você também pode acessar o painel do Insights de Performance escolhendo o item Sessões na lista de

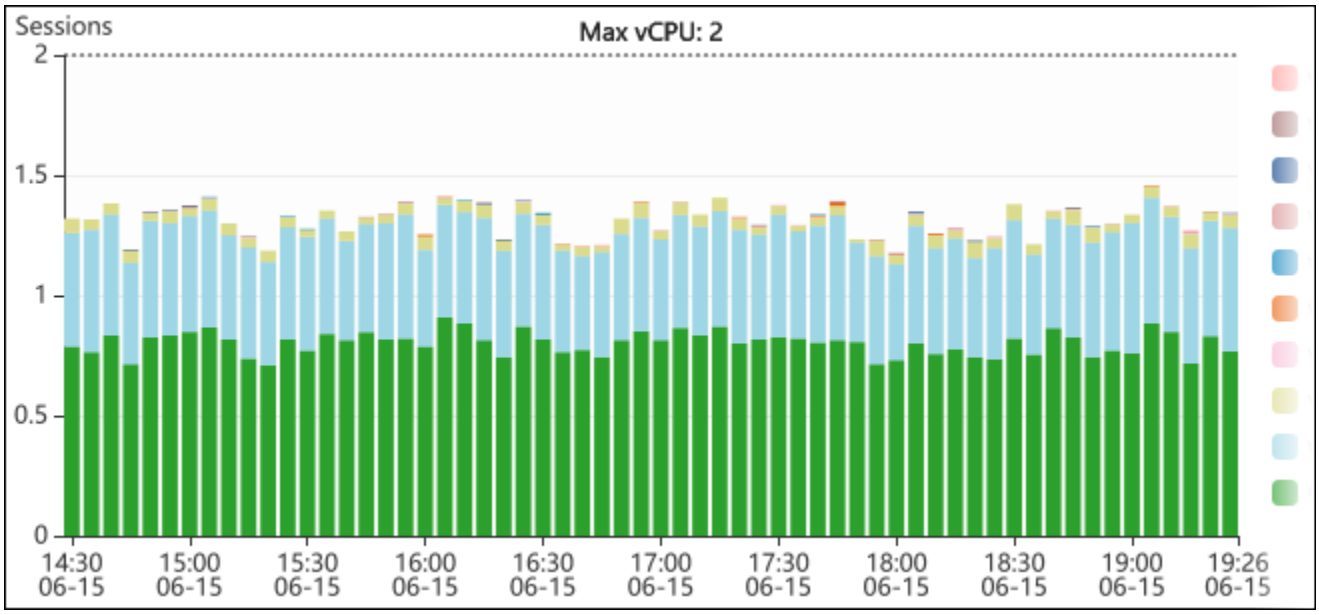
instâncias de banco de dados. Em Atividade atual, o item Sessões mostra a carga de banco de dados em sessões ativas médias nos últimos cinco minutos. A carga é mostrada graficamente por meio de barras. Quando a barra está vazia, a instância de banco de dados está ociosa. À medida que a carga aumenta, a barra é preenchida com a cor azul. Quando a carga ultrapassa o número de CPUs virtuais (vCPUs) na classe da instância de banco de dados, a barra se torna vermelha, indicando um possível gargalo.



4. (Opcional) Escolha a data ou o intervalo de tempo no canto superior direito e especifique um intervalo de tempo relativo ou absoluto diferente. Agora você pode especificar um período e gerar um relatório de análise de performance do banco de dados. O relatório fornece as recomendações e os insights identificados. Para ter mais informações, consulte Criar um relatório de análise de performance no Insights de Performance.

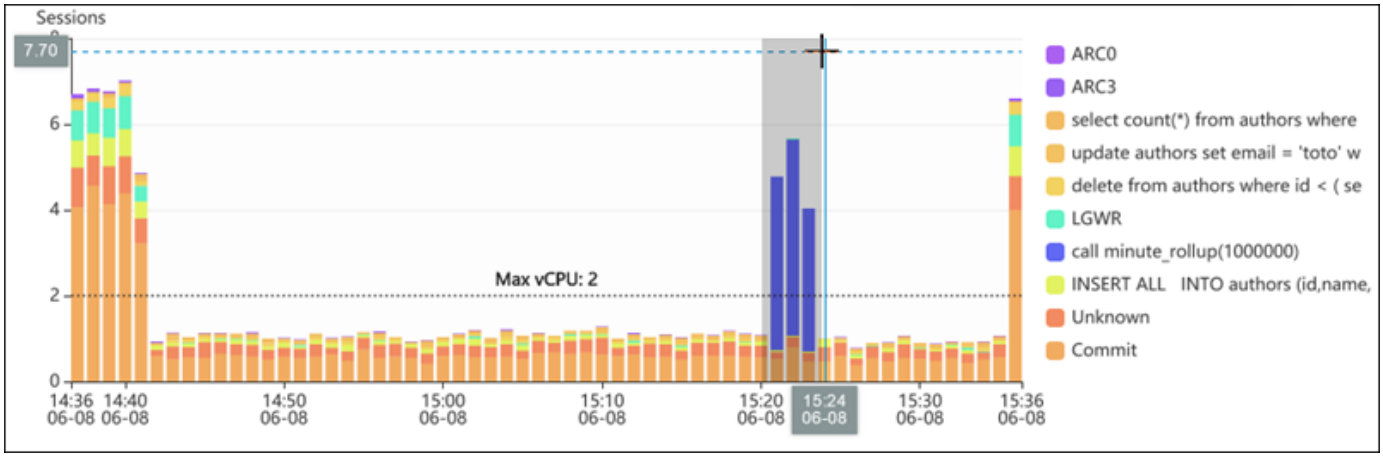


Na captura de tela a seguir, o intervalo da carga do banco de dados é de 5 horas.

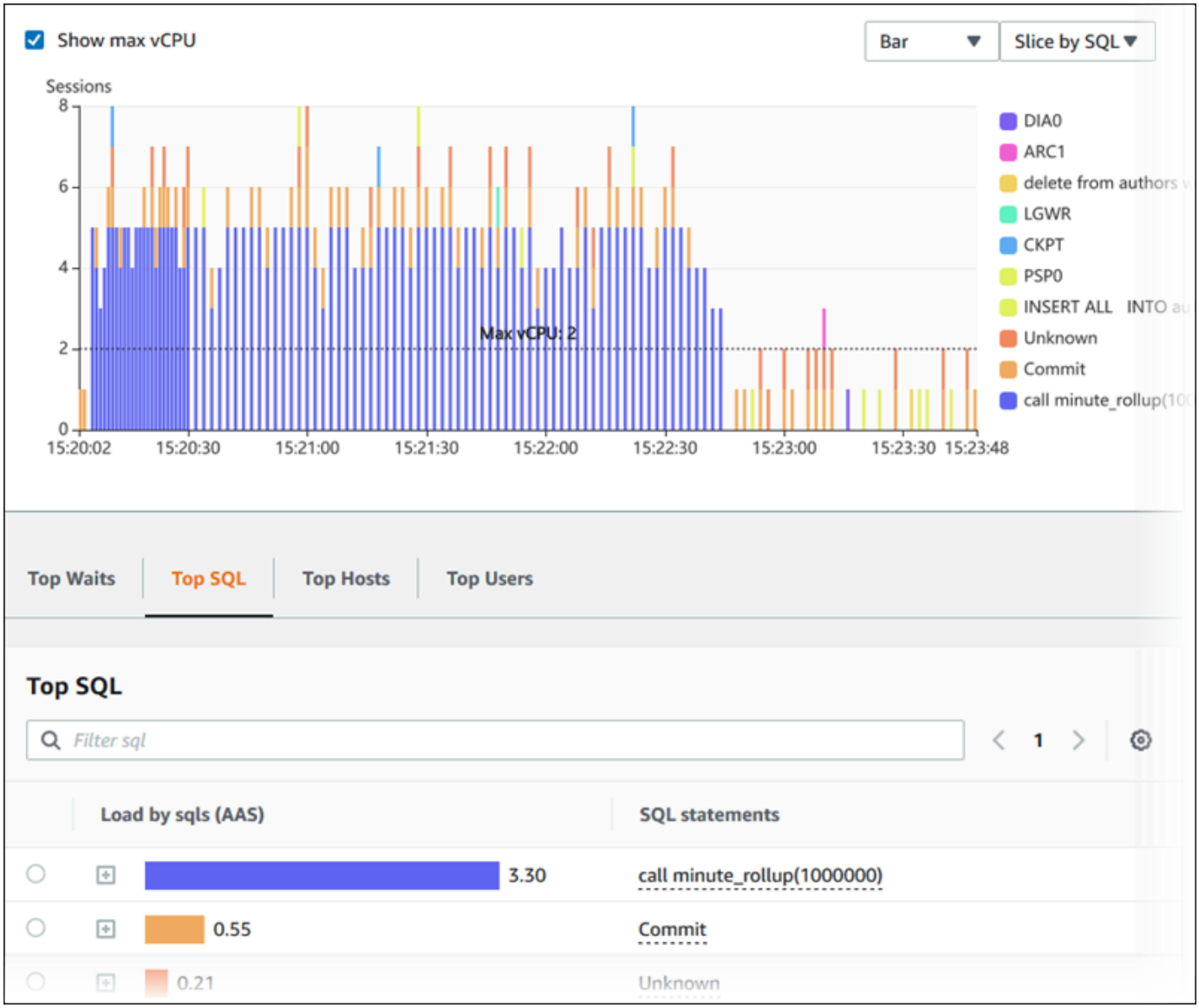


5. (Opcional) Para ampliar uma parte do grafo de carga de banco de dados, escolha a hora de início e arraste até o final do período desejado.

   A área selecionada é destacada no grafo de carga de banco de dados.



   Ao soltar o mouse, o grafo de carga de banco de dados amplia a Região da AWS selecionada e a tabela Top dimensions (Dimensões principais) é recalculada.



6. (Opcional) Para atualizar seus dados automaticamente, selecione Atualização automática.

O painel de Insights de Performance é atualizado automaticamente com novos dados. A taxa de atualização depende da quantidade de dados exibida:

- 5 minutos atualiza a cada 10 segundos.
- 1 hora atualiza a cada 5 minutos.
- 5 horas atualiza a cada 5 minutos.
- 24 horas atualiza a cada 30 minutos.
- 1 semana atualiza a cada hora.

- 1 mês atualiza todos os dias.

Analisar a carga do banco de dados por eventos de espera

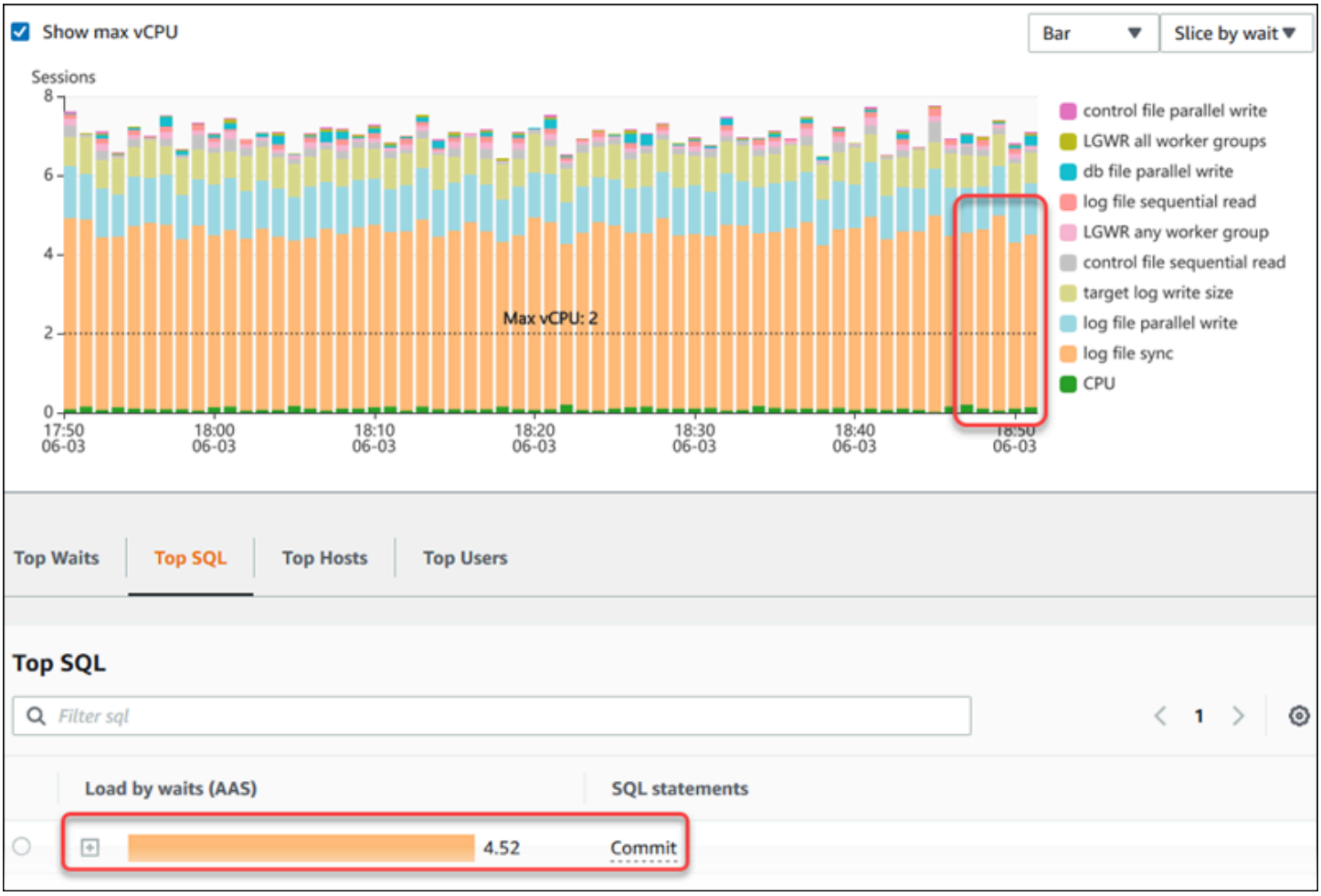
Se o gráfico Database load (Carga do banco de dados) mostrar um gargalo, você poderá descobrir de onde vem essa carga. Para fazer isso, examine a tabela de principais itens de carga abaixo do gráfico Database load (Carga do banco de dados). Escolha um item específico, como uma consulta SQL ou um usuário, para ver detalhes sobre ele.

A carga de banco de dados agrupada por espera e as principais consultas SQL compõem a visualização padrão do painel do Performance Insights. Em geral, essa combinação fornece os melhores insights sobre problemas de performance. A carga de banco de dados agrupada por espera mostra se há algum gargalo de recursos ou de concorrências no banco de dados. Nesse caso, a guia SQL da tabela Top Load Items (Principais itens de carga) mostra quais consultas estão gerando essa carga.

Seu fluxo de trabalho típico para diagnosticar problemas de performance é o seguinte:

1. Analise o gráfico Carregamento do banco de dados e veja se há casos de cargas de banco de dados que estejam ultrapassando a linha Máximo de CPU.
2. Se houver, examine o gráfico Database load (Carga do banco de dados) e identifique quais estados de espera são os principais responsáveis por isso.
3. Identifique as consultas resumidas que estão gerando a carga examinando quais consultas na guia SQL da tabela Top Load Items (Principais itens de carga) estão contribuindo mais para aqueles estados de espera. Você pode identificar essas consultas na coluna DB Load by Wait (Carga de banco de dados por espera).
4. Escolha uma dessas consultas resumidas na guia SQL para expandi-la e exibir as consultas secundárias que a compõem.

Por exemplo, no painel a seguir, as esperas de sincronização de arquivos de log compõem a maior parte da carga de banco de dados. A espera de Todos os grupos de operador de LGWR também é alta. O gráfico Top SQL (SQL principal) exibe o que está provocando as esperas de sincronização de arquivos de log: instruções `COMMIT` frequentes. Nesse caso, a confirmação com menos frequência reduzirá a carga de banco de dados.



Analisar a performance do banco de dados por um período

Analise a performance do banco de dados com análise sob demanda criando um relatório de análise de performance por um período. Visualize os relatórios de análise de performance para descobrir problemas de performance, como gargalos de recursos ou alterações em uma consulta na instância de banco de dados. O painel do Insights de Performance permite que você selecione um período e crie um relatório de análise de performance. Também é possível adicionar uma ou mais tags ao relatório.

Para usar esse atributo, você deve usar o período de retenção do nível pago. Para ter mais informações, consulte Preços e retenção de dados para o Performance Insights.

O relatório está disponível na guia Relatórios de análise de performance: novo para selecionar e visualizar. O relatório contém os insights, as métricas relacionadas e as recomendações para resolver o problema de performance. O relatório fica disponível para visualização durante o período de retenção do Insights de Performance.

O relatório será excluído se a hora de início do período de análise do relatório estiver fora do período de retenção. Você também pode excluir o relatório antes que o período de retenção termine.

Para detectar os problemas de performance e gerar o relatório de análise para sua instância de banco de dados, você deve ativar o Insights de Performance. Para obter mais informações sobre como ativar o Insights de Performance, consulte Ativar e desativar o Insights de Performance para o Aurora.

Para ter informações sobre compatibilidade de regiões, mecanismos de banco de dados e classes de instância com esse recurso, consulte Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance.

Nas seções a seguir, é possível criar, visualizar, adicionar tags e excluir um relatório de análise de performance.

Tópicos

- Criar um relatório de análise de performance no Insights de Performance
- Visualizar um relatório de análise de performance no Insights de Performance
- Adicionar tags a um relatório de análise de performance no Insights de Performance
- Excluir um relatório de análise de performance no Insights de Performance

Criar um relatório de análise de performance no Insights de Performance

Você pode criar um relatório de análise de performance para um período específico no painel do Insights de Performance. Você pode selecionar um período e adicionar uma ou mais tags ao relatório de análise.

O período de análise pode variar de cinco minutos a seis dias. Deve haver pelo menos 24 horas de dados de performance antes do horário de início da análise.

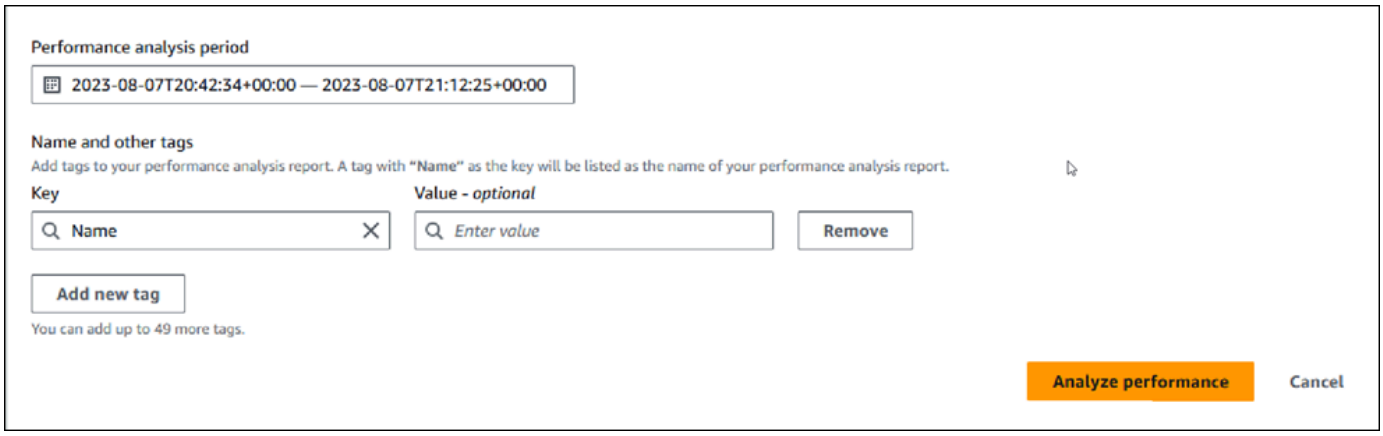
Para ter informações sobre compatibilidade de regiões, mecanismos de banco de dados e classes de instância com esse recurso, consulte Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance.

Como criar um relatório de análise de performance para um período

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.

3. Escolha uma instância de banco de dados.
4. Selecione Analisar performance na seção Carga do banco de dados no painel do Insights de Performance.

   Os campos para definir o período e adicionar uma ou mais tags ao relatório de análise de performance são exibidos.

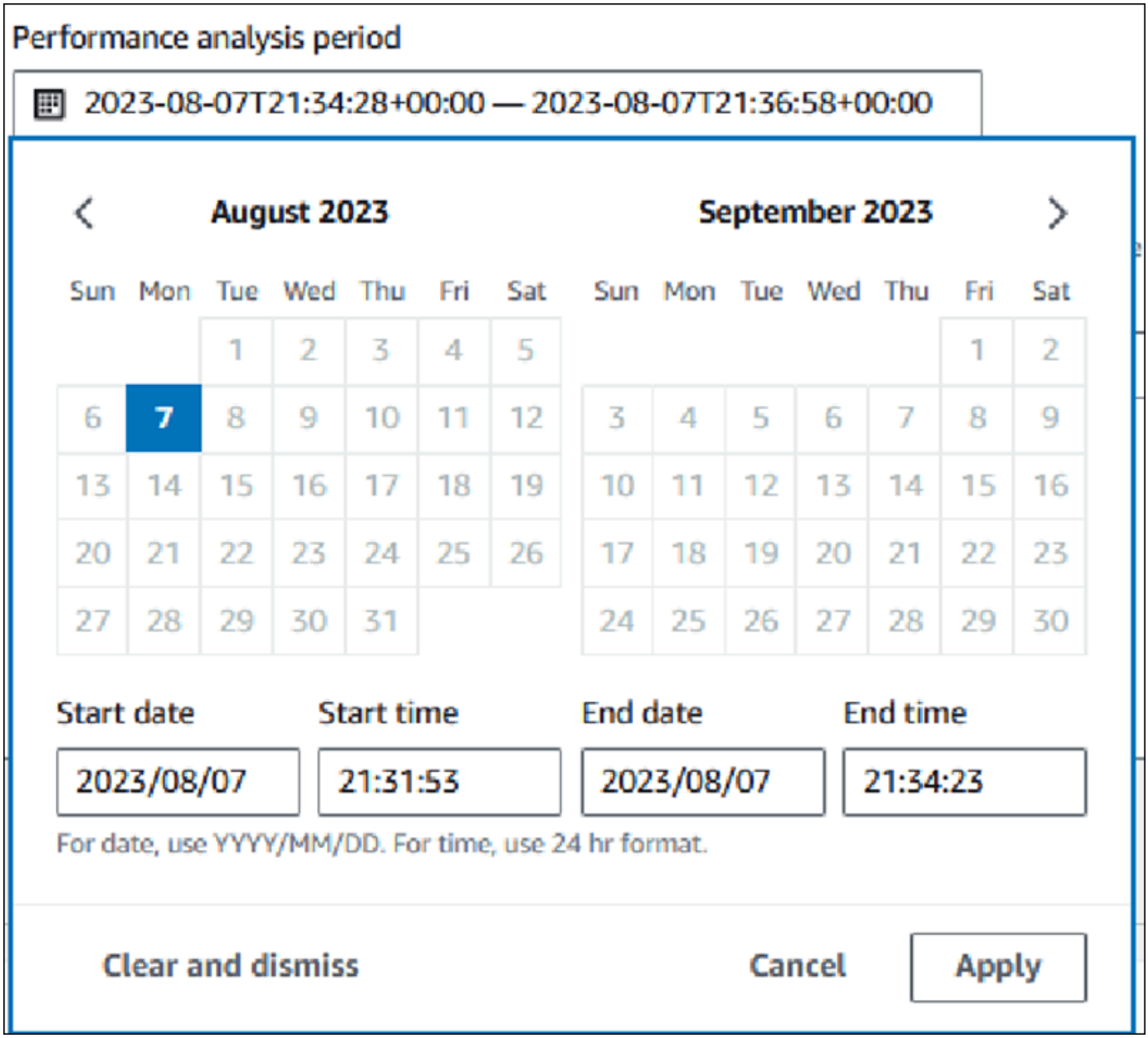


5. Escolha um período. Se definir um período no Intervalo relativo ou no Intervalo absoluto no canto superior direito, você poderá inserir ou selecionar a data e a hora do relatório de análise somente nesse período. Se você selecionar o período de análise fora desse período, uma mensagem de erro será exibida.

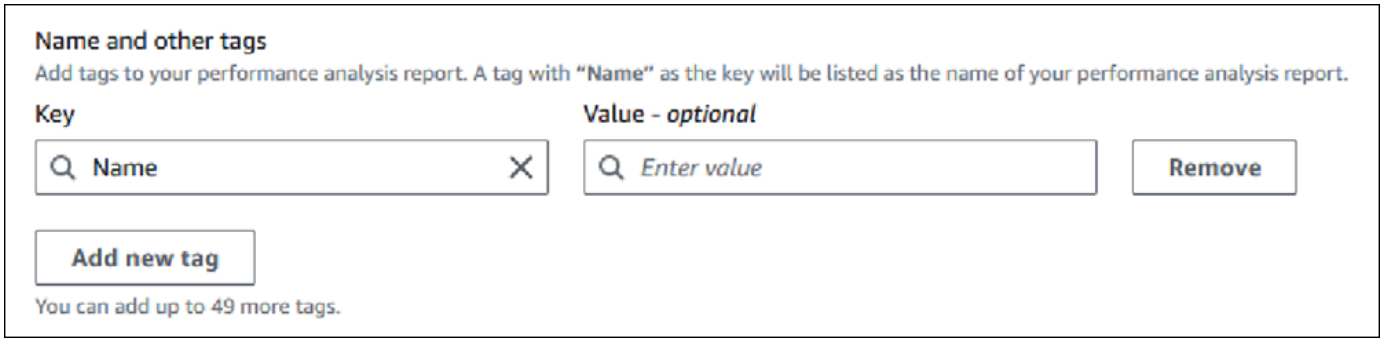
   Para definir o período, você pode fazer o seguinte:

   - Pressione e arraste qualquer um dos controles deslizantes no gráfico de carga do banco de dados.

     A caixa Período de análise de performance exibe o período selecionado, e o gráfico de carga do banco de dados destaca o período selecionado.
   - Selecione a data de início, a hora de início, a data de término e a hora de término na caixa Período de análise de performance.



6. (Opcional) Digite Chave e Valoropcionais para adicionar uma tag para o relatório.



7. Selecione Analisar performance.

   Um banner exibe uma mensagem se a geração do relatório foi bem-sucedida ou falhou. A mensagem também fornece o link para visualizar o relatório.

   O exemplo a seguir mostra o banner com a mensagem de sucesso na criação do relatório.

   O relatório está disponível na guia Relatórios de análise de performance: novo.

Você pode criar um relatório de análise de desempenho usando a AWS CLI. Para obter um exemplo de como criar um relatório usando a AWS CLI, consulte Criar um relatório de análise de performance para um período.

Visualizar um relatório de análise de performance no Insights de Performance

A guia Relatórios de análise de performance: novo lista todos os relatórios criados para a instância de banco de dados. Os itens a seguir são exibidos para cada relatório:

- ID: identificador exclusivo do relatório.
- Nome: chave de tag adicionada ao relatório.
- Hora da criação do relatório: hora em que você criou o relatório.
- Hora de início da análise: hora de início da análise no relatório.
- Hora de término da análise: hora de término da análise no relatório.

Como visualizar um relatório de análise de performance

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Selecione uma instância de banco de dados para a qual você deseja visualizar o relatório de análise.
4. Role para baixo e selecione a guia Relatórios de análises de performance: novo no painel do Insights de Performance.

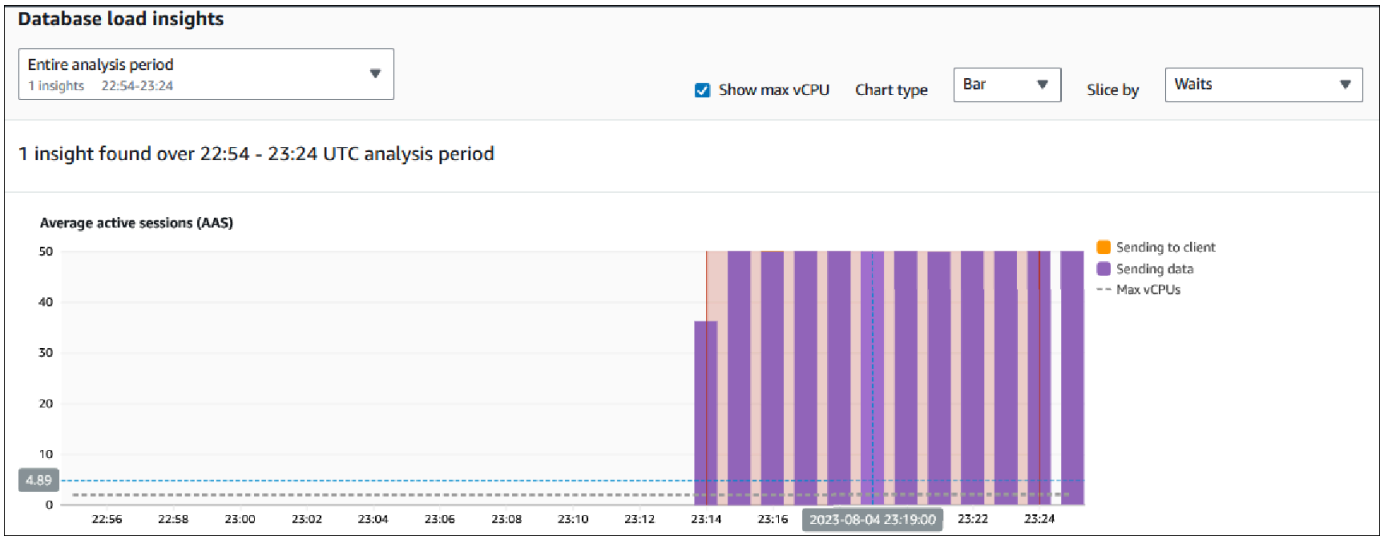
   Todos os relatórios de análise para os diferentes períodos são exibidos.

5. Selecione ID do relatório que você deseja visualizar.

   O gráfico de carga do banco de dados exibirá todo o período de análise por padrão se mais de um insight for identificado. Se o relatório tiver identificado um insight, o gráfico de carga do banco de dados exibirá o insight por padrão.

   O painel também lista as tags do relatório na seção Tags.

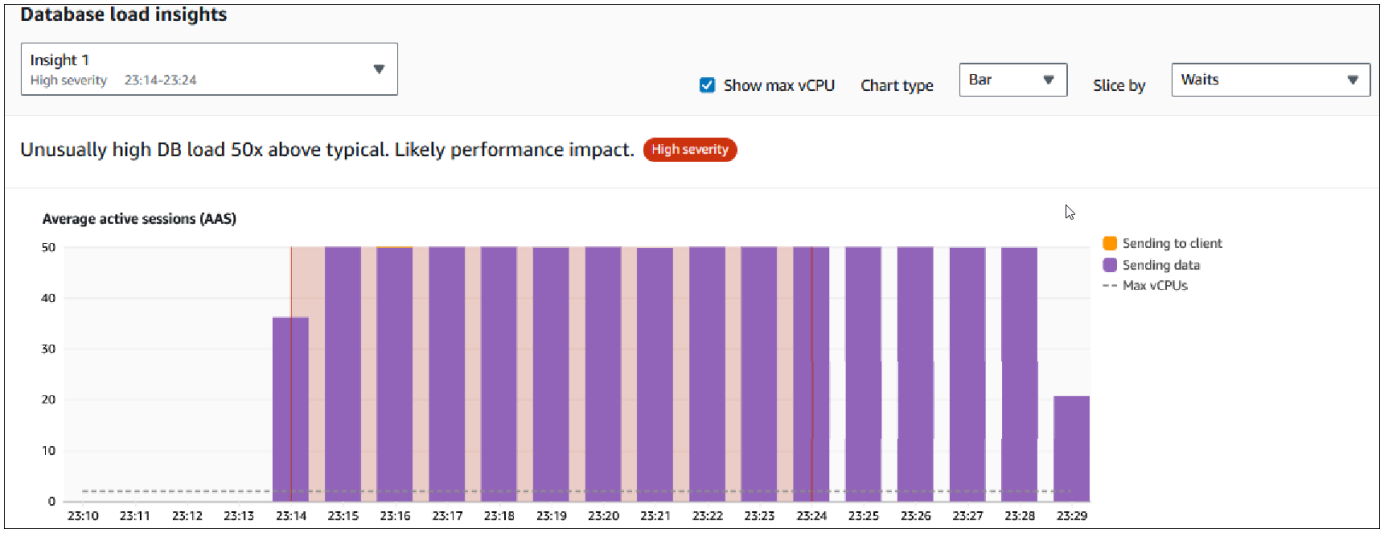
   O exemplo a seguir mostra todo o período de análise do relatório.

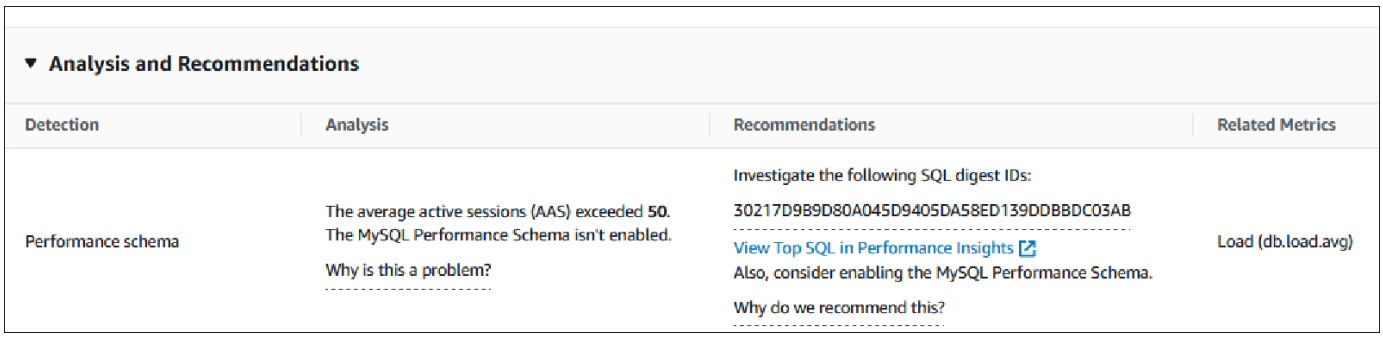


6. Selecione o insight na lista Insights da carga do banco de dados que você deseja visualizar se mais de um insight for identificado no relatório.

   O painel exibe a mensagem de insight, o gráfico de carga do banco de dados destacando o período do insight, a análise e as recomendações, bem como a lista de tags do relatório.

   O exemplo a seguir mostra o insight da carga do banco de dados no relatório.





Adicionar tags a um relatório de análise de performance no Insights de Performance

Você pode adicionar uma tag ao criar ou visualizar um relatório. Você pode adicionar até 50 tags a um relatório.

Você precisa de permissões para adicionar as tags. Para receber mais informações sobre as políticas de acesso para o Insights de Performance, consulte Configurar políticas de acesso para o Performance Insights.

Para adicionar uma ou mais tags ao criar um relatório, consulte a etapa 6 do procedimento Criar um relatório de análise de performance no Insights de Performance.

Como adicionar uma ou mais tags ao visualizar um relatório

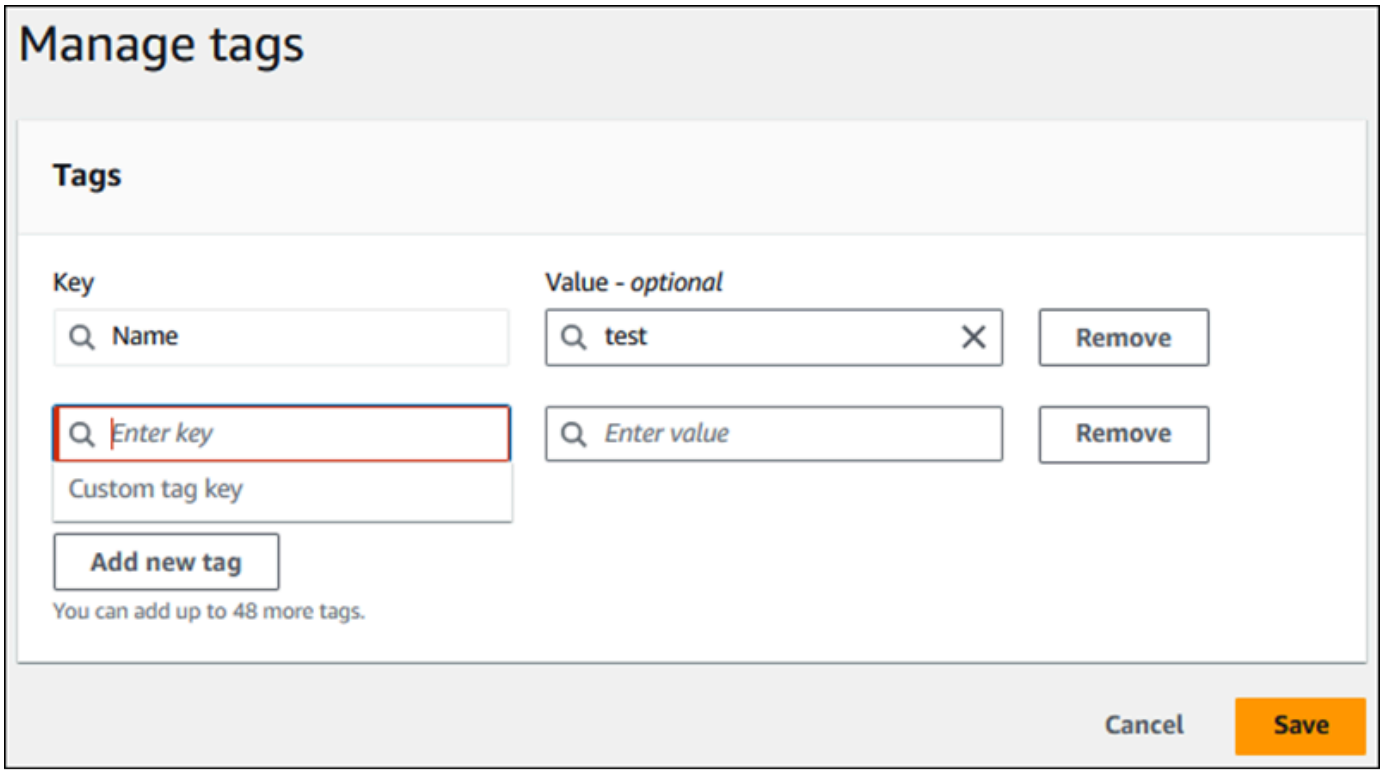
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

O painel do Insights de Performance é exibido para a instância de banco de dados.

4. Role para baixo e selecione a guia Relatórios de análise de performance: novo.
5. Escolha o relatório ao qual você deseja adicionar as tags.

   O painel exibe o relatório.
6. Role para baixo até Tags e selecione Gerenciar tags.
7. Selecione Adicionar nova tag.
8. Insira a Chave e o Valor: opcional e selecione Adicionar nova tag.

   O exemplo a seguir fornece a opção de adicionar uma nova tag ao relatório selecionado.



   Uma nova tag é criada para o relatório.

   A lista de tags do relatório é exibida na seção Tags no painel. Se você quiser remover uma tag do relatório, selecione Remover ao lado da tag.

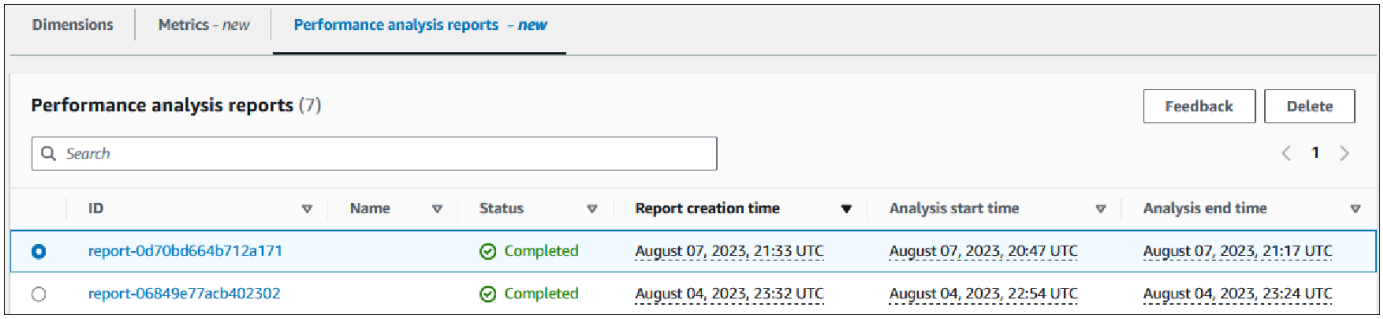
Excluir um relatório de análise de performance no Insights de Performance

Você pode excluir um relatório da lista de relatórios exibida na guia Relatórios de análise de performance ou ao visualizar um relatório.

Para excluir um relatório

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.

   O painel do Insights de Performance é exibido para a instância de banco de dados.
4. Role para baixo e selecione a guia Relatórios de análises de performance: novo.
5. Selecione o relatório que você deseja excluir e escolha Excluir no canto superior direito.



   Uma janela de confirmação é exibida. O relatório é excluído depois que você escolhe confirmar.
6. (Opcional) Selecione ID do relatório que você deseja excluir.

   Na página de relatório, selecione Excluir no canto superior direito.

   Uma janela de confirmação é exibida. O relatório é excluído depois que você escolhe confirmar.

Analisar consultas com a guia SQL principal no Insights de Performance

No painel do Amazon RDS Performance Insights, é possível encontrar informações sobre consultas recentes em execução na guia Top SQL (SQL principal), na tabela Top dimensions (Principais dimensões). É possível utilizar essas informações para ajustar suas consultas.

Tópicos

- Visão geral da guia Top SQL (SQL principal)
- Acessar mais texto SQL no painel do Performance Insights.
- Visualizar estatísticas SQL no painel do Performance Insights

Visão geral da guia Top SQL (SQL principal)

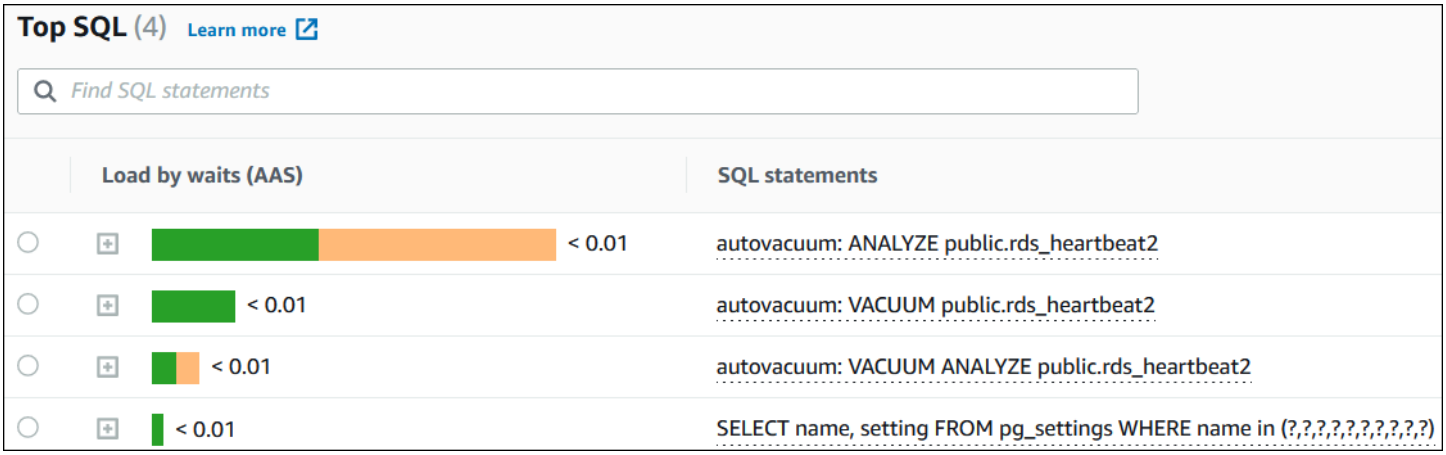
Por padrão, a guia Top SQL (SQL principal) mostra as 25 consultas que mais estão contribuindo para a carga do banco de dados. Para ajudar a ajustar as consultas, é possível analisar informações como o texto da consulta e as estatísticas de SQL. Você também pode escolher as estatísticas que deseja tornar visíveis na guia Top SQL (SQL principal).

Tópicos

- Texto SQL
- Estatísticas SQL
- Load by waits (AAS) (Carga por esperas)
- Visualizar informações do SQL
- Escolher preferências de estatísticas

Texto SQL

Por padrão, cada linha na tabela Top SQL (SQL principal) mostra 500 bytes de texto para cada declaração.



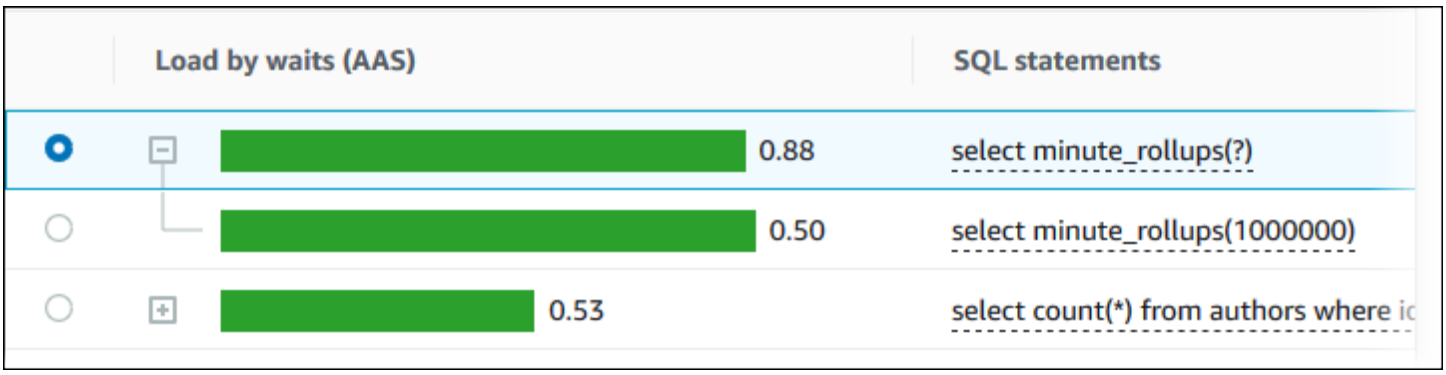
Para saber como ver mais do que os 500 bytes padrão de texto SQL, consulte Acessar mais texto SQL no painel do Performance Insights..

Um resumo SQL é formado por várias consultas reais com estruturas semelhantes, mas que possivelmente apresentam valores literais diferentes. O resumo substitui valores codificados por um ponto de interrogação. Por exemplo, um resumo pode ser `SELECT * FROM emp WHERE lname= ?`. Esse resumo pode incluir as seguintes consultas subordinadas:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
```

```
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Para ver as instruções SQL literais em um resumo, escolha a consulta e depois o sinal de mais (+). No exemplo a seguir, a consulta selecionada é de resumo.

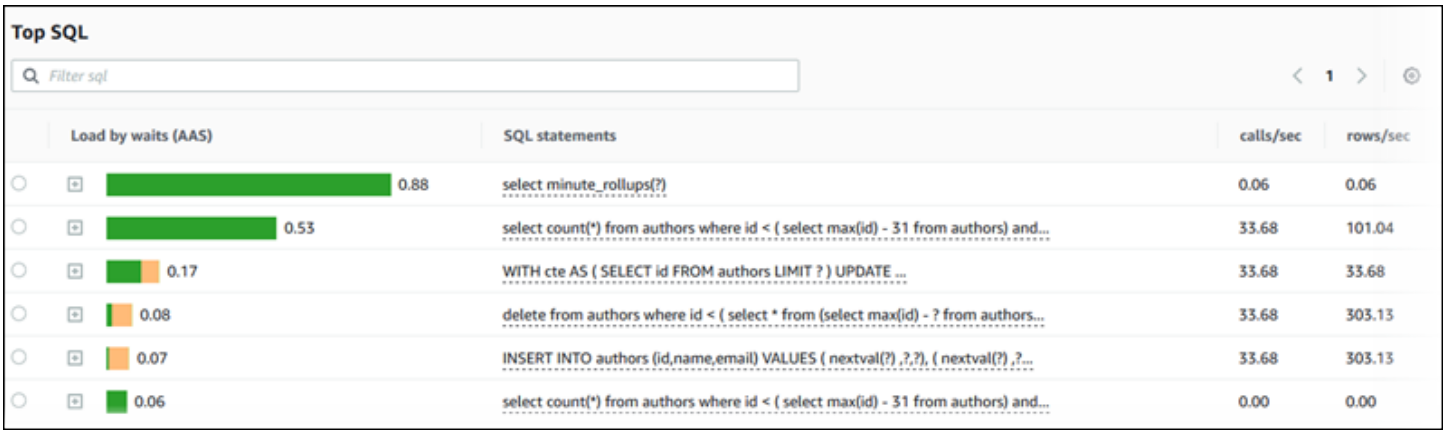


> **Note**
>
> Um resumo SQL agrupa instruções SQL semelhantes, mas não edita informações confidenciais.

Estatísticas SQL

Estatísticas SQL são métricas de performance sobre consultas SQL. Por exemplo, o Performance Insights pode exibir execuções por segundo ou linhas processadas por segundo. O Performance Insights apenas coleta estatísticas para as consultas mais comuns. Normalmente, elas correspondem às principais consultas por carga mostradas no painel do Performance Insights.

Todas as linhas da tabela Top SQL (SQL principal) mostram estatísticas relevantes para a instrução ou o resumo SQL, como mostra exemplo a seguir.



O Performance Insights pode relatar `0.00` e `-` (unknown) (desconhecido) para estatísticas SQL. Essa situação ocorre nas seguintes condições:

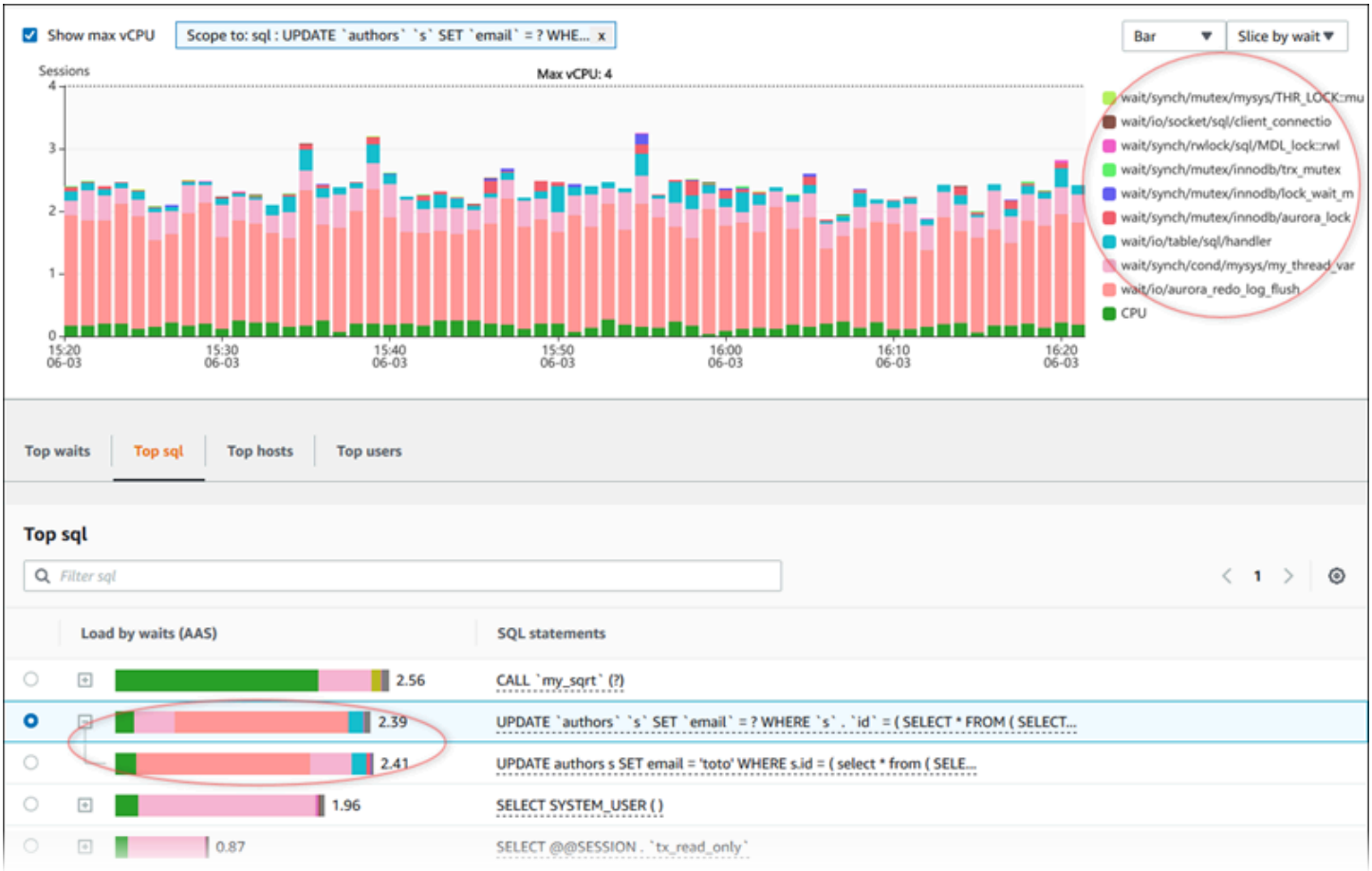
- Existe apenas uma amostra. Por exemplo, o Performance Insights calcula as taxas de alteração para consultas do Aurora PostgreSQL com base em várias amostras da visualização `pg_stat_statements`. Quando uma workload é executada por um curto período, o Performance Insights consegue coletar apenas uma amostra, o que significa que ele não consegue calcular uma taxa de alteração. O valor desconhecido é representado com um traço (`-`).
- Duas amostras têm os mesmos valores. O Performance Insights não consegue calcular uma taxa de alteração porque não ocorreu nenhuma alteração, portanto, ele relata a taxa como `0.00`.
- Uma instrução do Aurora PostgreSQL não tem um identificador válido. O PostgreSQL cria um identificador para uma instrução somente após a análise. Assim, pode existir uma instrução nas estruturas internas na memória do PostgreSQL sem identificador. Como o Performance Insights cria amostras de estruturas internas na memória uma vez por segundo, consultas de baixa latência podem aparecer apenas para uma única amostra. Se o identificador de consultas não estiver disponível para essa amostra, o Performance Insights não conseguirá associar essa instrução às suas estatísticas. O valor desconhecido é representado com um traço (`-`).

Para obter uma descrição das estatísticas SQL para o mecanismos do Aurora, consulte Estatísticas SQL para Performance Insights.

Load by waits (AAS) (Carga por esperas)

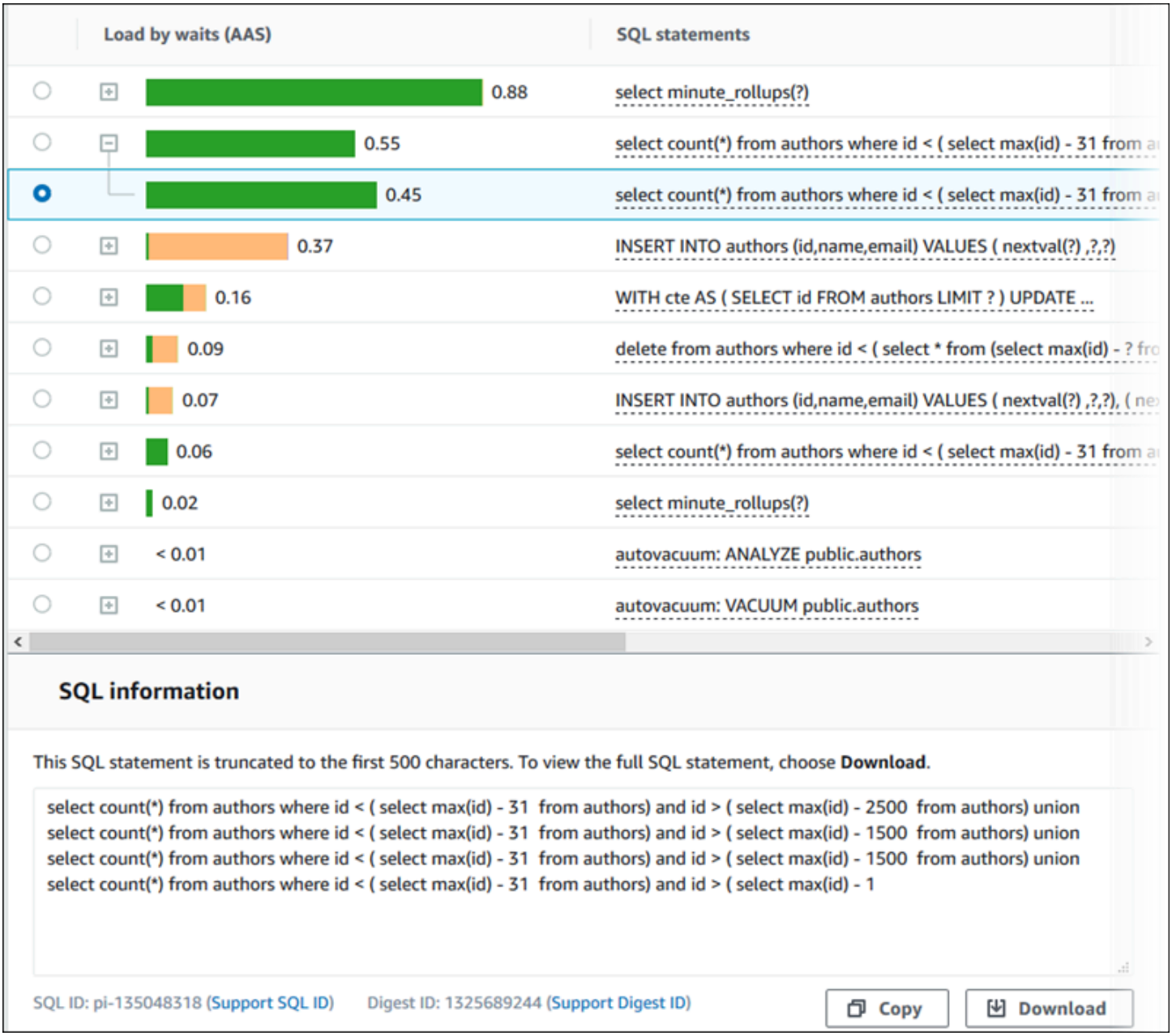
Em Top SQL (SQL principal), a coluna Load by waits (AAS) (Carga por esperas) mostra a porcentagem da carga do banco de dados associada a cada item de carga principal. Essa coluna reflete a carga desse item por qualquer agrupamento atualmente selecionado no Gráfico de carga de banco de dados. Para obter mais informações sobre média de sessões ativas (AAS), consulte Média de sessões ativas.

Por exemplo, é possível agrupar o gráfico DB load (Carga do banco de dados) com base em estados de espera. Examine consultas SQL na tabela de itens de carga principal. Nesse caso, a barra DB Load by Waits (Carga de banco de dados por espera) é dimensionada, segmentada e codificada por cores para mostrar com quanto de um determinado estado de espera a consulta está contribuindo. Ela também mostra quais estados de espera estão afetando a consulta selecionada.



Visualizar informações do SQL

Na tabela Top SQL (SQL principal), é possível abrir uma instrução para visualizar suas informações. As informações são exibidas no painel inferior.

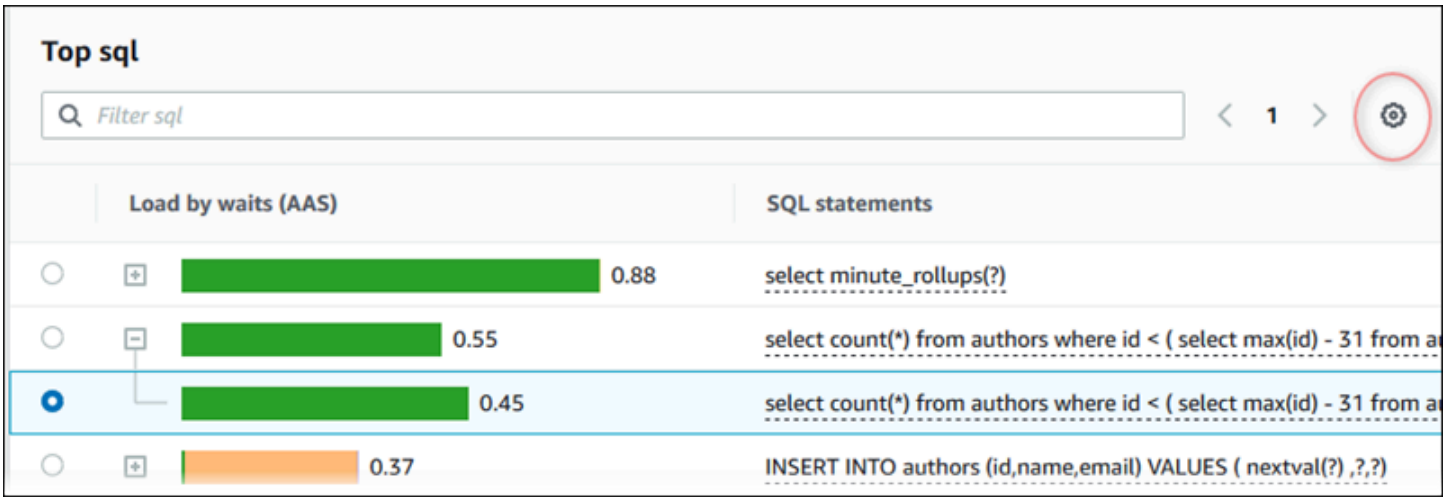


Os seguintes tipos de identificadores (IDs) associados a instruções SQL:

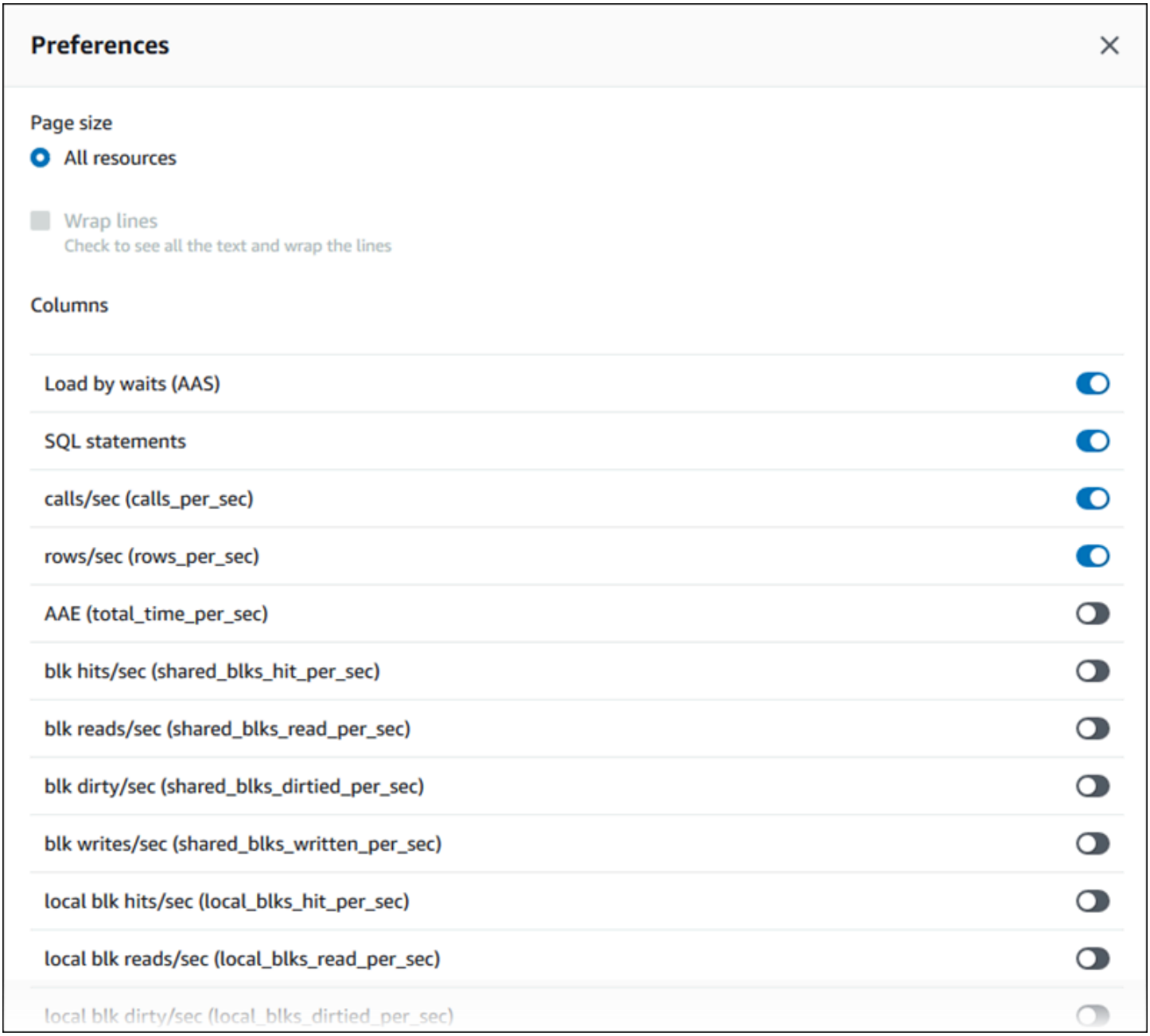
- ID SQL de suporte: Um valor de hash do ID SQL. Esse valor só se destina a referenciar um ID SQL quando você está trabalhando com o AWS Support. AWS O Support não tem acesso a IDs SQL reais e ao texto SQL.
- ID de arquivo de resumo de suporte: um valor de hash do ID de arquivo de resumo. Esse valor apenas se destina como referência a um ID de arquivo de resumo quando você está trabalhando com o AWS Support. AWS O Support não tem acesso a IDs de arquivo de resumo reais e ao texto SQL.

Escolher preferências de estatísticas

É possível controlar as estatísticas exibidas na guia Top SQL (SQL principal) selecionando o ícone Preferences (Preferências).



Quando você escolhe o ícone Preferences (Preferências), a janela Preferences (Preferências) abre. O screenshot a seguir é um exemplo da janela Preferences (Preferências).



Para habilitar as estatísticas que você deseja tornar visíveis na guia Top SQL (SQL principal), use o mouse para rolar até o final da janela e depois escolha Continue (Continuar).

Para obter mais informações sobre estatísticas por segundo ou por chamada para os mecanismos do Aurora, consulte a seção de estatísticas SQL específicas do mecanismo em[Estatísticas SQL para Performance Insights]

Acessar mais texto SQL no painel do Performance Insights.

Por padrão, cada linha na tabela Top SQL (SQL principal) mostra 500 bytes de texto SQL para cada instrução SQL.

Quando uma instrução SQL excede 500 bytes, você pode visualizar mais texto na seção SQL text (Texto SQL) abaixo da tabela Top SQL (SQL principal). Nesse caso, o comprimento máximo para o texto exibido em SQL text (Texto SQL) é de 4 KB. Esse limite é introduzido pelo console e está sujeito aos limites definidos pelo mecanismo de banco de dados. Para salvar o texto mostrado em SQL text (Texto SQL), escolha Download.

Tópicos

- Limites de tamanho de texto para Aurora MySQL
- Definir o limite do texto SQL para instâncias de banco de dados do Aurora PostgreSQL
- Visualizar e baixar texto SQL no painel do Performance Insights

Limites de tamanho de texto para Aurora MySQL

Quando você baixa um texto SQL, o mecanismo de banco de dados determina o tamanho máximo dele. Você pode baixar texto SQL até os seguintes limites por mecanismo.

Mecanismo de banco de dados	Tamanho máximo do texto para download
Aurora MySQL	O tamanho é fixado em 4.096 bytes.

A seção SQL text (Texto SQL) do console do Performance Insights mostra até o máximo que o mecanismo retorna. Por exemplo, se o Aurora MySQL retorna no máximo 1 KB para o Performance Insights, ele só pode coletar e mostrar 1 KB, mesmo que a consulta original seja maior. Assim, quando você visualiza ou baixa a consulta em SQL text (Texto SQL), o Performance Insights retorna o mesmo número de bytes.

Se você usar a AWS CLI ou a API, o Insights de Performance não terá o limite de 4 KB aplicado pelo console. `DescribeDimensionKeys` e `GetResourceMetrics` exibem no máximo 500 bytes.

> **Note**
>
> `GetDimensionKeyDetails` exibe a consulta completa, mas o tamanho está sujeito ao limite do mecanismo.

Definir o limite do texto SQL para instâncias de banco de dados do Aurora PostgreSQL

Aurora PostgreSQL lida com texto de forma diferente. Você pode definir o limite de tamanho de texto com o parâmetro de instância de banco de dados `track_activity_query_size`. Esse parâmetro tem as seguintes características:

Tamanho de texto padrão

No Aurora PostgreSQL versão 9.6, a configuração padrão do parâmetro `track_activity_query_size` é 1.024 bytes. No Aurora PostgreSQL versão 10 ou superior, o padrão é 4.096 bytes.

Tamanho máximo do texto

O limite para `track_activity_query_size` é de 102.400 bytes para Aurora PostgreSQL versão 12 e inferiores. O máximo é de 1 MB para a versão 13 e posterior.

Se o mecanismo retornar 1 MB ao Performance Insights, o console exibirá somente os primeiros 4 KB. Se você baixar a consulta, você obtém o total de 1 MB. Nesse caso, a visualização e o download retornam números diferentes de bytes. Para obter mais informações sobre o parâmetro de instância de banco de dados `track_activity_query_size`, consulte Estatísticas de tempo de execução na documentação do PostgreSQL.

Para aumentar o tamanho do texto SQL, aumente o limite `track_activity_query_size`. Para modificar o parâmetro, altere a configuração dele no grupo de parâmetros que está associado à instância de banco de dados do Aurora PostgreSQL.

Para alterar a configuração quando a instância usa o grupo de parâmetros padrão

1. Crie um novo grupo de parâmetros de instância de banco de dados para o mecanismo de banco de dados apropriado e sua respectiva versão.
2. Defina o parâmetro no novo grupo de parâmetros.
3. Associe o novo grupo de parâmetros à instância de banco de dados.

Para obter informações sobre como configurar um parâmetro de instância de banco de dados, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

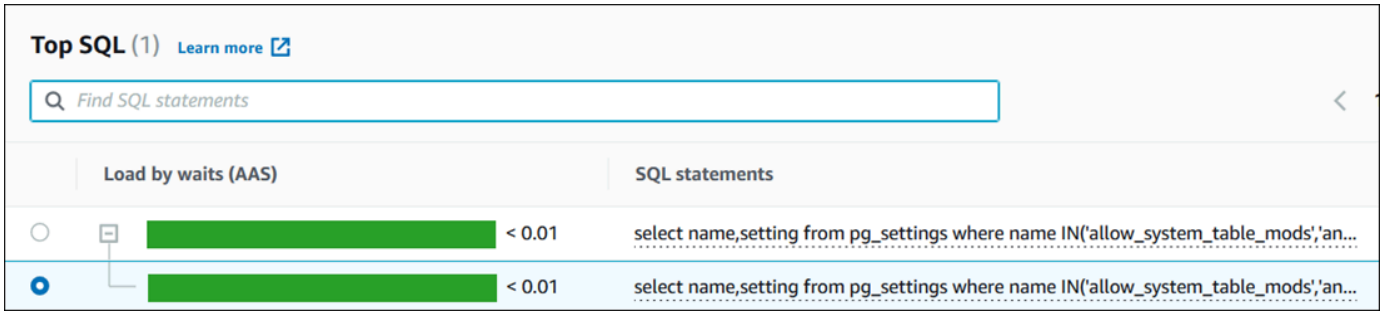
Visualizar e baixar texto SQL no painel do Performance Insights

No painel do Performance Insights, é possível visualizar ou baixar o texto SQL.

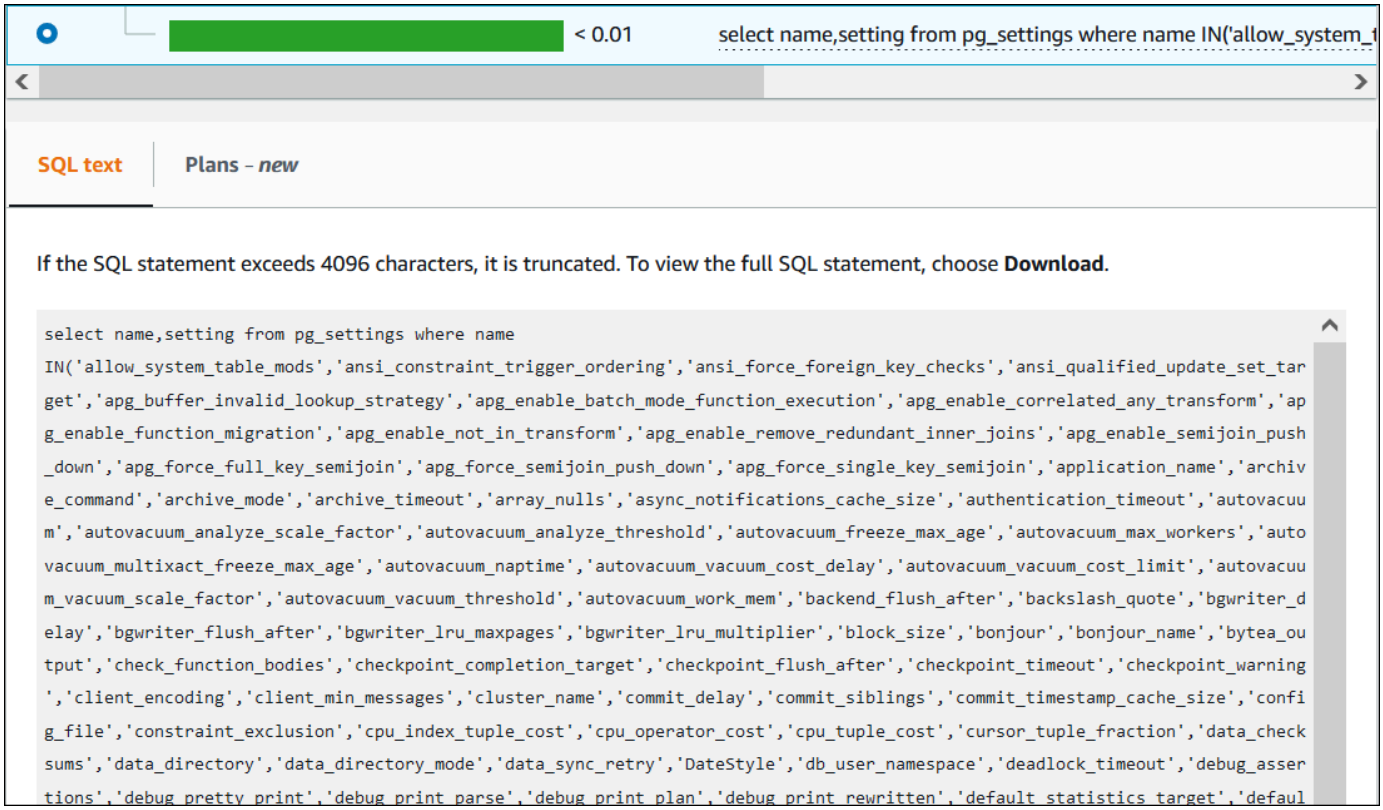
Para visualizar mais texto SQL no painel do Performance Insights

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados.
4. Role para baixo até a guia SQL principal no painel do Insights de Performance.
5. Escolha o sinal de adição para expandir um resumo SQL e selecione uma das consultas secundárias do resumo.

   Instruções SQL com texto maior que 500 bytes são semelhantes à imagem a seguir.



6. Role para baixo até a guia SQL text (texto SQL).



O painel do Performance Insights pode exibir até 4.096 bytes para cada instrução SQL.

7. (Opcional) Escolha Copy (Copiar) para copiar a instrução SQL exibida ou escolha Download (Baixar) para baixar a instrução SQL a fim de visualizar o texto SQL até o limite do mecanismo de banco de dados.

> **Note**
>
> Para copiar ou baixar a instrução SQL, desabilite bloqueadores de pop-up.

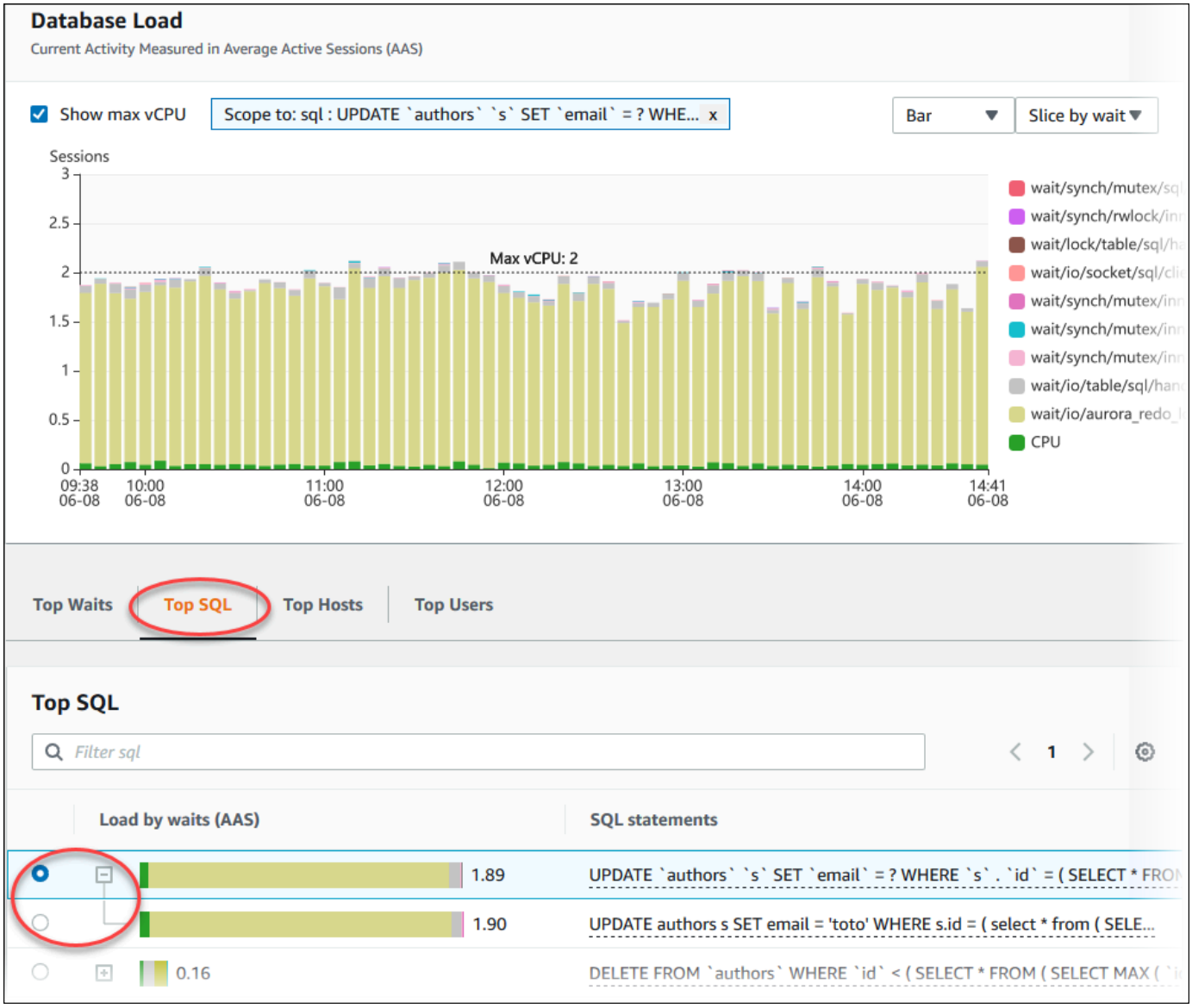
Visualizar estatísticas SQL no painel do Performance Insights

No painel do Performance Insights, as estatísticas SQL estão disponíveis na guia Top SQL (SQL principal) do grafo Database load (Carga do banco de dados).

Como visualizar estatísticas SQL

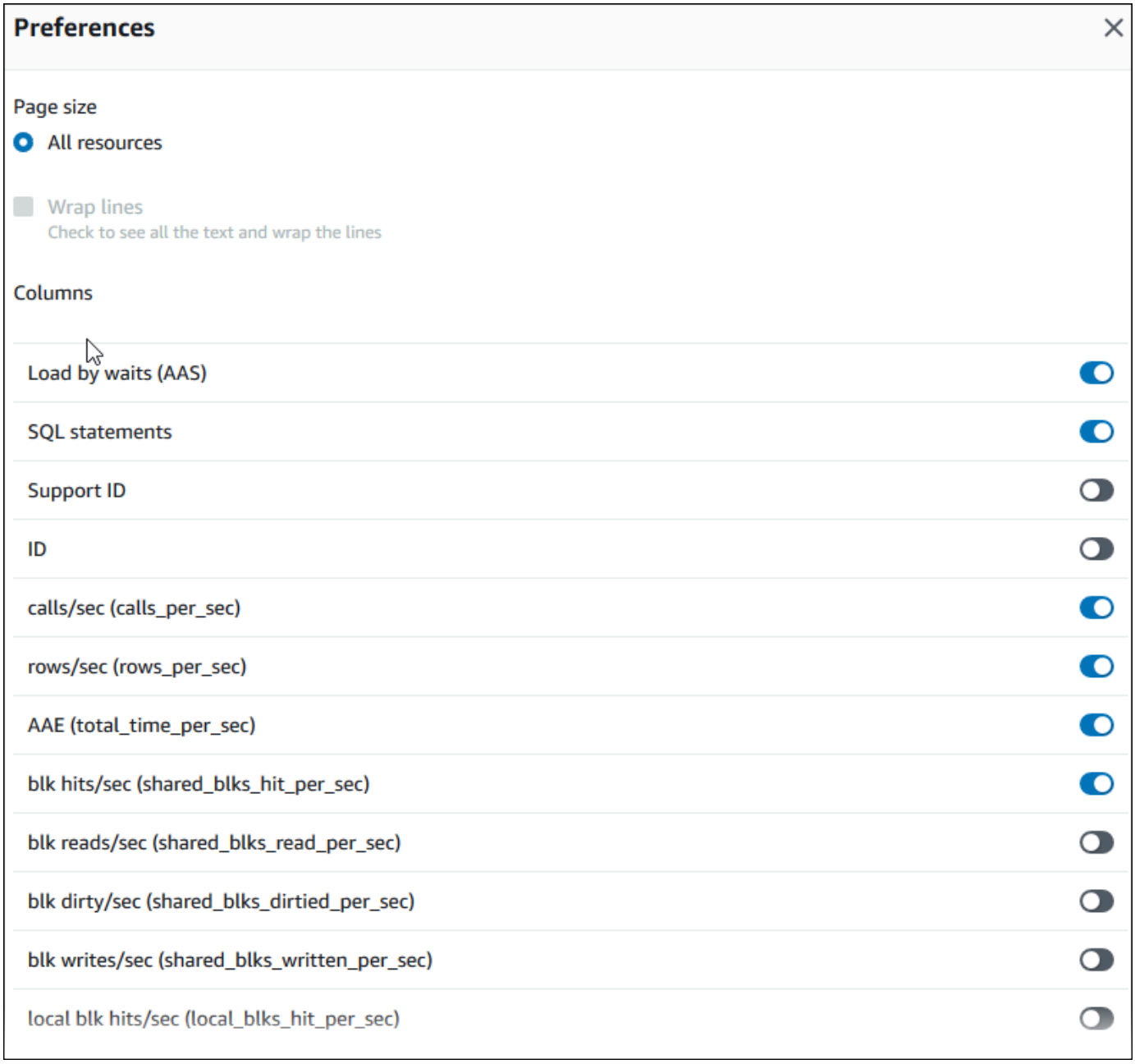
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.

3. Na parte posterior da página, escolha o banco de dados cujas estatísticas SQL você deseja ver.
4. Navegue até o final da página e escolha a guia Top SQL (SQL principal).
5. Escolha uma instrução individual (Somente Aurora MySQL)ou consulta de resumo.

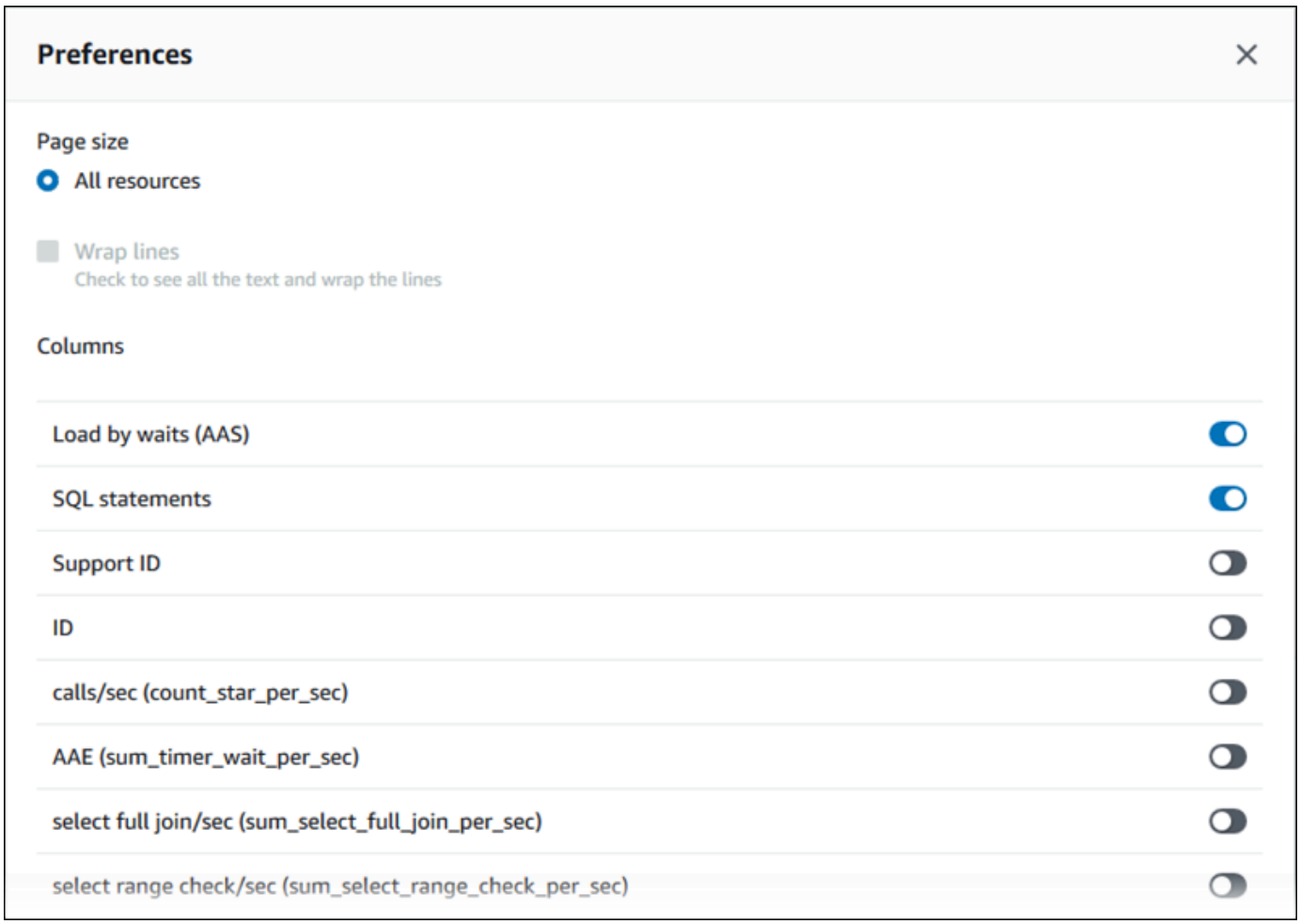


6. Escolha quais estatísticas exibir, escolhendo o ícone de engrenagem no canto superior direito do gráfico. Para obter as descrições das estatísticas SQL para o mecanismos do Amazon RDSAurora, consulte Estatísticas SQL para Performance Insights.

   O exemplo a seguir mostra as preferências do Aurora PostgreSQL.



O exemplo a seguir mostra as preferências para instâncias de banco de dados do Aurora MySQL.



7. Escolha Save (Salvar) para salvar suas preferências.

   A tabela Top SQL (SQL principal) é atualizada.

Visualizar as recomendações proativas do Performance Insights

O Insights de Performance do Amazon RDS monitora métricas específicas e cria limites automaticamente analisando quais níveis podem ser problemáticos para um recurso específico. Quando os novos valores de métricas ultrapassam um limite predefinido em um período específico, o Performance Insights gera uma recomendação proativa. Essa recomendação ajuda a evitar um impacto futuro na performance do banco de dados. Para receber essas recomendações proativas, é necessário ativar o Performance Insights com um período de retenção de nível pago.

Para obter mais informações sobre como ativar o Insights de Performance, consulte Ativar e desativar o Insights de Performance para o Aurora. Para ter informações sobre preços e retenção

de dados para o Performance Insights, consulte Preços e retenção de dados para o Performance Insights.

Para descobrir as regiões, os mecanismos de banco de dados e as classes de instância compatíveis com as recomendações proativas, consulte Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance.

É possível visualizar a análise detalhada e as investigações recomendadas de recomendações proativas na página de detalhes da recomendação.

Para obter mais informações e recomendações, consulte Recomendações do Amazon Aurora.

Como visualizar a análise detalhada de uma recomendação proativa

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, execute qualquer uma das seguintes opções:
   - Selecione Recomendações.

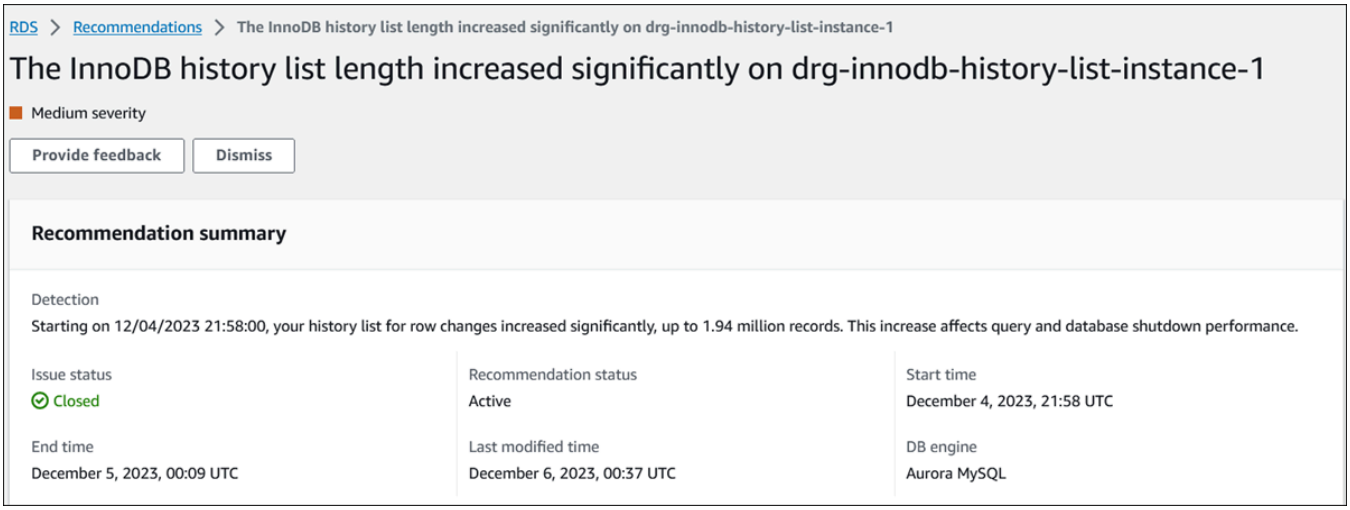
     A página Recomendações exibe uma lista de recomendações classificadas pela gravidade de todos os recursos da conta.
   - Selecione Bancos de dados e, depois, escolha Recomendações para um recurso na página de bancos de dados.

     A guia Recomendações exibe as recomendações e os detalhes do recurso selecionado.
3. Encontre uma recomendação proativa e escolha Visualizar detalhes.

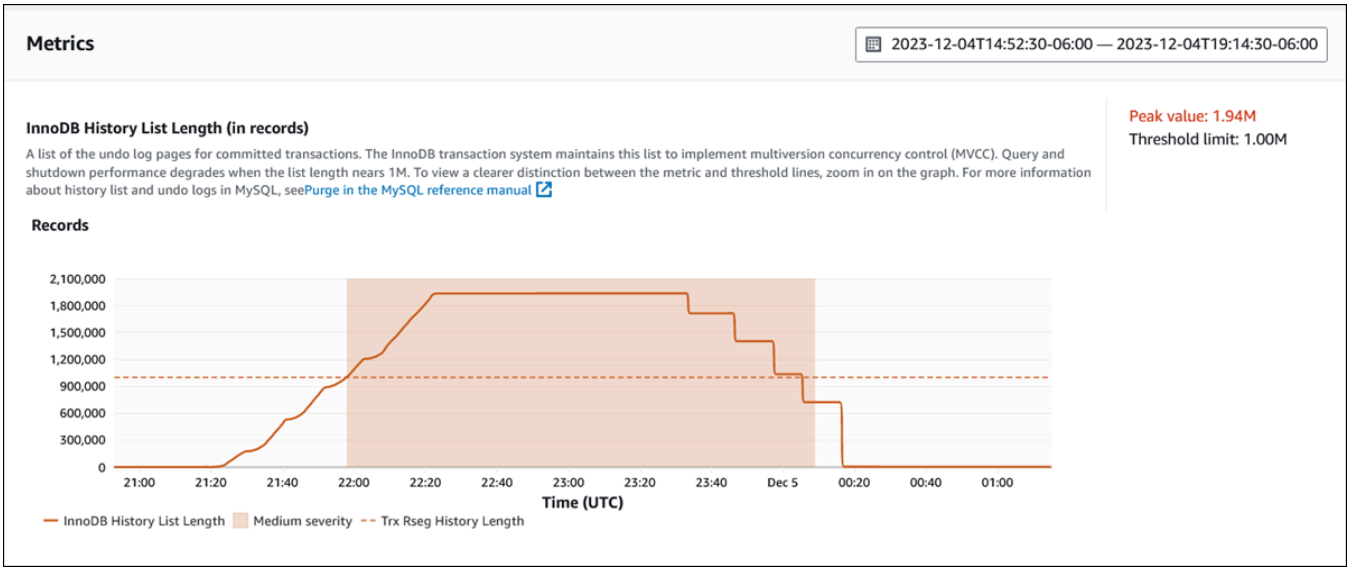
   A página de detalhes da recomendação é exibida. O título fornece o nome do recurso afetado com o problema detectado e a gravidade.

   Veja a seguir os componentes na página de detalhes da recomendação:

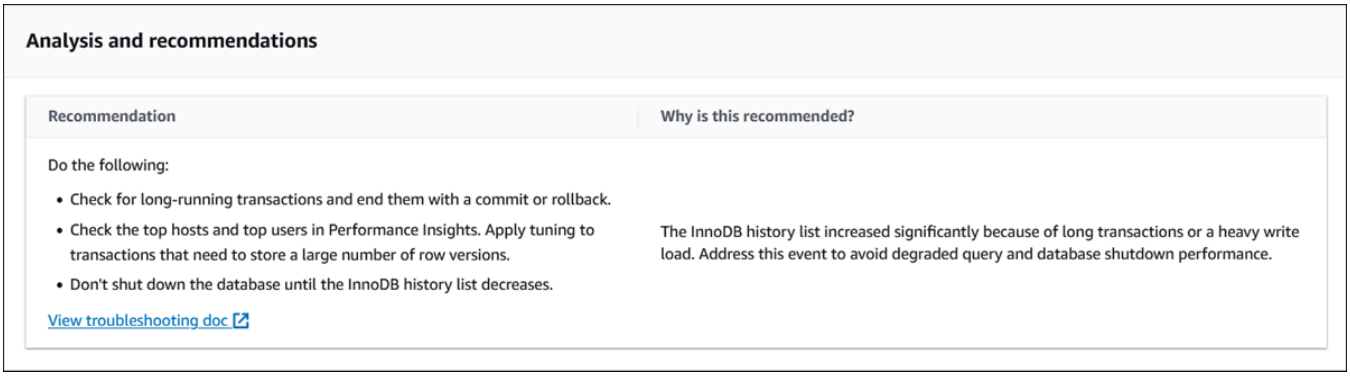
   - Resumo da recomendação: o problema detectado, a recomendação e o status do problema, a hora de início e término do problema, a hora de modificação da recomendação e o tipo de mecanismo.



- Métricas: os grafos do problema detectado. Cada grafo exibe um limite determinado pelo comportamento básico do recurso, bem como dados da métrica relatados a partir da hora de início do problema.



- Análise e recomendações: a recomendação e o motivo da recomendação sugerida.



É possível analisar a causa do problema e, depois, executar as ações recomendadas sugeridas para corrigir o problema ou escolher Dispensar no canto superior direito para dispensar a recomendação.

Recuperar métricas com a API do Insights de Performance para o Aurora

Quando o Insights de Performance está ativado, a API oferece visibilidade à performance da instância. O Amazon CloudWatch Logs fornece a fonte de autorização para métricas de monitoramento fornecidas para serviços da AWS.

O Performance Insights oferece uma visão específica do domínio da carga do banco de dados medida como sessões ativas médias (AAS). Essa métrica aparece para os consumidores de API como um conjunto de dados bidimensional de séries temporais. A dimensão de tempo dos dados fornece a carga do banco de dados para cada ponto de tempo no intervalo de tempo consultado. Cada ponto de tempo decompõe a carga geral em relação às dimensões solicitadas, como `SQL`, `Wait-event`, `User` ou `Host`, medidas naquele ponto de tempo.

O Amazon RDS Performance Insights monitora o cluster Amazon Aurora do , para que você possa analisar e solucionar problemas relacionados à performance do banco de dados. Uma maneira de visualizar os dados do Performance Insights está no Console de gerenciamento da AWS. O Performance Insights também fornece uma API pública para que você possa consultar seus próprios dados. É possível usar a API para fazer o seguinte:

- Descarregar dados em um banco de dados
- Adicione dados do Performance Insights aos painéis de monitoramento existentes
- Criar ferramentas de monitoramento

Para usar a API do Performance Insights, habilite o Performance Insights em uma das suas instâncias de banco de dados do Amazon RDS. Para obter informações sobre como habilitar o Performance Insights, consulte Ativar e desativar o Insights de Performance para o Aurora. Para obter mais informações sobre a API do Performance Insights, consulte a Referência de API do Amazon RDS Performance Insights.

A API do Performance Insights fornece as operações a seguir.

Ação do Performance Insights	AWS CLI command	Descrição
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	Cria um relatório de análise de performance referente a um período específico para a instância de banco de dados. O resultado é `AnalysisReportId`, que é o identificador exclusivo do relatório.
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	Exclui um relatório de análise de performance.
DescribeDimensionKeys	aws pi describe-dimension-keys	Recuperar as N principais chaves de dimensão de uma métrica por um período específico.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Recupera os atributos do grupo de dimensões especificado para uma instância de banco de dados ou fonte de dados. Por exemplo, se você especificar um ID SQL e se os detalhes da dimensão estiverem disponíveis, `GetDimensionKeyDetails` recuperará o texto completo da dimensão `db.sql.statement` associada a esse ID. Essa operação é útil porque `GetResourceMetrics` e `DescribeDimensionKeys` não oferecem suporte à

Ação do Performance Insights	AWS CLI command	Descrição
		recuperação de texto grande de instrução SQL.
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	Recupera o relatório, incluindo os insights do relatório. O resultado inclui o status do relatório, o ID do relatório , os detalhes do horário do relatório, os insights e as recomendações.
GetResourceMetadata	aws pi get-resource-metadata	Recupere os metadados para diferentes recursos. Por exemplo, os metadados podem indicar que um recurso está ativado ou desativado em uma instância de banco de dados específica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera as métricas do Performance Insights para um conjunto de fontes de dados, ao longo de um período. É possível fornecer grupos de dimensão e dimensões específicos e fornecer critérios de filtragem e agregação para cada grupo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupere as dimensões que podem ser consultadas para cada tipo de métrica especificado em uma instância especificada.

Ação do Performance Insights	AWS CLI command	Descrição
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupere todas as métricas disponíveis dos tipos de métrica especificados que podem ser consultados para uma instância de banco de dados especificada.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	Recupera todos os relatórios de análise disponíveis para a instância de banco de dados. Os relatórios são listados com base na hora de início de cada relatório.
ListTagsForResource	aws pi list-tags-for-resource	Lista todas as tags de metadados adicionadas ao recurso. A lista inclui o nome e o valor da tag.
TagResource	aws pi tag-resource	Adiciona tags de metadados ao recurso do Amazon RDS. A tag inclui um nome e um valor.
UntagResource	aws pi untag-resource	Remove a tag de metadados do recurso.

Para ter mais informações sobre como recuperar métricas de séries temporais e exemplos da AWS CLI para o Insights de Performance, consulte os tópicos a seguir.

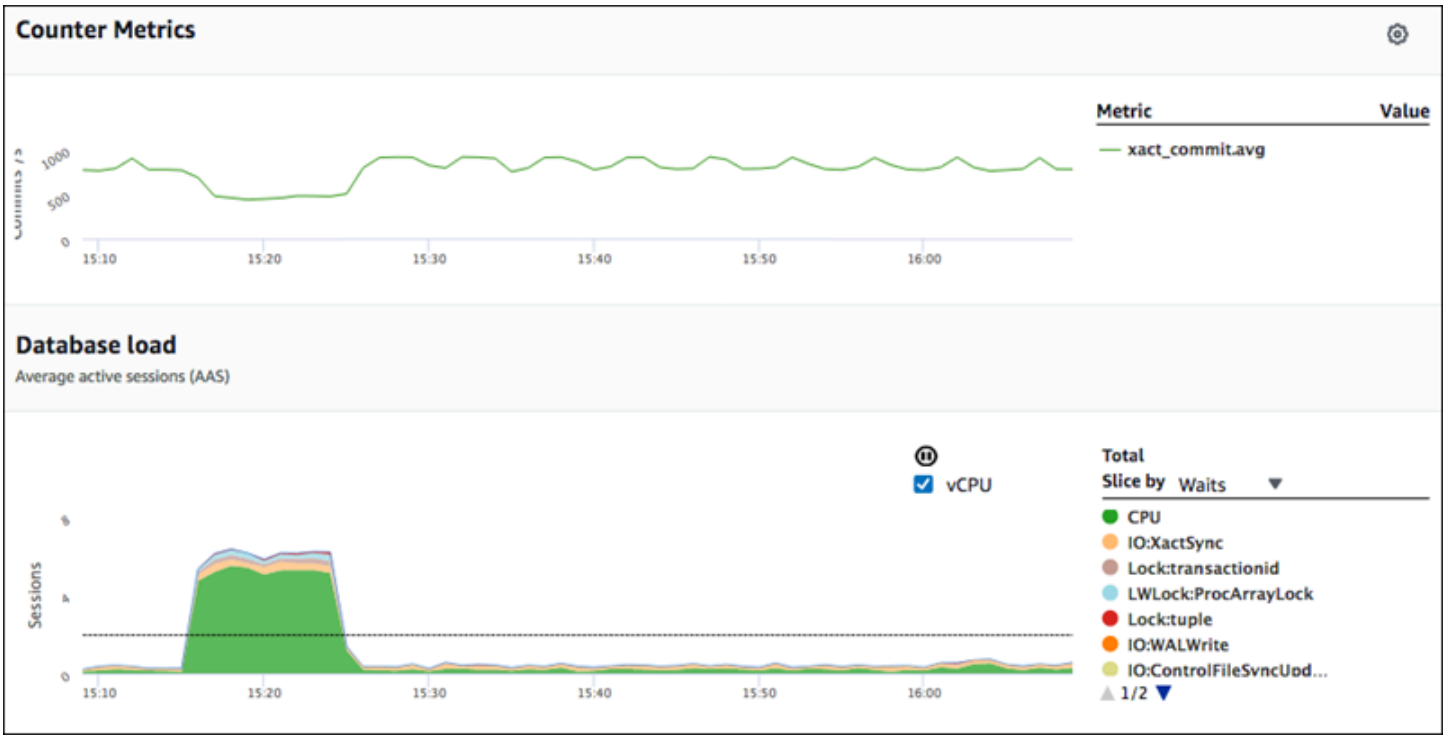
Tópicos

- Recuperar métricas de séries temporais do Insights de Performance
- Exemplos da AWS CLI para o Performance Insights

Recuperar métricas de séries temporais do Insights de Performance

A operação `GetResourceMetrics` recupera uma ou mais métricas de séries temporais dos dados do Performance Insights. `GetResourceMetrics` requer uma métrica e um período de tempo e retorna uma resposta com uma lista de pontos de dados.

Por exemplo, o Console de gerenciamento da AWS usa `GetResourceMetrics` para preencher o gráfico Métricas de contador e o gráfico Carregamento do banco de dados, como visto na imagem a seguir.



Todas as métricas retornadas por `GetResourceMetrics` são métricas de séries temporais padrão, com exceção de `db.load`. Essa métrica é exibida no gráfico Carregamento do banco de dados. A métrica `db.load` é diferente das outras métricas da série temporal, pois você pode fragmentá-la em subcomponentes chamados de dimensões. Na imagem anterior, `db.load` é dividido e agrupado pelos estados de espera que compõem o `db.load`.

> **Note**
>
> `GetResourceMetrics` também pode retornar a métrica `db.sampleload`, mas a métrica `db.load` é apropriada na maioria dos casos.

Para obter informações sobre as métricas de contador retornadas pelo `GetResourceMetrics`, consulte Métricas de contadores do Performance Insights.

Os cálculos a seguir são compatíveis com as métricas:

- Média: o valor médio para a métrica por um período. Adicione `.avg` ao nome da métrica.
- Mínimo: o valor mínimo para a métrica por um período. Adicione `.min` ao nome da métrica.
- Máximo: o valor máximo para a métrica por um período. Adicione `.max` ao nome da métrica.
- Soma: a soma dos valores da métrica por um período. Adicione `.sum` ao nome da métrica.
- Contagem de amostra: o número de vezes que a métrica foi coletada por um período. Adicione `.sample_count` ao nome da métrica.

Por exemplo, considere que uma métrica é coletada por 300 segundos (5 minutos) e que a métrica seja coletada uma vez por minuto. Os valores de cada minuto são 1, 2, 3, 4 e 5. Nesse caso, os seguintes cálculos são retornados:

- Média: 3
- Mínimo: 1
- Máximo: 5
- Soma: 15
- Contagem de amostras: 5

Para obter informações sobre como usar o comando `get-resource-metrics` AWS CLI, consulte `get-resource-metrics`.

Para a opção `--metric-queries`, especifique uma ou mais consultas para as quais deseja obter resultados. Cada consulta consiste em um parâmetro obrigatório `Metric` e opcional `GroupBy` e em parâmetros `Filter`. Veja a seguir um exemplo de uma especificação de opção `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
```

```
  },
  "Filter": {"string": "string"
    ...}
```

Exemplos da AWS CLI para o Performance Insights

Nas seções a seguir, saiba mais sobre a AWS Command Line Interface (AWS CLI) para o Insights de Performance e use exemplos da AWS CLI.

Tópicos

- Ajuda integrada da AWS CLI para o Insights de Performance
- Recuperar métricas de contador
- Recuperar a média de carga de banco de dados para eventos de espera superior
- Recuperar a média de carga de banco de dados para SQL principal
- Recuperação da média de carga de banco de dados filtrada por SQL
- Recuperar o texto completo de uma instrução SQL
- Criar um relatório de análise de performance para um período
- Recuperar um relatório de análise de performance
- Listar todos os relatórios de análise de performance da instância de banco de dados
- Excluir um relatório de análise de performance
- Adicionar tags a um relatório de análise de performance
- Listar todas as tags para um relatório de análise de performance
- Excluir tags de um relatório de análise de performance

Ajuda integrada da AWS CLI para o Insights de Performance

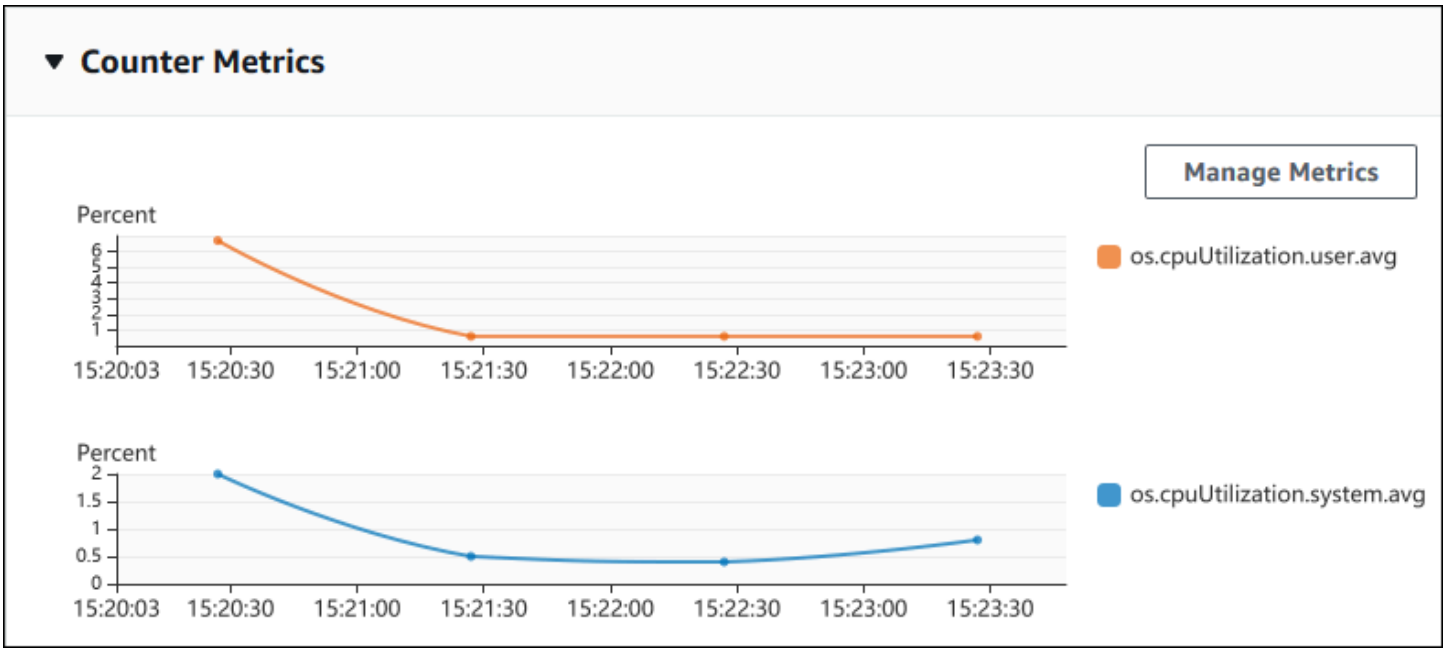
É possível visualizar dados do Performance Insights usando o AWS CLI. É possível visualizar a ajuda dos comandos da AWS CLI para o Performance Insights, inserindo o seguinte na linha de comando.

```
aws pi help
```

Se você não tiver a AWS CLI instalada, consulte informações sobre como instalá-la em Instalar a AWS CLI no Guia do usuário da AWS CLI.

Recuperar métricas de contador

A captura de tela a seguir mostra dois gráficos de métricas de contador no Console de gerenciamento da AWS.



O exemplo a seguir mostra como reunir os mesmos dados que o Console de gerenciamento da AWS usa para gerar os dois gráficos de métricas de contador.

Para Linux, macOS ou Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
```

```
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Você também pode tornar um comando mais fácil de ler, especificando um arquivo para a opção `--metrics-query`. O exemplo a seguir usa um arquivo chamado query.json para a opção. O arquivo tem o seguinte conteúdo.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Execute o seguinte comando para usar o arquivo.

Para Linux, macOS ou Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

O exemplo anterior especifica os seguintes valores para as opções:

- `--service-type`: `RDS` para Amazon RDS
- `--identifier` – O ID do recurso para a instância do banco de dados
- `--start-time` e `--end-time` – Os valores ISO 8601 de `DateTime` para o período a consultar, com vários formatos compatíveis

Ele consulta um intervalo de tempo de uma hora:

- `--period-in-seconds` – `60` para uma consulta por minuto
- `--metric-queries` – uma matriz de duas consultas, cada uma apenas para uma métrica.

  O nome da métrica usa pontos para classificar a métrica em uma categoria útil, com o elemento final sendo uma função. No exemplo, a função é `avg` para cada consulta. Como no Amazon CloudWatch, as funções com suporte são `min`, `max`, `total` e `avg`.

A resposta é semelhante à seguinte.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
```

```
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

A resposta tem `Identifier`, `AlignedStartTime` e `AlignedEndTime`. Se o valor de `--period-in-seconds` fosse `60`, as horas de início e término seriam alinhadas ao minuto. Se `--period-in-seconds` fosse `3600`, as horas de início e término teriam sido alinhadas à hora.

O `MetricList` na resposta tem um número de entradas, cada uma com uma entrada `Key` e `DataPoints`. Cada `DataPoint` tem um `Timestamp` e um `Value`. Cada lista `Datapoints` tem 60 pontos de dados, pois as consultas são para dados por minuto ao longo de uma hora, com `Timestamp1/Minute1`, `Timestamp2/Minute2` e assim por diante, até `Timestamp60/Minute60`.

Como a consulta é para duas métricas de contador diferentes, há dois elementos na resposta `MetricList`.

Recuperar a média de carga de banco de dados para eventos de espera superior

O exemplo a seguir é a mesma consulta que o Console de gerenciamento da AWS usa para gerar um gráfico de linha de área empilhada. Este exemplo recupera o `db.load.avg` para a última hora com carga dividida de acordo com os sete principais eventos de espera. O comando é o mesmo que o comando em Recuperar métricas de contador. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
]
```

Execute o comando a seguir.

Para Linux, macOS ou Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

O exemplo especifica a métrica de `db.load.avg` e um `GroupBy` dos sete principais eventos de espera. Para obter detalhes sobre valores válidos para esse exemplo, consulte DimensionGroup na Referência de API do Performance Insights.

A resposta é semelhante à seguinte.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
```

```
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
```

```
} //end of response
```

Nessa resposta, há oito entradas no `MetricList`. Há uma entrada para o total `db.load.avg`, e sete entradas cada para o `db.load.avg`, divididas de acordo com um dos sete principais eventos de espera. Ao contrário do primeiro exemplo, como havia uma dimensão de agrupamento, deve haver uma chave para cada agrupamento da métrica. Não pode haver apenas uma chave para cada métrica, como no caso de uso de métricas de contador.

Recuperar a média de carga de banco de dados para SQL principal

O exemplo a seguir agrupa `db.wait_events` pelas 10 principais instruções SQL. Existem dois grupos diferentes para instruções SQL:

- `db.sql` – a instrução SQL completa, como `select * from customers where customer_id = 123`
- `db.sql_tokenized` – a instrução SQL tokenizada, como `select * from customers where customer_id = ?`

Ao analisar a performance do banco de dados, pode ser útil considerar instruções SQL que diferem apenas por seus parâmetros como um item lógico. Então, você pode usar `db.sql_tokenized` ao consultar. No entanto, especialmente quando você está interessado em explicar planos, às vezes é mais útil examinar instruções SQL completas com parâmetros e agrupamentos de consulta por `db.sql`. Existe um relacionamento pai-filho entre o SQL tokenizado e o SQL completo, com vários SQL completos (filhos) agrupados sob o mesmo SQL (pai) tokenizado.

O comando neste exemplo é semelhante ao comando em Recuperar a média de carga de banco de dados para eventos de espera superior. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

O exemplo a seguir usa `db.sql_tokenized`.

Para Linux, macOS ou Unix:

```
aws pi get-resource-metrics \
```

```
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-29T00:00:00Z ^
   --end-time   2018-10-30T00:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

Este exemplo consulta mais de 24 horas, com um período de uma hora em segundos.

O exemplo especifica a métrica de `db.load.avg` e um `GroupBy` dos sete principais eventos de espera. Para obter detalhes sobre valores válidos para esse exemplo, consulte DimensionGroup na Referência de API do Performance Insights.

A resposta é semelhante à seguinte.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
```

```
        },
        {
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

Essa resposta tem 11 entradas no `MetricList` (1 total, 10 principais SQLs tokenizados), com cada entrada com 24 `DataPoints` por hora.

Para o SQL tokenizado, existem três entradas em cada lista de dimensões:

- `db.sql_tokenized.statement` – a instrução SQL tokenizada.
- `db.sql_tokenized.db_id` – o ID do banco de dados nativo usado para referência ao SQL ou um ID sintético que o Performance Insights gera para você quando o ID do banco de dados nativo não está disponível. Este exemplo retorna o ID sintético `pi-2372568224`.
- `db.sql_tokenized.id` – o ID da consulta dentro do Performance Insights.

  No Console de gerenciamento da AWS, esse ID é chamado de ID de suporte. Ele é chamado assim porque o ID consiste em dados que o AWS Support pode examinar para ajudar você a solucionar um problema no banco de dados. A AWS leva a segurança e privacidade de seus dados extremamente a sério, e quase todos os dados são armazenados criptografados com a chave do AWS KMS. Portanto, ninguém dentro da AWS pode examinar esses dados. No exemplo precedente, `tokenized.statement` e `tokenized.db_id` são armazenados em formato criptografado. Se você tiver um problema com o banco de dados, o Suporte da AWS poderá ajudar você consultando o ID de suporte.

Ao consultar, pode ser conveniente especificar `Group` em `GroupBy`. No entanto, para um controle mais refinado sobre os dados retornados, especifique a lista de dimensões. Por exemplo, se tudo o

que for necessário for o `db.sql_tokenized.statement`, um atributo `Dimensions` poderá ser adicionado ao arquivo query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recuperação da média de carga de banco de dados filtrada por SQL

A imagem anterior mostra que uma consulta específica está selecionada e que o gráfico de linhas da área empilhada das sessões ativas da média superior tem o escopo para essa consulta. Embora a consulta ainda seja para os sete principais eventos de espera geral, o valor da resposta é filtrado. O filtro faz com que ele leve em consideração apenas as sessões correspondentes ao filtro específico.

A consulta da API correspondente neste exemplo é semelhante ao comando em Recuperar a média de carga de banco de dados para SQL principal. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Para Linux, macOS ou Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

A resposta é semelhante à seguinte.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
```

```
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
```

```
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
```

```
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

Nessa resposta, todos os valores são filtrados de acordo com a contribuição de SQL tokenizado AKIAIOSFODNN7EXAMPLE especificado no arquivo query.json. As chaves também podem seguir uma ordem diferente de uma consulta sem um filtro, porque são os cinco principais eventos de espera que afetaram o SQL filtrado.

Recuperar o texto completo de uma instrução SQL

O exemplo a seguir recupera o texto completo de uma instrução SQL para a instância de banco de dados `db-10BCD2EFGHIJ3KL4M5NO6PQRS5`. O `--group` é `db.sql`, e o `--group-identifier` é `db.sql.id`. Nesse exemplo, *`my-sql-id`* representa um ID SQL recuperado que invoca `pi get-resource-metrics` ou `pi describe-dimension-keys`.

Execute o comando a seguir.

Para Linux, macOS ou Unix:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
```

```
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Para Windows:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

Nesse exemplo, os detalhes das dimensões estão disponíveis. Assim, o Performance Insights recupera o texto completo da instrução SQL, sem truncá-lo.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

Criar um relatório de análise de performance para um período

O exemplo a seguir cria um relatório de análise de performance com o horário de início `1682969503` e o horário de término `1682979503` do banco de dados `db-loadtest-0`.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

A resposta é o identificador exclusivo `report-0234d3ed98e28fb17` do relatório.

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

Recuperar um relatório de análise de performance

O exemplo a seguir recupera os detalhes do relatório de análise do relatório `report-0d99cc91c4422ee61`.

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

A resposta fornece o status, o ID, os detalhes do horário e os insights do relatório.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

Listar todos os relatórios de análise de performance da instância de banco de dados

O exemplo a seguir lista todos os relatórios de análise de performance disponíveis para o banco de dados `db-loadtest-0`.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
```

```
--region us-west-2
```

A resposta lista todos os relatórios com o ID do relatório, o status e os detalhes do período.

```
{
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

Excluir um relatório de análise de performance

O exemplo a seguir exclui o relatório de análise do banco de dados `db-loadtest-0`.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
```

```
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

Adicionar tags a um relatório de análise de performance

O exemplo a seguir adiciona uma tag com uma chave `name` e um valor `test-tag` ao relatório `report-01ad15f9b88bcbd56`.

```
aws pi tag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

Listar todas as tags para um relatório de análise de performance

O exemplo a seguir lista todas as tags do relatório `report-01ad15f9b88bcbd56`.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

A resposta lista o valor e a chave de todas as tags adicionadas ao relatório:

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

Excluir tags de um relatório de análise de performance

O exemplo a seguir exclui a tag `name` do relatório `report-01ad15f9b88bcbd56`.

```
aws pi untag-resource \
```

```
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tag-keys name \
--region us-west-2
```

Depois que a tag for excluída, chamar a API `list-tags-for-resource` não listará essa tag.

Registrar em log as chamadas do Performance Insights usando o AWS CloudTrail

O Performance Insights é executado com o AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um serviço da AWS no Performance Insights. O CloudTrail captura todas as chamadas de API para o Performance Insights como eventos. Essa captura inclui chamadas do console do Amazon RDS e chamadas de código para as operações de API do Performance Insights.

Se você criar uma trilha, será possível ativar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3, incluindo eventos do Performance Insights. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail no Histórico de eventos. Ao usar os dados coletados pelo CloudTrail, é possível determinar certas informações. Essas informações incluem a solicitação que foi feita ao Performance Insights, o endereço IP do qual a solicitação foi feita, quem fez a solicitação e quando ela foi feita. Também inclui detalhes adicionais.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Trabalhar com informações do Performance Insights no CloudTrail

O CloudTrail é habilitado em sua conta AWS ao criá-la. Quando ocorre uma atividade no Performance Insights, ela é registrada em um evento do CloudTrail com outros eventos de serviços da AWS no console do CloudTrail em Event history (Histórico de eventos). Você pode visualizar, pesquisar e baixar eventos recentes em sua conta da AWS. Para obter mais informações, consulte Viewing Events with CloudTrail Event History (Visualizar eventos com o histórico de eventos CloudTrail) no Guia do Usuário do AWS CloudTrail.

Para obter um registro contínuo de eventos na conta da AWS, incluindo eventos para o Performance Insights, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando uma trilha é criada no console, a mesma é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões da AWS na partição da AWS

e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros AWS serviços para melhor analisar e agir de acordo com dados coletados do evento nos logs CloudTrail. Para obter mais informações, consulte os seguintes tópicos no Guia do usuário do AWS CloudTrail:

- Visão geral da criação de uma trilha
- Serviços e integrações com suporte no CloudTrail
- Configurando Notificações Amazon SNS para CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e receber arquivos de log do CloudTrail de várias contas

Todas as operações do Performance Insights são registradas em log pelo CloudTrail e documentadas na Referência de API do Performance Insights. Por exemplo, as chamadas para as operações `DescribeDimensionKeys` e `GetResourceMetrics` geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário-raiz ou usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte o Elemento userIdentity do CloudTrail.

Entradas de arquivo de log do Performance Insights

Uma trilha é uma configuração que permite a entrega de eventos como registros de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma solicitação única de qualquer fonte. Cada evento inclui informações sobre a operação solicitada, a data e a hora da operação, os parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública. Dessa forma, eles não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a operação `GetResourceMetrics`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

API do Performance Insights e endpoints da VPC de interface (AWS PrivateLink)

É possível usar o AWS PrivateLink para criar uma conexão privada entre a VPC e o Insights de Performance do Amazon RDS. É possível acessar o Performance Insights como se estivesse em sua VPC, sem usar um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do Direct Connect. As instâncias na VPC não precisam de endereços IP públicos para acessar o Performance Insights.

Estabeleça essa conectividade privada criando um endpoint de interface, habilitado pelo AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Estas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Performance Insights.

Para obter mais informações, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink.

Considerações sobre o Performance Insights

Antes de configurar um endpoint de interface para o Performance Insights, leia Considerations no Guia do AWS PrivateLink.

O Performance Insights aceita fazer chamadas para todas as suas ações de API por meio do endpoint da interface.

Por padrão, o acesso total ao Performance Insights é permitido no endpoint da interface. Para controlar o tráfego pra o Performance Insights por meio do endpoint da interface, associe um grupo de segurança às interfaces de rede do endpoint.

Disponibilidade

Atualmente, a API do Performance Insights comporta endpoints da VPC em Regiões da AWS que aceitam o Performance Insights. Para ter informações sobre a disponibilidade do Performance Insights, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Insights de Performance.

Criar um endpoint de interface para o Performance Insights

É possível criar um endpoint de interface para o Performance Insights usando o console da Amazon VPC ou a AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário do AWS PrivateLink.

Crie um endpoint de interface para o Performance Insights usando o seguinte nome de serviço:

Se você habilitar o DNS privado para o endpoint da interface, poderá fazer solicitações de API ao Performance Insights usando seu nome do DNS regional padrão. Por exemplo, `pi.us-east-1.amazonaws.com`.

Criar uma política de endpoint da VPC para a API do Performance Insights

Uma política de endpoint é um recurso do IAM que pode ser anexado ao endpoint de interface. A política de endpoint padrão permite acesso total ao Performance Insights por meio do endpoint de interface. Para controlar o acesso permitido ao Performance Insights pela VPC anexe uma política de endpoint personalizada ao endpoint de interface.

Uma política de endpoint especifica as seguintes informações:

- As entidades principais que podem realizar ações (Contas da AWS, usuários do IAM e perfis do IAM).
- As ações que podem ser executadas.
- Os recursos nos quais as ações podem ser executadas.

Para obter mais informações, consulte Controlar o acesso aos serviços usando políticas de endpoint no Guia do AWS PrivateLink.

Exemplo: Política de endpoint da VPC para ações do Performance Insights

Veja a seguir um exemplo de uma política de endpoint personalizado. Quando anexada ao endpoint de interface, essa política concede acesso às ações do Performance Insights listadas para todas as entidades principais em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
```

```
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

Exemplo: política de endpoint da VPC que nega todo o acesso de uma conta da AWS especificada

A política de endpoint da VPC a seguir nega à conta da AWS 123456789012 todo o acesso aos recursos que usam o limite. A política permite todas as ações de outras contas.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Endereçamento IP para Performance Insights

Os endereços IP habilitam recursos na sua VPC para se comunicar com outros e com recursos na Internet. O Performance Insights comporta protocolos de endereçamento IPv4 e IPv6. Por padrão, o Performance Insights e a Amazon VPC usam o protocolo de endereçamento IPv4. Você não pode desativar esse comportamento. Ao criar uma VPC, especifique um bloco CIDR IPv4 (um intervalo de endereços IPv4 privados).

Também é possível atribuir um bloco CIDR IPv6 à VPC e às sub-redes e atribuir endereços IPv6 desse bloco a recursos do RDS na sub-rede. A compatibilidade com o protocolo IPv6 expande

o número de endereços IP compatíveis. Ao usar o protocolo IPv6, você tem a garantia de que terá endereços disponíveis suficientes para o crescimento futuro da Internet. Os recursos do RDS novos e existentes podem usar endereços IPv4 e IPv6 na VPC. Configurar, proteger e converter o tráfego de rede entre os dois protocolos usados em diferentes partes de uma aplicação pode causar sobrecarga operacional. Você pode padronizar o protocolo IPv6 para recursos do Amazon RDS a fim de simplificar sua configuração de rede. Para ter mais informações sobre os endpoints e cotas de serviço, consulte Endpoints e cotas do Amazon Relational Database Service.

Para ter mais informações sobre o endereçamento IP do Aurora, consulte Endereçamento IP do Aurora.

Analisar anomalias de desempenho do Aurora com o Amazon DevOps Guru para Amazon RDS

O Amazon DevOps Guru é um serviço de operações totalmente gerenciado que ajuda desenvolvedores e operadores a melhorar a performance e a disponibilidade de suas aplicações. O DevOps Guru dispensa as tarefas associadas à identificação de problemas operacionais, para que você possa implementar rapidamente recomendações para melhorar sua aplicação. Para ter mais informações, consulte O que é Configurar o Amazon DevOps Guru? no Guia do usuário do Amazon DevOps Guru.

O DevOps Guru detecta, analisa e faz recomendações de problemas operacionais para todos os mecanismos de banco de dados do Amazon RDS. O DevOps Guru para RDS amplia essa capacidade, aplicando machine learning a métricas do Performance Insights para bancos de dados do Amazon Aurora . Esses recursos de monitoramento permitem que o DevOps Guru para RDS detecte e diagnostique gargalos de performance e recomende ações corretivas específicas. O DevOps Guru para RDS também pode detectar condições problemáticas nos bancos de dados Aurora antes que elas ocorram.

Agora é possível ver essas recomendações no console do RDS. Para obter mais informações, consulte Recomendações do Amazon Aurora.

> **Important**
>
> O Amazon DevOps Guru não está disponível para clusters de banco de dados do Aurora Serverless.

O vídeo a seguir apresenta uma visão geral do DevOps Guru para RDS.

Para se aprofundar no assunto, consulte Bastidores do Amazon DevOps Guru para RDS.

Tópicos

- Benefícios do DevOps Guru para RDS
- Como funciona o DevOps Guru para RDS
- Configurar o DevOps Guru para RDS

Benefícios do DevOps Guru para RDS

Se você é responsável por um banco de dados Amazon Aurora, talvez não esteja ciente da existência de um evento ou regressão que está afetando esse banco de dados. Quando você ficar sabendo do problema, talvez não saiba por que ele está ocorrendo ou o que fazer a respeito. Em vez de recorrer a um administrador de banco de dados (DBA) para obter ajuda ou depender de ferramentas de terceiros, você pode seguir as recomendações do DevOps Guru para RDS.

Estas são as vantagens das análises detalhadas do DevOps Guru para RDS:

Diagnóstico rápido

O DevOps Guru para RDS monitora e analisa continuamente a telemetria do banco de dados. O Performance Insights, o Enhanced Monitoring e o Amazon CloudWatch coletam dados de telemetria para seu cluster de banco de dados. O DevOps Guru para RDS usa técnicas estatísticas e de machine learning para explorar esses dados e detectar anomalias. Para saber mais sobre dados de telemetria, consulte Monitoramento da carga do banco de dados com Performance Insights no Amazon Aurora e Monitoramento de métricas do sistema operacional com monitoramento aprimorado no Guia do usuário do Amazon Aurora .

Resolução rápida

Cada anomalia identifica o problema de performance e sugere rotas de investigação ou ações corretivas. Por exemplo, o DevOps Guru para RDS pode recomendar que você investigue eventos de espera específicos. Ou ele pode recomendar que você ajuste as configurações do seu grupo de aplicações para limitar o número de conexões de banco de dados. Com base nessas recomendações, é possível resolver problemas de performance mais rapidamente do que solucionando problemas manualmente.

Insights proativos

O DevOps Guru para RDS utiliza métricas de seus recursos para detectar um comportamento possivelmente problemático antes que ele se torne um problema maior. Por exemplo, ele pode detectar quando seu banco de dados está utilizando um número crescente de tabelas temporárias em disco, o que pode começar a afetar a performance. Depois, o DevOps Guru fornece recomendações para ajudar você a resolver problemas antes que eles se tornem problemas maiores.

Conhecimento profundo dos engenheiros da Amazon e de "machine learning

Para detectar problemas de performance e ajudar você a resolver gargalos, o DevOps Guru para RDS conta com machine learning (ML) e fórmulas matemáticas avançadas. Os engenheiros de

banco de dados da Amazon contribuíram para o desenvolvimento das descobertas do DevOps Guru para RDS, que englobam muitos anos gerenciando centenas de milhares de bancos de dados. Com base nesse conhecimento coletivo, o DevOps Guru para RDS pode ensinar práticas recomendadas para você.

Como funciona o DevOps Guru para RDS

O DevOps Guru para RDS coleta dados sobre seus bancos de dados Aurora por meio do Amazon RDS Performance Insights. A métrica mais importante é `DBLoad`. O DevOps Guru for RDS consome as métricas do Performance Insights, analisa-as com machine learning e publica insights no painel.

Um insight é uma coleção de anomalias relacionadas que foram detectadas pelo DevOps Guru.

No DevOps Guru para RDS, uma anomalia é um padrão que se desvia do que é considerada a performance normal do seu banco de dados Amazon Aurora.

Insights proativos

Um insight proativo informa você sobre um comportamento problemático antes que ele ocorra. Contém anomalias com recomendações e métricas relacionadas para ajudar você a resolver problemas em seus bancos de dados Amazon Aurora antes que se tornem problemas maiores. Esses insights são publicados no painel do DevOps Guru.

Por exemplo, o DevOps Guru pode detectar que seu banco de dados do Aurora PostgreSQL está criando muitas tabelas temporárias em disco. Se não for tratada, essa tendência poderá gerar problemas de performance. Cada insight proativo inclui recomendações para comportamento corretivo e links para tópicos relevantes em Ajustar o Aurora MySQL com insights proativos do Amazon DevOps Guru ou Ajustar o Aurora PostgreSQL com insights proativos do Amazon DevOps Guru. Para ter mais informações, consulte Trabalhar com insights no DevOps Guru no Guia do usuário do Amazon DevOps Guru.

Insights reativos

Um insight reativo identifica um comportamento anômalo quando ele ocorre. Se o DevOps Guru para RDS encontrar problemas de performance nas suas instâncias de banco de dados do Amazon Aurora , ele publicará um insight reativo no painel do DevOps Guru. Para ter mais informações, consulte Trabalhar com insights no DevOps Guru no Guia do usuário do Amazon DevOps Guru.

Anomalias causais

Uma anomalia causal é uma anomalia de nível superior dentro de um insight reativo. Carga do banco de dados é a anomalia causal do DevOps Guru para RDS.

Uma anomalia mede o impacto na performance, atribuindo um nível de gravidade de Alto, Médio ou Baixo. Para saber mais, consulte os Principais conceitos do DevOps Guru para RDS, no Guia do usuário do Amazon DevOps Guru.

Se o DevOps Guru detectar uma anomalia em sua instância de banco de dados, você será alertado na página Databases (Bancos de dados) do console do RDS. O console também alerta você sobre anomalias que ocorreram nas últimas 24 horas. Para acessar a página de anomalias no console do RDS, escolha o link na mensagem de alerta. O console do RDS também alerta você na página do cluster de banco de dados do Amazon Aurora .

Anomalias contextuais

Uma anomalia contextual é uma descoberta em Carga do banco de dados (carga do BD) que é relatada a um insight reativo. Cada anomalia contextual descreve um problema de performance específico do Amazon Aurora que requer investigação. Por exemplo, o DevOps Guru para RDS pode recomendar que você aumente a capacidade da CPU ou investigue eventos de espera que estão contribuindo para a carga do banco de dados.

> **Important**
>
> Convém testar todas as alterações na instância de teste antes de modificar a instância de produção. Dessa forma, você pode compreender o impacto da alteração.

Para saber mais, consulte Analisar anomalias no Amazon RDS no Guia do usuário do Amazon DevOps Guru.

Configurar o DevOps Guru para RDS

Para permitir que o DevOps Guru para Amazon RDS publique insights de um banco de dados do Amazon Aurora , conclua as tarefas a seguir.

Tópicos

- Configurar políticas de acesso do IAM para DevOps Guru para RDS

- Ativar o Performance Insights para suas instâncias de banco de dados do Aurora
- Ativar o DevOps Guru e especificar a cobertura de recursos

Configurar políticas de acesso do IAM para DevOps Guru para RDS

Para visualizar alertas do DevOps Guru no console do RDS, seu usuário ou perfil do AWS Identity and Access Management (IAM) deve ter uma das seguintes políticas:

- A política `AmazonDevOpsGuruConsoleFullAccess` gerenciada pelo AWS
- A política gerenciada `AmazonDevOpsGuruConsoleReadOnlyAccess` da AWS e uma das seguintes políticas:
  - A política `AmazonRDSFullAccess` gerenciada pelo AWS
  - Uma política gerenciada pelo cliente que inclua `pi:GetResourceMetrics` e `pi:DescribeDimensionKeys`

Para ter mais informações, consulte Configurar políticas de acesso para o Performance Insights.

Ativar o Performance Insights para suas instâncias de banco de dados do Aurora

O DevOps Guru para RDS depende do Performance Insights para seus dados. Sem o Performance Insights, o DevOps Guru publica anomalias, mas não inclui a análise e as recomendações detalhadas.

Ao criar um cluster de banco de dados do Aurora ou modificar uma instância de cluster, você pode ativar o Performance Insights. Para ter mais informações, consulte Ativar e desativar o Insights de Performance para o Aurora.

Ativar o DevOps Guru e especificar a cobertura de recursos

Você pode ativar o DevOps Guru para que ele monitore seus bancos de dados do Amazon Aurora de uma das maneiras a seguir.

Tópicos

- Ativar o DevOps Guru no console do RDS
- Adicionar recursos do Aurora no console do DevOps Guru
- Adicionar recursos do Aurora usando o CloudFormation

Ativar o DevOps Guru no console do RDS

Você pode seguir vários caminhos no console do Amazon RDS para ativar o DevOps Guru.

Tópicos

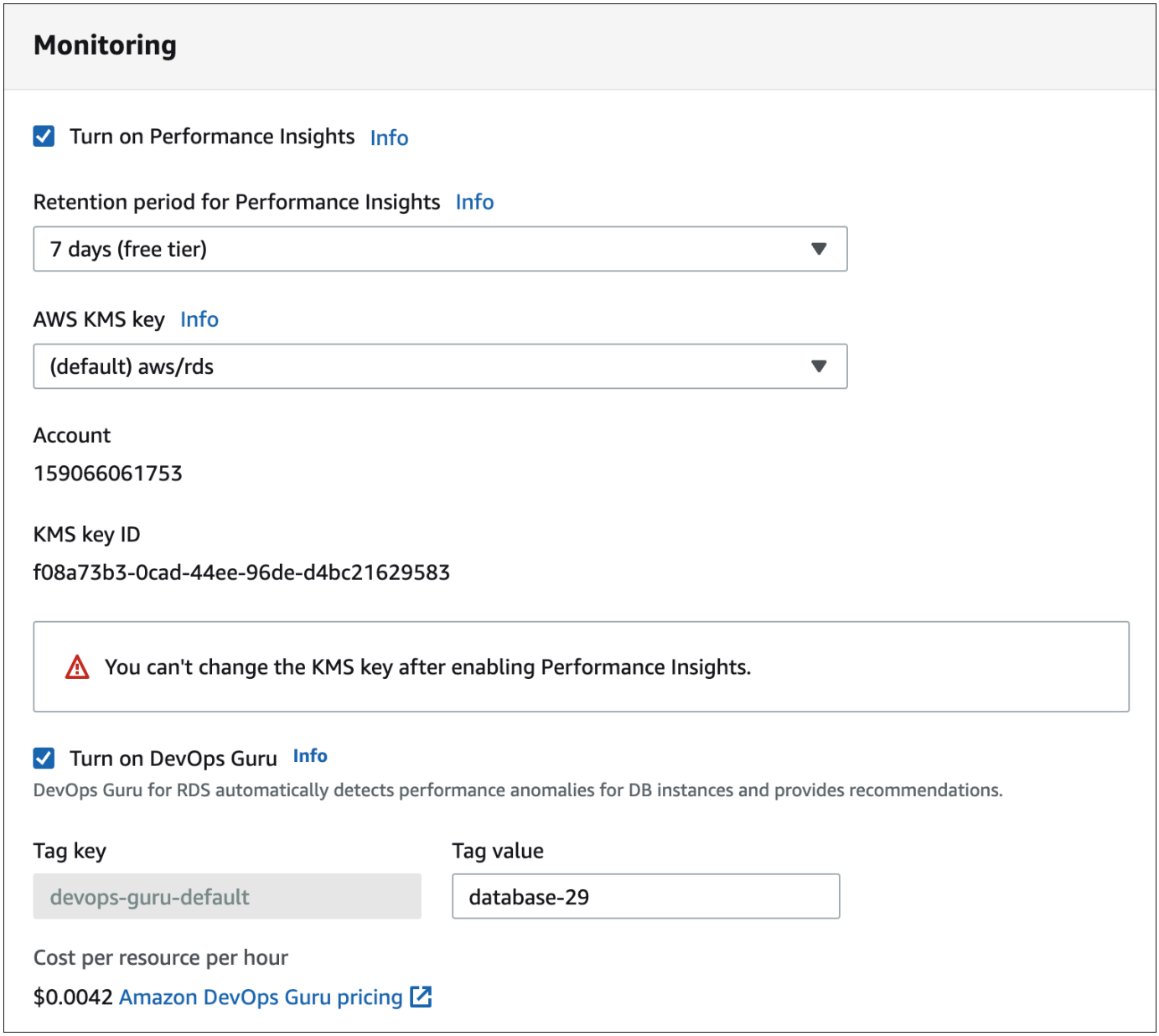
- Ativar o DevOps Guru ao criar um banco de dados do Aurora
- Ativar o DevOps Guru a partir do banner de notificação
- Responder a um erro de permissões quando você ativa o DevOps Guru

Ativar o DevOps Guru ao criar um banco de dados do Aurora

O fluxo de trabalho de criação inclui uma configuração que ativa a cobertura do DevOps Guru para seu banco de dados. Essa configuração é ativada por padrão quando você escolhe o modelo Production (Produção).

Como ativar o DevOps Guru ao criar um banco de dados do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Siga as etapas em Criar um cluster de banco de dados exceto a etapa em que você escolhe as configurações de monitoramento.
3. Em Monitoring (Monitoramento), escolha Turn on Performance Insights (Ativar Performance Insights). Para que o DevOps Guru para RDS forneça uma análise detalhada de anomalias de performance, o Performance Insights deve estar ativado.
4. Escolha Turn on DevOps Guru (Ativar DevOps Guru).



5. Crie uma tag para seu banco de dados para que o DevOps Guru possa monitorá-lo. Faça o seguinte:

   - No campo de texto para Tag key (Chave de tag), insira um nome que comece com **`Devops-Guru-`**.
   - No campo de texto para Tag value (Valor da tag), insira qualquer valor. Por exemplo, se você inserir **`rds-database-1`** para obter o nome do banco de dados do Aurora, também é possível inserir **`rds-database-1`** como o valor da tag.

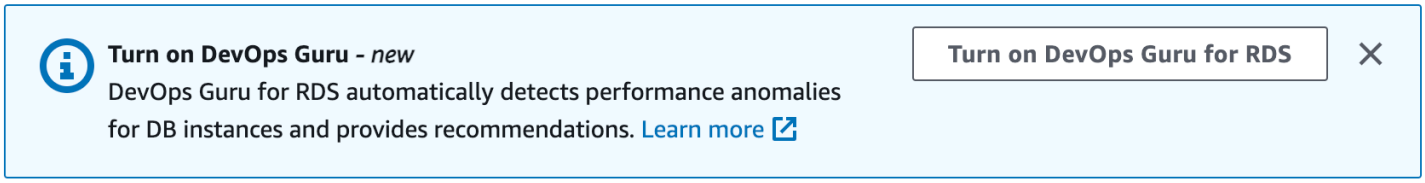
Para ter mais informações sobre tags, consulte "Usar tags para identificar recursos em suas aplicações do DevOps Guru" no Guia do usuário do Amazon DevOps Guru.

6. Conclua as etapas restantes em Criar um cluster de banco de dados.

Ativar o DevOps Guru a partir do banner de notificação

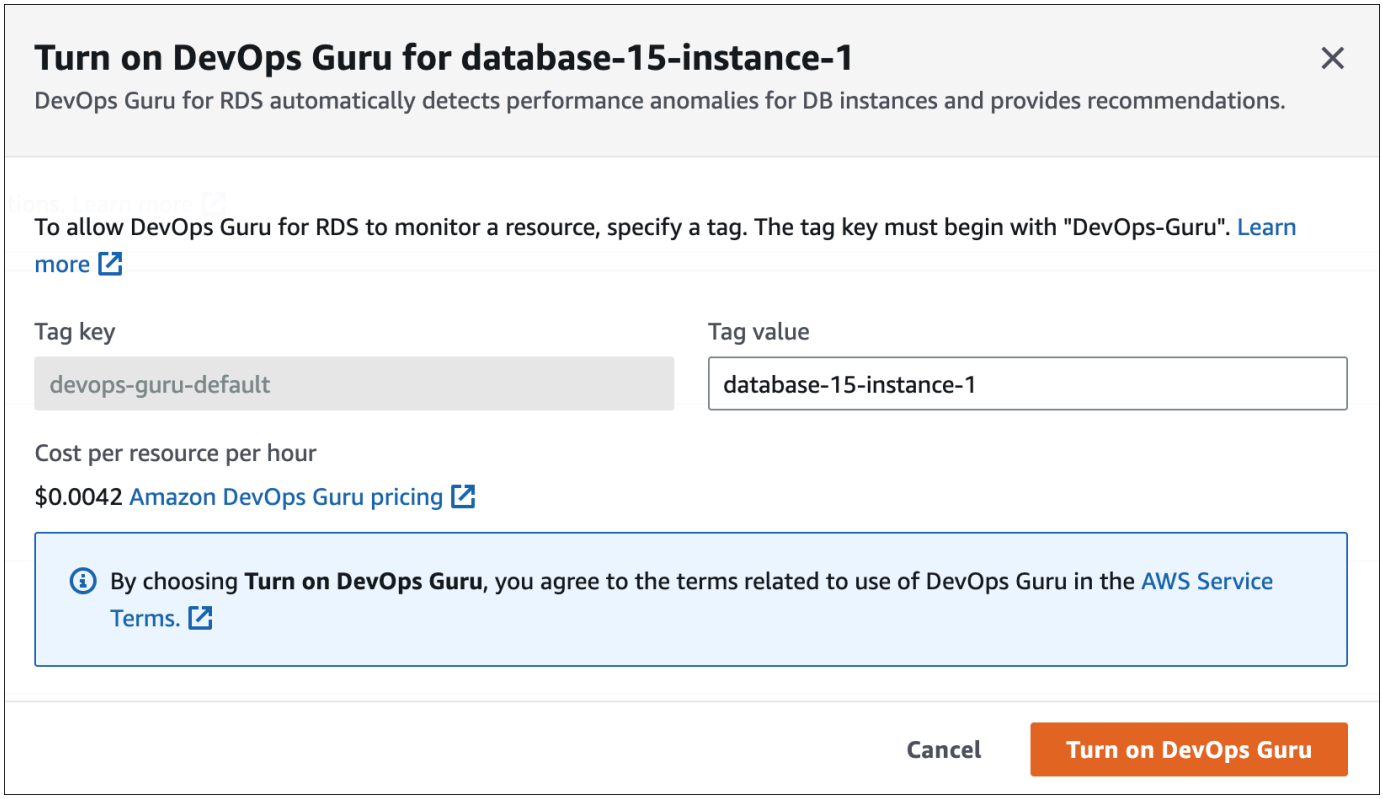
Se seus recursos não forem cobertos pelo DevOps Guru, o Amazon RDS notificará você com um banner nos seguintes locais:

- A guia Monitoring (Monitoramento) de uma instância de cluster de banco de dados
- O painel do Performance Insights



Como ativar o DevOps Guru para seu banco de dados do Aurora

1. No banner, escolha Turn on DevOps Guru for RDS (Ativar DevOps Guru para RDS).
2. Insira um nome de chave e um valor para a tag. Para ter mais informações sobre tags, consulte "Usar tags para identificar recursos em suas aplicações do DevOps Guru" no Guia do usuário do Amazon DevOps Guru.



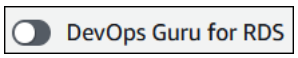
3. Escolha Turn on DevOps Guru (Ativar DevOps Guru).

Responder a um erro de permissões quando você ativa o DevOps Guru

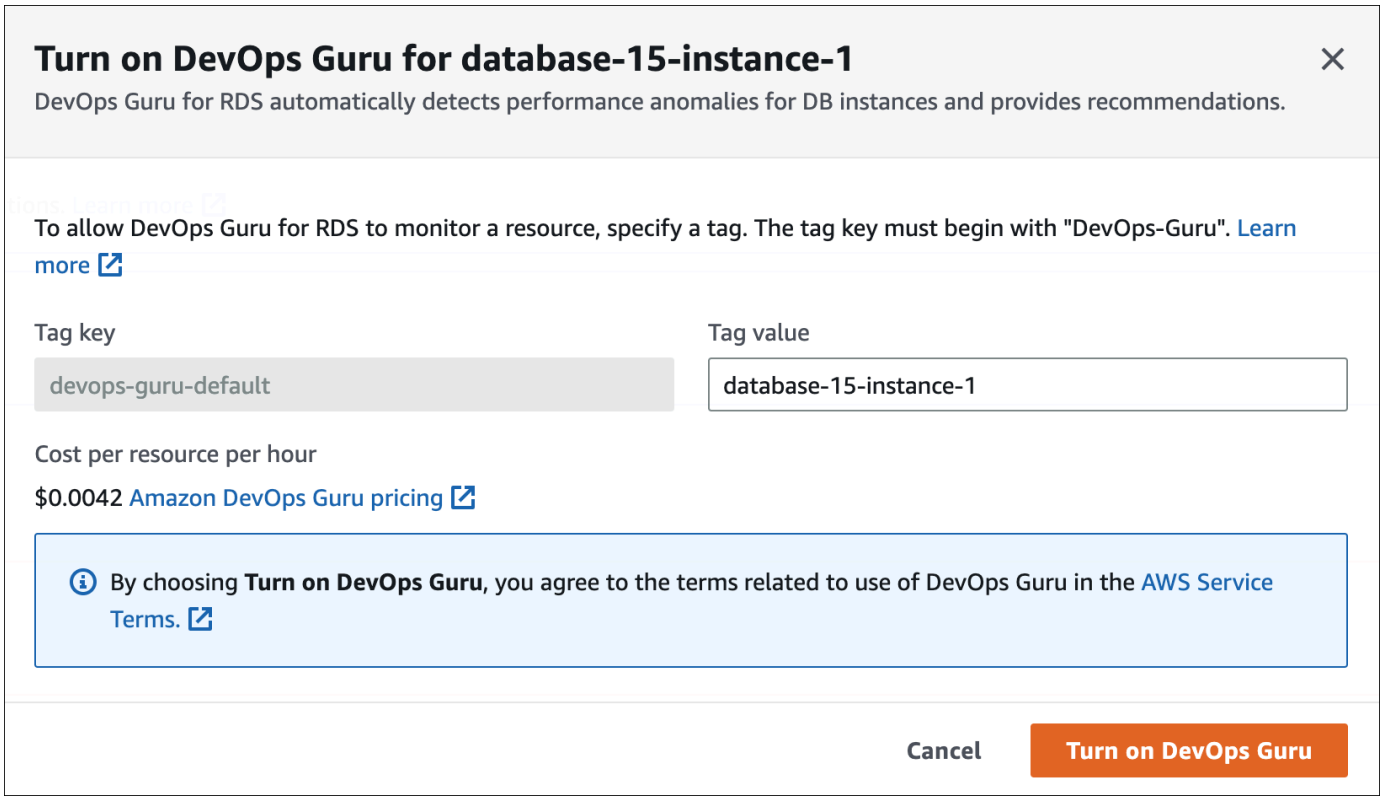
Se você ativar o DevOps Guru no console do RDS ao criar um banco de dados, o RDS poderá exibir o banner a seguir sobre permissões ausentes.

Como responder a um erro de permissões

1. Conceda ao usuário ou ao perfil do IAM a função gerenciada pelo usuário `AmazonDevOpsGuruConsoleFullAccess`. Para ter mais informações, consulte Configurar políticas de acesso do IAM para DevOps Guru para RDS.
2. Abra o console do RDS.
3. No painel de navegação, escolha Performance Insights.
4. Escola uma instância de banco de dados no cluster que você acabou de criar.
5. Selecione o switch para ativar o DevOps Guru para RDS.



6. Escolha um valor de tag. Para ter mais informações, consulte "Usar tags para identificar recursos em suas aplicações do DevOps Guru" no Guia do usuário do Amazon DevOps Guru.



7. Escolha Turn on DevOps Guru (Ativar DevOps Guru).

Adicionar recursos do Aurora no console do DevOps Guru

Você pode especificar a cobertura de recursos do DevOps Guru no console do DevOps Guru. Siga a etapa descrita em Especificar a cobertura de recursos do DevOps Guru no Guia do usuário do Amazon DevOps Guru. Ao editar os recursos analisados, escolha uma das seguintes opções:

- Selecione Todos os recursos da conta para analisar todos os recursos compatíveis, inclusive os bancos de dados do Aurora, em sua Conta da AWS e região.
- Selecione Pilhas do CloudFormation para analisar os bancos de dados do Aurora que estão nas pilhas escolhidas. Para ter mais informações, consulte Usar pilhas do AWS CloudFormation para identificar recursos em suas aplicações do DevOps Guru no Guia do usuário do Amazon DevOps Guru.

- Selecione Tags para analisar os bancos de dados do Aurora marcados. Para ter mais informações, consulte Usar tags para identificar recursos em suas aplicações do DevOps Guru no Guia do usuário do Amazon DevOps Guru.

Para ter mais informações, consulte Enable DevOps Guru (Ativar o DevOps Guru), no Guia do usuário do Amazon DevOps Guru.

Adicionar recursos do Aurora usando o CloudFormation

Você pode usar tags para adicionar cobertura dos recursos do Aurora aos modelos do CloudFormation. O procedimento a seguir pressupõe que você tenha um modelo do CloudFormation para sua instância de banco de dados do Aurora e para a pilha do DevOps Guru.

Como especificar uma instância de banco de dados do Aurora usando uma tag do CloudFormation

1. No modelo do CloudFormation para sua instância de banco de dados, defina uma tag usando um par de chave/valor.

   O exemplo a seguir atribui o valor `my-aurora-db-instance1` a `Devops-guru-cfn-default` para uma instância de banco de dados Aurora.

   ```
   MyAuroraDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBClusterIdentifier: my-aurora-db-cluster
       DBInstanceIdentifier: my-aurora-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-aurora-db-instance1
   ```

2. No modelo do CloudFormation para sua pilha do DevOps Guru, especifique a mesma tag em seu filtro de coleta de recursos.

   O exemplo a seguir configura o DevOps Guru para fornecer cobertura para o recurso com o valor da tag `my-aurora-db-instance1`.

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
   ```

```
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "devopsguru-my-aurora-db-instance1"
```

O exemplo a seguir fornece cobertura para todos os recursos dentro dos limites da aplicação `Devops-guru-cfn-default`.

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

Para ter mais informações, consulte AWS::DevOpsGuru::ResourceCollection e AWS::RDS::DBInstance no Guia do usuário do CloudFormation.

Monitorar métricas do SO com o monitoramento avançado

Com o monitoramento avançado, você pode monitorar o sistema operacional da sua instância de banco de dados em tempo real. As métricas de monitoramento avançado são úteis quando você quiser ver como os diferentes processos ou threads usam a CPU.

Tópicos

- Visão geral do monitoramento avançado
- Configurar e habilitar o monitoramento avançado
- Como visualizar métricas do SO no console do RDS
- Visualizar métricas do SO usando CloudWatch Logs

Visão geral do monitoramento avançado

O Amazon RDS dispõe de métricas em tempo real para o sistema operacional (SO) no qual sua instância de banco de dados é executada. Você pode visualizar todas as métricas e informações de processo do sistema das suas instâncias de banco de dados do RDS no console. Você pode gerenciar quais métricas deseja monitorar para cada instância e personalizar o painel de acordo com os seus requisitos. Para ver as descrições das métricas do monitoramento avançado, consulte Métricas do sistema operacional no monitoramento avançado.

O RDS fornece as métricas do monitoramento avançado à sua conta do Amazon CloudWatch Logs. Você pode criar filtros de métricas no CloudWatch com o CloudWatch Logs e exibir os gráficos no painel do CloudWatch. É possível consumir o resultado do JSON de monitoramento avançado do CloudWatch Logs em um sistema de monitoramento de sua escolha. Para obter mais informações, consulte Monitoramento avançado nas Perguntas frequentes do Amazon RDS.

Tópicos

- Diferenças entre o CloudWatch e métricas de monitoramento avançado
- Retenção de métricas do monitoramento avançado
- Custo do monitoramento avançado

Diferenças entre o CloudWatch e métricas de monitoramento avançado

Um hipervisor cria e executa as máquinas virtuais (VMs). Usando um hipervisor, a instância pode oferecer suporte a várias VMs convidadas compartilhando memória e CPU virtualmente. O

CloudWatch coleta métricas sobre a utilização da CPU do hipervisor para uma instância de banco de dados. Em contraste, o monitoramento avançado reúne as métricas de um agente na instância de banco de dados.

Você poderá encontrar diferenças entre as CloudWatch medidas do monitoramento avançado, pois a camada do hipervisor realiza uma pequena quantidade de trabalho. As diferenças podem ser maiores se as instâncias de banco de dados usarem classes de instância menores. Nesse cenário, mais máquinas virtuais (VMs) provavelmente são gerenciadas pela camada de hipervisor em uma única instância física.

Para ver as descrições das métricas do monitoramento avançado, consulte Métricas do sistema operacional no monitoramento avançado. Para ter mais informações sobre as métricas do CloudWatch, consulte o Guia do usuário do Amazon CloudWatch.

Retenção de métricas do monitoramento avançado

Por padrão, as métricas do monitoramento avançado são armazenadas por 30 dias no CloudWatch Logs. Esse período de retenção é diferente das métricas típicas do CloudWatch.

Para modificar o tempo em que as métricas são armazenadas nos logs do CloudWatch Logs, altere a retenção do grupo de logs do `RDSOSMetrics` no console do CloudWatch. Para obter mais informações, consulte Alterar retenção de dados de log em logs do CloudWatch no Amazon CloudWatch Logs User Guide.

Custo do monitoramento avançado

As métricas de monitoramento avançado são armazenadas no CloudWatch Logs e não métricas do CloudWatch. O custo do monitoramento avançado depende dos seguintes fatores:

- Só haverá cobrança pelo monitoramento avançado se você exceder a quantidade de transferência de dados e armazenamento oferecida pelo Amazon CloudWatch Logs. As cobranças são baseadas nas taxas de transferência de dados e armazenamento do CloudWatch Logs.
- A quantidade de informações transferidas para uma instância do RDS é diretamente proporcional ao detalhamento definido para o recurso de monitoramento avançado. Um menor intervalo de monitoramento resulta em relatórios mais frequentes das métricas do sistema operacional e aumenta seu custo de monitoramento. Para gerenciar custos, defina diferentes detalhamentos para diferentes instâncias em suas contas.

- Os custos de uso do Monitoramento avançado são aplicados a cada instância de banco de dados para a qual o Monitoramento avançado esteja habilitado. O monitoramento de um grande número de instâncias de banco de dados é mais caro do que o monitoramento de apenas algumas.
- Instâncias de bancos de dados que oferecem suporte a uma workload que exige computação intensiva têm mais atividade de processos de SO para reportar e custos mais altos para Monitoramento avançado.

Para obter mais informações sobre a definição de preço, consulte Definição de preço do Amazon CloudWatch.

Configurar e habilitar o monitoramento avançado

Para usar o monitoramento avançado, você deve criar uma função do IAM e habilitar o monitoramento avançado.

Tópicos

- Criar uma função do IAM para o monitoramento avançado
- Ativar e desativar o monitoramento aprimorado
- Proteção contra o problema do substituto confuso

Criar uma função do IAM para o monitoramento avançado

O Monitoramento avançado requer permissão para agir em seu nome para enviar informações de métricas do SO ao CloudWatch Logs. Você concede as permissões ao monitoramento avançado usando uma função do AWS Identity and Access Management (IAM). Você pode criar essa função ao ativar o monitoramento aprimorado ou criá-la de antemão.

Tópicos

- Criar a função do IAM ao habilitar o Monitoramento Avançado
- Criar a função do IAM antes de ativar o Monitoramento Avançado

Criar a função do IAM ao habilitar o Monitoramento Avançado

Quando você habilita o Monitoramento Avançado no console do RDS, o Amazon RDS pode criar a função do IAM necessária para você. A função é chamada `rds-monitoring-role`. O RDS usa

essa função para a instância de banco de dados especificada, réplica de leitura ou cluster de banco de dados multi-AZ.

Para criar a função do IAM ao habilitar o Monitoramento Avançado

1. Siga as etapas em Ativar e desativar o monitoramento aprimorado.
2. Defina a Monitoring Role (Função de monitoramento) como Default (Padrão) na etapa em que você escolhe uma função.

Criar a função do IAM antes de ativar o Monitoramento Avançado

Você pode criar a função necessária antes de habilitar o Monitoramento Avançado. Ao habilitar o Monitoramento Avançado, especifique o nome da nova função. Você deverá criar essa função necessária se habilitar o Monitoramento avançado usando a AWS CLI ou a API do RDS.

O usuário que habilita o monitoramento aprimorado precisa receber a permissão `PassRole`. Para obter mais informações, consulte o Exemplo 2 em Conceder permissões ao usuário para transmitir uma função para um serviço da AWS no Manual do usuário do IAM.

Como criar uma função do IAM para o monitoramento avançado do Amazon RDS

1. Abra o Console do IAM em https://console.aws.amazon.com.
2. No painel de navegação, selecione Roles.
3. Selecione Create role.
4. Escolha a guia Serviço da AWS e, em seguida, RDS na lista de serviços.
5. Escolha RDS - Enhanced Monitoring (RDS: monitoramento aprimorado) e Next (Próximo).
6. A página Permissions policies (Políticas de permissões) deve mostrar AmazonRDSEnhancedMonitoringRole. Escolha Next (Próximo).
7. Em Role name (Nome da função), digite um nome para sua função. Por exemplo, digite **`emaccess`**.

   A entidade confiável para sua função é o serviço da AWS monitoring.rds.amazonaws.com.
8. Selecione Create role (Criar função).

Ativar e desativar o monitoramento aprimorado

É possível gerenciar o monitoramento avançado usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Você pode definir detalhamentos diferentes para a coleta de métricas em cada cluster de banco de dados. Você também pode ativar o Monitoramento avançado para um cluster de banco de dados existente pelo console.

Console

É possível ativar o monitoramento avançado ao criar um cluster de banco de dados ou uma réplica de leitura, ou ao modificar um cluster de banco de dados. Se você modificar uma instância ou um cluster de banco de dados para ativar o Monitoramento avançado, não será necessário reinicializar a instância de banco de dados para que a alteração entre em vigor.

Você pode ativar o monitoramento aprimorado no console do RDS ao realizar uma das seguintes ações na página Databases (Banco de dados):

- ”Create a DB cluster” (Criar um cluster de banco de dados): escolha Create database (Criar banco de dados).
- Create a read replica (Criar uma réplica de leitura) — Escolha Actions (Ações) e depois Create read replica (Criar réplica de leitura).
- Modificar uma instância de banco de dados ou cluster de banco de dados : escolha Modificar.

> **Note**
>
> Ao habilitar o Monitoramento avançado para um cluster de banco de dados, não é possível gerenciá-lo para instâncias de banco de dados individuais dentro do cluster.

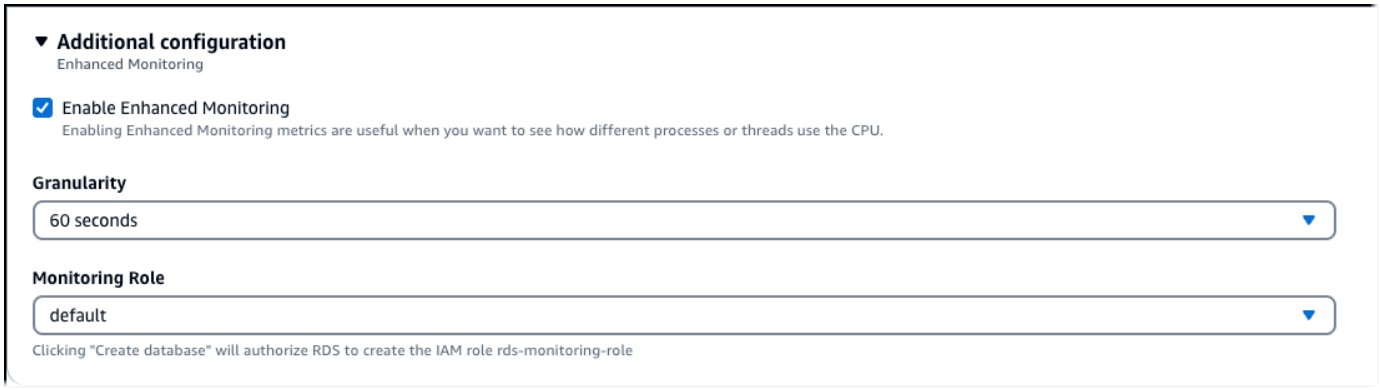
Se você estiver gerenciando o monitoramento avançado para instâncias de banco de dados individuais em um cluster de banco de dados e o detalhamento estiver definido com valores diferentes para instâncias diferentes, o cluster de banco de dados será heterogêneo em relação ao monitoramento avançado. Nesses casos, você não pode modificar o cluster de banco de dados para gerenciar o Monitoramento avançado no nível do cluster.

Para ativar ou desativar o monitoramento aprimorado no console do RDS

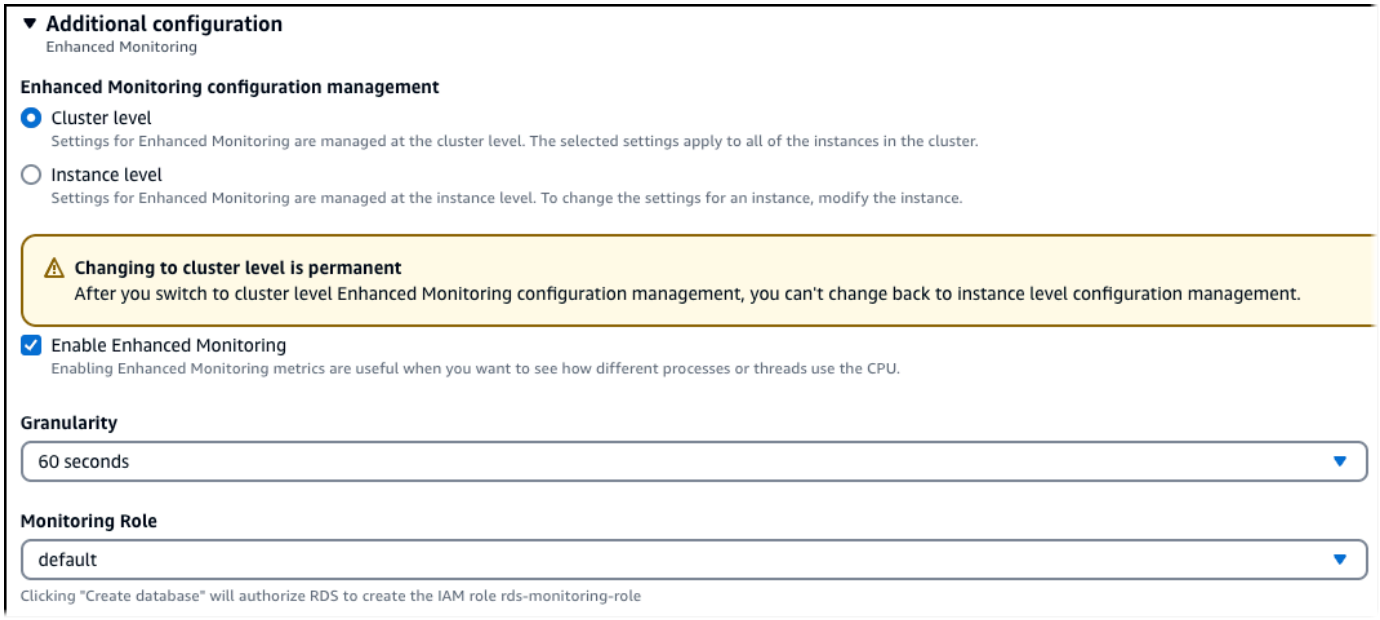
1. Role até Additional configuration (Configuração adicional).

2. Em Monitoramento, escolha Habilitar o Monitoramento avançado para o cluster de banco de dados ou a réplica de leitura. Habilitar o Monitoramento avançado no nível do cluster permite gerenciar as configurações e opções desse recurso no nível do cluster. As configurações no nível do cluster se aplicam a todas as instâncias de banco de dados no cluster.Desmarque a opção para desabilitar o Monitoramento avançado no nível do cluster. Posteriormente, é possível modificar as configurações do Monitoramento avançado para instâncias de banco de dados individuais no cluster.

   Na página Criar banco de dados, você pode optar por ativar o Monitoramento avançado no nível do cluster.



   Se não habilitar o Monitoramento avançado ao criar um cluster, você poderá modificar o cluster na página Modificar cluster de banco de dados.



> **Note**
>
> Não é possível gerenciar o monitoramento avançado para uma instância de banco de dados individual em um cluster de banco de dados que já tenha o monitoramento avançado gerenciado no nível do cluster.

3. Não é possível optar por gerenciar o monitoramento avançado no nível do cluster se o cluster for heterogêneo em relação ao monitoramento avançado. Para gerenciar o Monitoramento avançado no nível do cluster, altere as configurações do Monitoramento avançado de cada instância para que elas correspondam. Agora você pode optar por gerenciar o Monitoramento avançado no nível do cluster ao modificar o cluster.
4. Defina a propriedade Monitoring Role (Função de monitoramento) como a função do IAM que você criou para permitir que o Amazon RDS se comunique com o Amazon CloudWatch Logs por você. Ou escolha Default (Padrão) para que o RDS crie uma função para você chamada `rds-monitoring-role`.
5. Defina a propriedade Granularidade como o intervalo, em segundos, entre pontos quando as métricas são coletadas para a instância de banco de dados, o cluster de banco de dados ou a réplica de leitura. A propriedade Granularity (Granularidade) pode ser definida como um dos seguintes valores: `1`, `5`, `10`, `15`, `30` ou `60`.

   A frequência de atualização mais rápida do console do RDS é a cada 5 segundos. Se você definir a granularidade como 1 segundo no console do RDS, ainda verá as métricas atualizadas apenas a cada 5 segundos. É possível recuperar atualizações de métricas de 1 segundo usando a CloudWatch Logs.

AWS CLI

Para ativar o monitoramento avançado usando a AWS CLI, defina a opção `--monitoring-interval` nos comandos a seguir como um valor diferente de `0` e defina a opção `--monitoring-role-arn` para a função criada em Criar uma função do IAM para o monitoramento avançado.

- create-db-cluster
- modify-db-cluster
- create-db-instance
- create-db-instance-read-replica

- modify-db-instance

A opção `--monitoring-interval` especifica o intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas. Os valores válidos para a opção são `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Para desativar o monitoramento aprimorado usando a AWS CLI, defina a opção `--monitoring-interval` para `0` nestes comandos.

Example

O seguinte exemplo ativa o monitoramento avançado para uma instância de banco de dados:

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

O seguinte exemplo ativa o Monitoramento avançado para um cluster de banco de dados:

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

O seguinte exemplo ativa o monitoramento avançado para um cluster de banco de dados multi-AZ:

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

API do RDS

Para ativar o monitoramento aprimorado usando a API do RDS, defina o parâmetro `MonitoringInterval` como um valor diferente de `0` e defina o parâmetro `MonitoringRoleArn` para a função criada em Criar uma função do IAM para o monitoramento avançado. Defina esses parâmetros nas seguintes ações:

- CreateDBCluster
- ModifyDBCluster
- CreateDBInstance
- CreateDBInstanceReadReplica
- ModifyDBInstance

O parâmetro `MonitoringInterval` especifica o intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas. Os valores válidos são `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Para desativar o monitoramento aprimorado usando a API do RDS, defina `MonitoringInterval` como `0`.

Proteção contra o problema do substituto confuso

"Confused deputy" é um problema de segurança no qual uma entidade sem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Na AWS, a personificação entre serviços pode resultar no problema do 'confused deputy'. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta. Para obter mais informações, consulte O problema do substituto confuso.

Para limitar as permissões ao recurso que o Amazon RDS pode fornecer a outro serviço, recomendamos usar as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em uma política de confiança para sua função de monitoramento aprimorado. Se você usar as duas chaves de contexto de condição global, elas deverão usar o mesmo ID de conta.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Para o Amazon RDS, defina `aws:SourceArn` como `arn:aws:rds:`*`Region`*`:`*`my-account-id`*`:db:`*`dbname`*.

O exemplo a seguir usa as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em uma política de confiança para evitar o problema de substituto confuso.

JSON

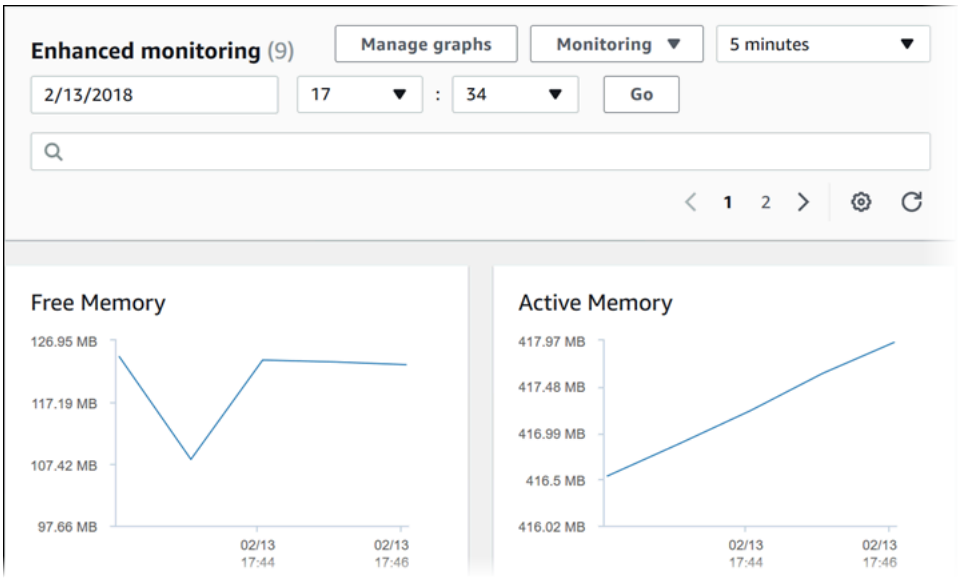
```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
```

```
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-123456789012:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-123456789012"
        }
      }
    }
  ]
}
```

Como visualizar métricas do SO no console do RDS

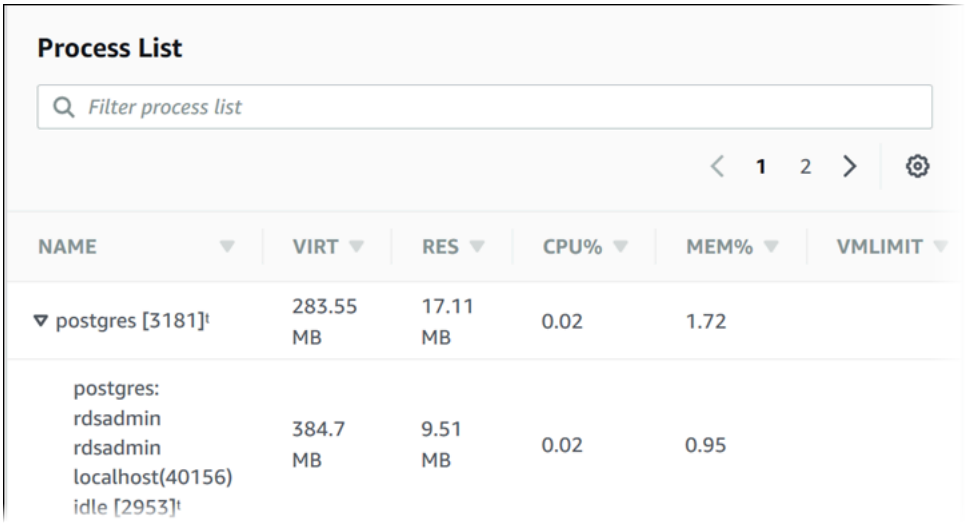
Você pode visualizar as métricas do sistema operacional informadas pelo Monitoramento avançado no console do RDS escolhendo a exibição Enhanced monitoring (Monitoramento avançado) para Monitoring (Monitoramento).

O exemplo a seguir mostra a página Monitoramento avançado. Para ver as descrições das métricas do monitoramento avançado, consulte Métricas do sistema operacional no monitoramento avançado.



Se quiser ver detalhes dos processos em execução na sua instância de banco de dados, escolha OS process list (Lista de processos do SO) para Monitoring (Monitoramento).

A visualização Process List (Lista de processos) é mostrada a seguir.



As métricas de Monitoramento avançado mostradas na visualização Process list (Lista de processos) estão organizadas da seguinte forma:

- RDS child processes (Processos filho do RDS) – mostra um resumo dos processos do RDS que oferecem suporte à instância de banco de dados, por exemplo, `aurora` para clusters de bancos de dados Amazon Aurora e . Os segmentos de processos aparecem aninhados abaixo do processo pai. Os threads de processos mostram a utilização da CPU apenas enquanto as outras métricas forem as mesmas para todos os threads do processo. O console exibe um máximo de 100 processos e threads. Os resultados são uma combinação dos principais processos e threads que consomem CPU e memória. Se houver mais de 50 processos e mais de 50 threads, o console exibirá os 50 melhores consumidores em cada categoria. Essa exibição ajuda a identificar quais processos estão tendo o maior impacto sobre a performance.
- RDS processes (Processos do RDS): mostra um resumo dos recursos utilizados pelo agente de gerenciamento do RDS, processos de monitoramento de diagnóstico e outros processos da AWS necessários para dar suporte a instâncias de bancos de dados do RDS.
- OS processes (Processos do SO) – Mostra um resumo dos processos de kernel e do sistema, que geralmente têm um impacto mínimo sobre a performance.

Os itens listados para cada processo são:

- VIRT – Exibe o tamanho virtual do processo.
- RES – Exibe a memória física real que está sendo usada pelo processo.
- CPU% – exibe a porcentagem da largura de banda total da CPU que está sendo usada pelo processo.

- MEM% – exibe a porcentagem da memória total que está sendo consumida pelo processo.

Os dados de monitoramento que são mostrados no console do RDS são recuperados do Amazon CloudWatch Logs. Você também pode recuperar as métricas para uma instância de banco de dados como um stream de log do CloudWatch Logs. Para obter mais informações, consulte Visualizar métricas do SO usando CloudWatch Logs.

Métricas de Monitoramento avançado não são retornadas durante o seguinte:

- Um failover da instância de banco de dados.
- Alteração da classe da instância de banco de dados (computação de escala).

As métricas de Monitoramento avançado são retornadas durante uma reinicialização de uma instância de banco de dados, pois somente o mecanismo do banco de dados é reinicializado. Métricas para o sistema operacional ainda serão informadas.

Visualizar métricas do SO usando CloudWatch Logs

Depois de ativar o monitoramento avançado em seu cluster de banco de dados, você poderá exibir as respectivas métricas usando o CloudWatch Logs, com cada transmissão de log representando uma única instância ou cluster de banco de dados em monitoramento. O identificador da transmissão de log é o identificador de recurso (`DbiResourceId`) da instância ou cluster de banco de dados.

Para visualizar os dados de log de Monitoramento avançado

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. Se necessário, selecione a Região da AWS em que seu cluster de banco de dados está. Para obter mais informações, consulte Regiões e endpoints do na Referência geral do Amazon Web Services.
3. Escolha Logs no painel de navegação.
4. Escolha RDSOSMetrics na lista de grupos de log.
5. Escolha o stream de log que você deseja visualizar na lista de streams de log.

Referência de métricas do Amazon Aurora

Nesta referência, você pode encontrar descrições de métricas do Amazon Aurora referentes ao Amazon CloudWatch, Performance Insights e monitoramento avançado.

Tópicos

- Métricas do Amazon CloudWatch para o Amazon Aurora
- Dimensões do Amazon CloudWatch para o Aurora
- Disponibilidade de métricas Aurora no console Amazon RDS
- Métricas do Amazon CloudWatch para Insights de Performance do Amazon RDS
- Métricas de contadores do Performance Insights
- Estatísticas SQL para Performance Insights
- Métricas do sistema operacional no monitoramento avançado

Métricas do Amazon CloudWatch para o Amazon Aurora

O namespace `AWS/RDS` inclui as métricas a seguir aplicáveis a entidades de bancos de dados em execução no Amazon Aurora. Algumas métricas se aplicam ao Aurora MySQL, ao Aurora PostgreSQL ou a ambos. Além disso, algumas métricas são específicas de um cluster de banco de dados, uma instância de banco de dados primária, uma instância de banco de dados de réplica ou todas as instâncias de banco de dados.

Para métricas de banco de dados globais do Aurora, consulte Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora MySQL e Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora PostgreSQL. Para métricas de consulta paralelas do Aurora, consulte Monitorar a consulta paralela do Aurora MySQL.

Tópicos

- Métricas no nível do cluster do Amazon Aurora
- Métricas no nível da instância do Amazon Aurora
- Métricas de uso do Amazon CloudWatch para Amazon Aurora

Métricas no nível do cluster do Amazon Aurora

A tabela a seguir descreve métricas específicas de clusters do Aurora.

Métrica	Descrição	Aplica-se a	Unidades
`AuroraGlobalDBDataTransferBytes`	Em um banco de dados global do Aurora, a quantidade de dados de redo log transferidos da região da AWS de origem para uma região secundária da AWS. **Note** Essa métrica está disponível somente na secundária Regiões da AWS.	Aurora MySQL e Aurora PostgreSQL	Bytes
`AuroraGlobalDBProgressLag`	Em um Aurora Global Database, a avaliação da distância entre o volume de armazenamento do cluster secundário e o volume de armazenamento do cluster primário para transações de usuário e transações do sistema. **Note** Essa métrica está disponível somente na secundária Regiões da AWS.	Aurora MySQL e Aurora PostgreSQL	Milissegundos
`AuroraGlobalDBReplicatedWriteIO`	Em um banco de dados global Aurora, o número	Aurora MySQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
	de operações de E/S de gravação replicadas da região principal da AWS para o volume do cluster em uma região secundária da AWS. Os cálculos de faturamento para as regiões secundárias da AWS em um banco de dados global usam o `VolumeWriteIOPs` para contabilizar as gravações realizadas no cluster. Os cálculos de faturamento da região principal da AWS em um banco de dados global usam o `VolumeWriteIOPs` para contabilizar a atividade de gravação nesse cluster e o `AuroraGlobalDBReplicatedWriteIO` para contabilizar a replicação entre regiões no banco de dados global. **Note** Essa métrica está disponível somente na secundária Regiões da AWS.	e Aurora PostgreSQL	

Métrica	Descrição	Aplica-se a	Unidades
`AuroraGlobalDBReplicationLag`	Para um Aurora Global Database, o tempo médio decorrido para replicar atualizações entre o servidor de replicação do cluster primário e o servidor de replicação do cluster secundário. **Note** Essa métrica está disponível somente na secundária Regiões da AWS.	Aurora MySQL e Aurora PostgreSQL	Milissegundos
`AuroraGlobalDBRPOLag`	Em um banco de dados global Aurora, o tempo de atraso do objetivo de ponto de recuperação (RPO). Essa métrica avalia até onde o cluster secundário está atrás do cluster primário para transações de usuários. **Note** Essa métrica está disponível somente na secundária Regiões da AWS.	Aurora MySQL e Aurora PostgreSQL	Milissegundos

Métrica	Descrição	Aplica-se a	Unidades
`AuroraVolumeBytesLeftTotal`	O espaço disponível restante para o volume do cluster. À medida que o volume do cluster aumenta, esse valor diminui. Se chegar a zero, o cluster reportará um erro de falta de espaço. Se você quiser detectar se o cluster do Aurora MySQL está se aproximando do limite de tamanho de 128 tebibytes (TiB), esse valor é mais simples e mais confiável de monitorar do que `VolumeBytesUsed` . O `AuroraVolumeBytesLeftTotal` leva em consideração o armazenamento utilizado para a manutenção interna e outras alocações que não afetam o faturamento de armazenamento.	Aurora MySQL	Bytes
`BacktrackChangeRecordsCreationRate`	O número de registros de alteração de retrocesso criados durante cinco minutos para seu cluster de banco de dados.	Aurora MySQL	Contagem a cada 5 minutos

Métrica	Descrição	Aplica-se a	Unidades
`BacktrackChangeRecordsStored`	O número de registros de alteração de retrocesso usados por seu cluster de banco de dados.	Aurora MySQL	Contagem
`BackupRetentionPeriodStorageUsed`	A quantidade total de armazenamento de backup usada para oferecer suporte ao recurso de restauração point-in-time dentro da janela de retenção do backup do cluster de banco de dados do Aurora. Esse valor está incluído no total relatado pela métrica `TotalBackupStorageBilled` . Ele é calculado separadamente para cada cluster do Aurora. Para obter instruções, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Bytes
`ServerlessDatabaseCapacity`	A capacidade atual de um cluster de banco de dados do Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
`SnapshotStorageUsed`	A quantidade total de armazenamento de backup consumida por todos os snapshots do Aurora para um cluster de banco de dados do Aurora fora da janela de retenção de backup. Esse valor está incluído no total relatado pela métrica `TotalBackupStorageBilled`. Ele é calculado separadamente para cada cluster do Aurora. Para obter instruções, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`TotalBackupStorageBilled`	A quantidade total de armazenamento de backup em bytes para a qual você é cobrado para determina do cluster de banco de dados do Aurora. A métrica inclui o armazenamento de backup medido pelas métricas `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed` . Essa métrica é calculada separadamente para cada cluster do Aurora. Para obter instruções, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`VolumeBytesUsed`	A quantidade de armazenamento usada pelo cluster de bancos de dados do Aurora. Esse valor afeta o custo do cluster de bancos de dados do Aurora (para obter informações sobre preços, consulte a Página de preços do Amazon RDS). Esse valor não reflete algumas alocações de armazenamento interno que não afetam o faturamento de armazenamento. Sobre o Aurora MySQL, é possível antecipar problemas de falta de espaço com maior precisão ao testar se `AuroraVolumeBytesLeftTotal` está se aproximando de zero em vez de comparar `VolumeBytesUsed` com o limite de armazenamento de 128 TiB. Em relação a clusters que são clones, o valor dessa métrica depende do volume de dados adicionados ou alterados no clone. A métrica também pode	Aurora MySQL e Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
	aumentar ou diminuir quando o cluster original é excluído ou à medida que novos clones são adicionados ou excluídos. Para obter detalhes, consulte Excluir um volume de cluster de origem Observe que não faz sentido escolher um valor de `--period` que seja pequeno, porque o Amazon RDS coleta essas métricas em intervalos, não continuamente.		

Métrica	Descrição	Aplica-se a	Unidades
`VolumeReadIOPs`	O número de operações de E/S de leitura faturadas a partir de um volume de cluster em um intervalo de 5 minutos. As operações de leitura faturadas são calculadas no nível de volume do cluster, agregadas a partir de todas as instâncias no cluster de bancos de dados Aurora e posteriormente relatadas em intervalos de 5 minutos. O valor é calculado tomando o valor da métrica de Operações de Leitura em um período de 5 minutos. Você pode determinar a quantidade de operações de leitura faturadas por segundo, tomando o valor da métrica de Operações de leitura faturadas e dividindo por 300 segundos. Por exemplo, se as Operações de leitura faturadas retornarem 13.686, as operações de leitura faturadas por segundo serão 45 (13.686/300 = 45,62). Você acumula operações de leitura faturadas para	Aurora MySQL e Aurora PostgreSQL	Contagem a cada 5 minutos

Métrica	Descrição	Aplica-se a	Unidades
	consultas que solicitam páginas de banco de dados que não estão no cache do buffer e devem ser carregadas a partir do armazenamento. Você pode perceber picos em operações de leitura faturadas, pois os resultados da consulta são lidos a partir do armazenamento e depois são carregados no cache do buffer. Observe que não faz sentido escolher um valor de `--period` que seja pequeno, porque o Amazon RDS coleta essas métricas em intervalos, não continuamente. **Tip** Se seu cluster do Aurora MySQL usar consulta paralela, você poderá ver um aumento nos valores de `VolumeReadIOPS` . As consultas paralelas não usam o grupo de buffers. Assim,		

Métrica	Descrição	Aplica-se a	Unidades
	embora as consultas sejam rápidas, esse processamento otimizado pode resultar em aumento nas operações de leitura e nas cobranças associadas.		
`VolumeWriteIOPs`	O número de operações de E/S do disco de gravação no volume de cluster, relatado em intervalos de 5 minutos. Para obter uma descrição detalhada de como as operações de gravação faturadas são calculadas, consulte `VolumeReadIOPs` . Observe que não faz sentido escolher um valor de `--period` que seja pequeno, porque o Amazon RDS coleta essas métricas em intervalos, não continuamente.	Aurora MySQL e Aurora PostgreSQL	Contagem a cada 5 minutos

Métricas no nível da instância do Amazon Aurora

As métricas específicas da instância do Amazon CloudWatch a seguir se aplicam a todas as instâncias do Aurora MySQL e do Aurora PostgreSQL, salvo indicação em contrário.

Métrica	Descrição	Aplica-se a	Unidades
`AbortedClients`	O número de conexões de cliente que foram fechadas corretamente.	Aurora MySQL	Contagem
`ActiveTransactions`	O número médio de transações atuais em execução em uma instância de banco de dados Aurora por segundo. Por padrão, o Aurora não habilita essa métrica. Para começar a medir esse valor, defina `innodb_monitor_enable='all'` no grupo de parâmetros de banco de dados para uma instância de banco de dados específica.	Aurora MySQL	Contagem por segundo
`ACUUtilization`	O valor da métrica `ServerlessDatabaseCapacity` dividida pelo valor máximo de ACU do cluster de banco de dados. Essa métrica é aplicável somente para o Aurora Sem Servidor v2.	Aurora MySQL e Aurora PostgreSQL	Porcentagem
`AuroraBinlogReplicaLag`	O tempo de atraso de um cluster de banco de dados de réplica de log binário em execução no Aurora MySQL em relação à origem de replicação de log binário. Um	Principal para Aurora MySQL	Segundos

Métrica	Descrição	Aplica-se a	Unidades
	atraso significa que a fonte está gerando registros mais rapidamente do que a réplica pode aplicá-los. Essa métrica relata valores diferentes dependendo da versão do mecanismo: Aurora MySQL versão 2 O campo `Seconds_Behind_Master` de `SHOW SLAVE STATUS` do MySQL Aurora MySQL versão 3 `SHOW REPLICA STATUS` Você pode usar essa métrica para monitorar erros e atraso de réplica em um cluster que atua como uma réplica de log binário. O valor da métrica indica o seguinte: Um alto valor A réplica está atrasando a fonte da replicação. **`0`** ou um valor próximo de **`0`** O processo de réplica está ativo e atual.		

Métrica	Descrição	Aplica-se a	Unidades
	-1 O Aurora não consegue determinar o atraso, o que pode acontecer durante a configuração da réplica ou quando a réplica estiver em um estado de erro. Como a replicação de log binário ocorre somente na instância do gravador do cluster, recomendamos usar a versão dessa métrica associada à função WRITER. Para ter mais informações sobre como administrar a replicação, consulte Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS. Para ter mais informações sobre solução de problemas, consulte Problemas de replicação no Amazon Aurora MySQL.		
`AuroraDMLRejectedMasterFull`	O número de consultas encaminhadas que são rejeitadas porque a sessão está cheia na instância de banco de dados do gravador.	Primária para o Aurora MySQL versão 2	Contagem

Métrica	Descrição	Aplica-se a	Unidades
`AuroraDMLRejectedWriterFull`	O número de consultas encaminhadas que são rejeitadas porque a sessão está cheia na instância de banco de dados do gravador.	Primária para o Aurora MySQL versão 3	Contagem
`AuroraEstimatedSharedMemoryBytes`	A quantidade estimada de memória compartilhada de buffer ou grupo de buffers que foi usada ativamente durante o último intervalo de pesquisa configurado.	Aurora PostgreSQL	Bytes
`AuroraMemoryHealthState`	Indica o estado de integrida de da memória. Um valor de `0` é igual a `NORMAL`. Um valor de `10` é igual a `RESERVED`, o que significa que o servidor está se aproximando de um nível crítico de uso de memória. Para obter mais informaçõ es, consulte Soluciona r problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.06.1 e posterior	Indicador

Métrica	Descrição	Aplica-se a	Unidades
`AuroraMemoryNumDeclinedSqlTotal`	O número incremental de consultas recusadas como parte de uma tentativa de evitar o esgotamento de memória (OOM). Para obter mais informações, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.06.1 e posterior	Contagem
`AuroraMemoryNumKillConnTotal`	O número incremental de conexões fechadas como parte de uma tentativa de evitar o OOM. Para obter mais informações, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.06.1 e posterior	Contagem
`AuroraMemoryNumKillQueryTotal`	O número incremental de consultas finalizadas como parte de uma tentativa de evitar o OOM. Para obter mais informações, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.06.1 e posterior	Contagem

Métrica	Descrição	Aplica-se a	Unidades
`AuroraMilliseconds SpentInOomRecovery`	O tempo decorrido desde que a integridade da memória caiu abaixo do estado normal. Para obter mais informaçõ es, consulte Soluciona r problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.08.0 e posterior	Milissegu ndos
`AuroraNumOomRecove rySuccessful`	O número de vezes em que a integridade da memória foi restaurada ao estado normal. Para obter mais informaçõ es, consulte Soluciona r problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.08.0 e posterior	Contagem
`AuroraNumOomRecove ryTriggered`	O número de vezes em que a integridade da memória caiu abaixo do estado normal. Para obter mais informaçõ es, consulte Soluciona r problemas de falta de memória em bancos de dados do Aurora MySQL.	Aurora MySQL versão 3.08.0 e posterior	Contagem

Métrica	Descrição	Aplica-se a	Unidades
`AuroraOptimizedReadsCacheHitRatio`	A porcentagem de solicitações atendidas pelo cache de leituras otimizadas. O valor é calculado usando a seguinte fórmula: `orcache_blks_hit/(orcache_blks_hit + storage_blks_read)` Quando `AuroraOptimizedReadsCacheHitRatio` é 100%, significa que todas as páginas foram lidas no cache de leituras otimizadas. Quando `AuroraOptimizedReadsCacheHitRatio` é `0`, significa que nenhuma página foi lida no cache de leituras otimizadas.	Principal para Aurora PostgreSQL	Porcentagem

Métrica	Descrição	Aplica-se a	Unidades
`AuroraReplicaLag`	Em um cluster do Aurora de região única ou cluster primário de banco de dados global, o tempo decorrido na replicação de atualizações em uma instância de réplica da instância do gravador na mesma região. Em um cluster secundário de banco de dados global, a o tempo decorrido na replicação de atualizações na instância de réplica e no servidor de replicação do cluster secundário na mesma região.	Réplica do Aurora MySQL e do Aurora PostgreSQL	Milissegundos

Métrica	Descrição	Aplica-se a	Unidades
`AuroraReplicaLagMaximum`	A quantidade máxima de atraso entre a instância principal e qualquer instância de bancos de dados Aurora no cluster de banco de dados. Quando as réplicas de leitura são excluídas ou renomeadas, pode haver um aumento temporário no atraso de replicação à medida que o recurso antigo passa por um processo de reciclagem. Para ter uma representação precisa do atraso de replicação durante esse período, recomendamos que você monitore a métrica `AuroraReplicaLag` em cada instância de réplica de leitura.	Principal para Aurora MySQL e Aurora PostgreSQL	Milissegundos
`AuroraReplicaLagMinimum`	A quantidade mínima de atraso entre a instância principal e qualquer instância de bancos de dados Aurora no cluster de banco de dados.	Principal para Aurora MySQL e Aurora PostgreSQL	Milissegundos

Métrica	Descrição	Aplica-se a	Unidades
`AuroraSlowConnectionHandleCount`	O número de conexões que esperaram dois segundos ou mais para iniciar o handshake. Esse parâmetro se aplica apenas ao Aurora MySQL versão 3.	Aurora MySQL	Contagem
`AuroraSlowHandshakeCount`	O número de conexões que levaram 50 milissegundos ou mais para concluir o handshake. Esse parâmetro se aplica apenas ao Aurora MySQL versão 3.	Aurora MySQL	Contagem
`BacktrackWindowActual`	A diferença entre a janela retrógrada de destino e a janela retrógrada real.	Principal para Aurora MySQL	Minutos
`BacktrackWindowAlert`	O número de vezes que a janela retrógrada real é inferior à janela retrógrada de destino por um determinado período.	Principal para Aurora MySQL	Contagem
`BlockedTransactions`	O número médio de transações no banco de dados que são bloqueadas por segundo.	Aurora MySQL	Contagem por segundo

Métrica	Descrição	Aplica-se a	Unidades
`BufferCacheHitRatio`	A porcentagem de solicitações atendidas pelo cache de buffer.	Aurora MySQL e Aurora PostgreSQL	Porcentagem
`CommitLatency`	Tempo médio que o mecanismo e o armazenamento levam para concluir as operações de confirmação.	Aurora MySQL e Aurora PostgreSQL	Milissegundos
`CommitThroughput`	O número médio de operações de confirmação por segundo.	Aurora MySQL e Aurora PostgreSQL	Contagem por segundo
`ConnectionAttempts`	O número de tentativas de conexão com uma instância, sejam elas bem-sucedidas ou não.	Aurora MySQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
`CPUCreditBalance`	O número de créditos de CPU acumulados por uma instância, relatados em intervalos de 5 minutos. É possível usar essa métrica para determinar quanto tempo uma instância de banco de dados pode ser intermitente além do nível de performance da linha de base a uma taxa específica. Essa métrica se aplica apenas às seguintes classes de instância: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais	Aurora MySQL e Aurora PostgreSQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
	detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados. Os créditos de execução funcionam da mesma forma no Amazon RDS e no Amazon EC2. Para ter mais informações, consulte Launch credits (Créditos de execução) no “Amazon Elastic Compute Cloud User Guide for Linux Instances” (Guia do usuário do Amazon Elastic Compute Cloud para instâncias do Linux).		

Métrica	Descrição	Aplica-se a	Unidades
`CPUCreditUsage`	O número de créditos de CPU consumidos durante o período especificado, relatados em intervalos de 5 minutos. Essa métrica mede o tempo durante o qual as CPUs físicas foram usadas para instruções de processamento por CPUs virtuais alocadas à instância de banco de dados. Essa métrica se aplica apenas às seguintes classes de instância: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais	Aurora MySQL e Aurora PostgreSQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
	detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.		
`CPUSurplusCreditBalance`	O número de créditos excedentes gastos por uma instância ilimitada quando seu valor `CPUCreditBalance` é zero. O valor `CPUSurplusCreditBalance` é pago pelos créditos de CPU ganhos. Se o número de créditos excedentes ultrapassar o número máximo de créditos que a instância pode ganhar em um período de 24 horas, os créditos excedentes gastos acima do limite máximo incorrerão em uma taxa adicional. As métricas de crédito de CPU estão disponíveis apenas a uma frequência de 5 minutos.	Aurora MySQL e Aurora PostgreSQL	Créditos (minutos de vCPU)

Métrica	Descrição	Aplica-se a	Unidades
`CPUSurplusCreditsCharged`	O número de créditos excedentes gastos que não são pagos pelos créditos de CPU ganhos e que, portanto, incorrem em uma cobrança adicional. Os créditos excedentes gastos são cobrados quando uma das seguintes situações ocorre: • Os créditos excedentes ultrapassaram o número máximo de créditos que a instância pode obter em um período de 24 horas. Os créditos excedentes gastos acima do limite máximo são cobrados no final da hora. • A instância é interrompida ou encerrada. • A instância é alterada de `unlimited` para `standard`. As métricas de crédito de CPU estão disponíveis apenas a uma frequência de 5 minutos.	Aurora MySQL e Aurora PostgreSQL	Créditos (minutos de vCPU)

Métrica	Descrição	Aplica-se a	Unidades
`CPUUtilization`	A porcentagem de CPU usada por uma instância de bancos de dados Aurora.	Aurora MySQL e Aurora PostgreSQL	Porcentagem
`DatabaseConnections`	O número de conexões de rede cliente com a instância do banco de dados. O número de sessões de banco de dados pode ser maior que o valor da métrica porque o valor da métrica não inclui o seguinte: • Sessões que não têm mais uma conexão de rede, mas que o banco de dados não limpou • Sessões criadas pelo mecanismo de banco de dados para seus próprios propósitos • Sessões criadas pelos recursos de execução paralela do mecanismo de banco de dados • Sessões criadas pelo programador de trabalhos do mecanismo de banco de dados • Conexões do Amazon Aurora	Aurora MySQL e Aurora PostgreSQL	Contagem

Métrica	Descrição	Aplica-se a	Unidades
DDLLatency	A duração média das solicitações, como exemplo, solicitações create (criar), alter (alterar) e drop (descartar).	Aurora MySQL	Milissegundos
DDLThroughput	O número médio de solicitações de DDL por segundo.	Aurora MySQL	Contagem por segundo
Deadlocks	O número médio de deadlocks no banco de dados por segundo.	Aurora MySQL e Aurora PostgreSQL	Contagem por segundo
DeleteLatency	A duração média das operações de exclusão.	Aurora MySQL	Milissegundos
DeleteThroughput	O número médio de consultas de exclusão por segundo.	Aurora MySQL	Contagem por segundo
DiskQueueDepth	O número de solicitações de leitura/gravação pendentes aguardando para acessar o disco.	Aurora MySQL e Aurora PostgreSQL	Contagem
DMLLatency	A duração média das inserções, atualizações e exclusões.	Aurora MySQL	Milissegundos
DMLThroughput	O número médio de inserções, atualizações e exclusões por segundo.	Aurora MySQL	Contagem por segundo

Métrica	Descrição	Aplica-se a	Unidades
`EngineUptime`	A quantidade de tempo em que a instância está em execução.	Aurora MySQL e Aurora PostgreSQL	Segundos
`FreeableMemory`	A quantidade de memória de acesso aleatório disponível. Nos bancos de dados do Aurora MySQL e do Aurora PostgreSQL, essa métrica informa o valor do campo `MemAvailable` de `/proc/meminfo` .	Aurora MySQL e Aurora PostgreSQL	Bytes
`FreeEphemeralStorage`	A quantidade de espaço de armazenamento NVMe Ephemeral disponível.	Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`FreeLocalStorage`	A quantidade de armazenamento local disponível. Diferentemente de outros mecanismos de banco de dados, para instâncias de bancos de dados Aurora, esta métrica informa a quantidade de armazenamento disponível para cada instância de banco de dados. Esse valor depende da classe da instância de banco de dados (para obter informações sobre preços, consulte a Página de preços do Amazon RDS). Você pode aumentar a quantidade de espaço de armazenamento gratuito de uma instância escolhendo uma classe de instância de banco de dados maior para ela. (Isso não se aplica ao Aurora Serverless v2.)	Aurora MySQL e Aurora PostgreSQL	Bytes
`InsertLatency`	A duração média das operações de inserção.	Aurora MySQL	Milissegundos
`InsertThroughput`	A quantidade média de operações de inserção por segundo.	Aurora MySQL	Contagem por segundo

Métrica	Descrição	Aplica-se a	Unidades
`LoginFailures`	O número médio de tentativas falhas de login por segundo.	Aurora MySQL	Contagem por segundo
`MaximumUsedTransactionIDs`	A idade do ID da transação sem vacuum mais antigo, em transações. Se esse valor atingir 2.146.483.648 (2^31 - 1.000.000), o banco de dados será forçado para o modo somente leitura a fim de evitar a conclusão do ID da transação. Para ter mais informações, consulte Evitar falhas de conclusão do ID da transação na documentação do PostgreSQL.	Aurora PostgreSQL	Contagem
`NetworkReceiveThroughput`	A quantidade de throughput de rede recebida dos clientes por instância no cluster de bancos de dados do Aurora. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de bancos de dados Aurora e o volume do cluster.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo (o console mostra Megabytes por segundo)

Métrica	Descrição	Aplica-se a	Unidades
`NetworkThroughput`	O throughput de rede recebido e transmitido aos clientes por instância no cluster de bancos de dados Aurora. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de bancos de dados Aurora e o volume do cluster.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`NetworkTransmitThroughput`	A taxa de transferência de rede enviada aos clientes por cada instância no cluster de banco de dados do Aurora. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de banco de dados do e o volume do cluster.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo (o console mostra Megabytes por segundo)
`NumBinaryLogFiles`	O número de arquivos binlog gerados.	Aurora MySQL	Contagem
`OldestReplicationSlotLag`	O tamanho do atraso da réplica que demora mais em termos de dados de log com gravação antecipada (WAL) recebidos.	Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`PurgeBoundary`	Número da transação até o qual a limpeza do InnoDB é permitida. Se essa métrica não avançar por longos períodos, é uma boa indicação de que a limpeza do InnoDB está bloqueada por transações de longa duração. Para investigar, confira as transações ativas no cluster de banco de dados do Aurora MySQL.	Aurora MySQL versões 2 e 2.11 e posterior Aurora MySQL versões 3, 3.08 e posterior	Contagem
`PurgeFinishedPoint`	Número da transação até o qual a limpeza do InnoDB é realizada. Essa métrica pode ajudar a examinar a rapidez com que a limpeza do InnoDB está progredindo.	Aurora MySQL versões 2 e 2.11 e posterior Aurora MySQL versões 3, 3.08 e posterior	Contagem
`Queries`	O número médio de consultas executadas por segundo.	Aurora MySQL	Contagem por segundo
`RDSToAuroraPostgreSQLReplicaLag`	O atraso ao replicar atualizações da instância principal do RDS para PostgreSQL para outros nós no cluster.	Réplica para Aurora PostgreSQL	Segundos

Métrica	Descrição	Aplica-se a	Unidades
`ReadIOPS`	O número médio de operações de E/S de disco por segundo, com leitura e gravação separadas, em intervalos de um minuto.	Aurora MySQL e Aurora PostgreSQL	Contagem por segundo
`ReadIOPSEphemeralStorage`	O número médio de operações de E/S de leitura de disco no armazenamento NVMe Ephemeral.	Aurora PostgreSQL	Contagem por segundo
`ReadLatency`	O tempo médio necessário por operação de E/S de disco.	Aurora MySQL e Aurora PostgreSQL	Segundos
`ReadLatencyEphemeralStorage`	O tempo médio necessário por operação de I/O de disco para armazenamento NVMe Ephemeral.	Aurora PostgreSQL	Segundos
`ReadThroughput`	O número médio de bytes lidos do disco por segundo.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`ReadThroughputEphemeralStorage`	O número médio de bytes lidos do disco por segundo para armazenamento local.	Aurora PostgreSQL	Bytes por segundo
`ReplicationSlotDiskUsage`	A quantidade de espaço em disco consumido por arquivos de slot de replicação.	Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`ResultSetCacheHitRatio`	A porcentagem de solicitações atendidas pelo cache de conjunto de resultados.	Aurora MySQL versão 2	Porcentagem
`RollbackSegmentHistoryListLength`	Os logs de ações desfeitas que registram transações confirmadas com registros marcados para exclusão. Esses registros estão programados para serem processados pela operação de limpeza do InnoDB.	Aurora MySQL	Contagem
`RowLockTime`	O tempo total gasto adquirindo bloqueios de linha para tabelas do InnoDB.	Aurora MySQL	Milissegundos
`SelectLatency`	A quantidade média de tempo para as operações selecionadas.	Aurora MySQL	Milissegundos
`SelectThroughput`	O número médio de consultas de seleção por segundo.	Aurora MySQL	Contagem por segundo
`ServerlessDatabaseCapacity`	A capacidade atual de um cluster de banco de dados do Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Contagem
`StorageNetworkReceiveThroughput`	A quantidade de taxa de transferência da rede recebida do subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo

Métrica	Descrição	Aplica-se a	Unidades
`StorageNetworkThroughput`	A quantidade de throughput de rede recebida e enviada ao subsistema de armazenamento do Aurora por instância no cluster de banco de dados do Aurora.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`StorageNetworkTransmitThroughput`	A quantidade de throughput de rede enviada ao subsistema de armazenamento do Aurora por instância no cluster de banco de dados do Aurora.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`SumBinaryLogSize`	O tamanho total dos arquivos de log binário.	Aurora MySQL	Bytes
`SwapUsage`	A quantidade de espaço de troca usada. Essa métrica não está disponível para as seguintes classes de instância de banco de dados: • db.r3.*, db.r4.* e db.r7g.* (Aurora MySQL) • db.r7g.* (Aurora PostgreSQL)	Aurora MySQL e Aurora PostgreSQL	Bytes

Métrica	Descrição	Aplica-se a	Unidades
`TempStorageIOPS`	O número de IOPS de leituras e gravações no armazenamento local anexado à instância de banco de dados. Essa métrica representa uma contagem e é medida uma vez por segundo. Essa métrica é aplicável somente para o Aurora Sem Servidor v2.	Aurora MySQL e Aurora PostgreSQL	Contagem por segundo
`TempStorageThroughput`	A quantidade de dados transferidos do armazenamento local associado à instância de banco de dados. Essa métrica é apresentada em bytes e é medida uma vez por segundo. Essa métrica é aplicável somente para o Aurora Sem Servidor v2.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`TransactionAgeMaximum`	A idade da transação em execução ativa mais antiga.	Aurora MySQL versões 3, 3.08 e posterior	Segundos

Métrica	Descrição	Aplica-se a	Unidades
`TransactionLogsDiskUsage`	A quantidade de espaço em disco consumido por logs de transação na instância de bancos de dados Aurora PostgreSQL. Essa métrica é gerada somente quando o Aurora PostgreSQL está usando replicação lógica ou o AWS Database Migration Service. Por padrão, o Aurora PostgreSQL usa registros de log, não logs de transação. Quando os logs de transação não estão em uso, o valor dessa métrica é `-1`.	Principal para Aurora PostgreSQL	Bytes
`TruncateFinishedPoint`	Identificador de transação até o qual o truncamento é desfeito.	Aurora MySQL versões 2 e 2.11 e posterior Aurora MySQL versões 3, 3.08 e posterior	Contagem
`UpdateLatency`	A quantidade média de tempo para as operações de atualização.	Aurora MySQL	Milissegundos
`UpdateThroughput`	A quantidade média de atualizações por segundo.	Aurora MySQL	Contagem por segundo

Métrica	Descrição	Aplica-se a	Unidades
`WriteIOPS`	O número de registros de gravação de armazenamento do Aurora gerados por segundo. Esse é mais ou menos o número de registros de log gerados pelo banco de dados. Eles não correspondem a gravações de página de 8K e não correspondem a pacotes de rede enviados.	Aurora MySQL e Aurora PostgreSQL	Contagem por segundo
`WriteIOPSEphemeralStorage`	O número médio de operações de E/S de gravação de disco no armazenamento NVMe Ephemeral.	Aurora PostgreSQL	Contagem por segundo
`WriteLatency`	O tempo médio necessário por operação de E/S de disco.	Aurora MySQL e Aurora PostgreSQL	Segundos
`WriteLatencyEphemeralStorage`	O tempo médio necessário por operação de I/O de disco para armazenamento NVMe Ephemeral.	Aurora PostgreSQL	Segundos
`WriteThroughput`	A quantidade média de bytes gravados no armazenamento persistente a cada segundo.	Aurora MySQL e Aurora PostgreSQL	Bytes por segundo
`WriteThroughputEphemeralStorage`	O número médio de bytes gravados no disco por segundo para o armazenamento local.	Aurora PostgreSQL	Bytes por segundo

Métricas de uso do Amazon CloudWatch para Amazon Aurora

O namespace `AWS/Usage` no Amazon CloudWatch inclui métricas de uso específico da conta para suas cotas de serviço do Amazon RDS. O CloudWatch coleta métricas de uso automaticamente para todas as Regiões da AWS.

Para ter mais informações, consulte Métricas de uso do CloudWatch no Guia do usuário do Amazon CloudWatch. Para ter mais informações sobre cotas, consulte Cotas e restrições do Amazon Aurora e Solicitar um aumento da cota no Guia do usuário do Service Quotas.

Métrica	Descrição	Unidades*
`AuthorizationsPerDBSecurityGroup`	O número de regras de entrada por grupo de segurança do banco de dados na sua Conta da AWS. O valor usado é o maior número de regras de entrada em um grupo de segurança de banco de dados na conta. Outros grupos de segurança de banco de dados na conta podem ter um número menor de regras de entrada.	Contagem
`CustomEndpointsPerDBCluster`	O número de endpoints personalizados por cluster de banco de dados na sua Conta da AWS. O valor usado é o maior número de endpoints personalizados em um cluster de banco de dados na conta. Outros clusters de banco de dados na conta podem ter um número menor de endpoints personalizados.	Contagem
`DBClusterParameterGroups`	O número máximo de grupos de parâmetros de cluster de banco de dados em sua Conta da AWS. A contagem exclui grupos de parâmetros padrão.	Contagem
`DBClusterRoles`	O número de perfis do AWS Identity and Access Management (IAM) associados por cluster de banco de dados na sua Conta da AWS. O valor usado é o maior número de perfis do IAM associadas a um cluster de banco de dados na conta. Outros clusters de banco de dados na conta podem ter um número menor de perfis do IAM associados.	Contagem

Métrica	Descrição	Unidades*
DBClusters	O número de clusters de banco de dados do Amazon Aurora em sua Conta da AWS.	Contagem
DBInstanceRoles	O número de perfis do AWS Identity and Access Management (IAM) associados por instância de banco de dados na sua Conta da AWS. O valor usado é o maior número de perfis do IAM associadas a uma instância de banco de dados na conta. Outras instâncias de banco de dados na conta podem ter um número menor de perfis do IAM associados.	Contagem
DBInstances	O número de instâncias de banco de dados em sua Conta da AWS.	Contagem
DBParameterGroups	O número de grupos de parâmetros de banco de dados em sua Conta da AWS. A contagem exclui grupos de parâmetros de banco de dados padrão.	Contagem
DBSubnetGroups	O número de grupos de sub-redes de banco de dados em sua Conta da AWS. A contagem exclui o grupo de sub-redes padrão.	Contagem
EventSubscriptions	O número de assinaturas de notificação de eventos em sua Conta da AWS.	Contagem
Integrations	O número de Integrações ETL zero com o Amazon Redshift em sua Conta da AWS.	Contagem
ManualClusterSnapshots	O número de snapshots de cluster de banco de dados criados manualmente em sua Conta da AWS. A contagem exclui snapshots inválidos.	Contagem
OptionGroups	O número de grupos de opções em sua Conta da AWS. A contagem exclui os grupo de opções padrão.	Contagem
Proxies	O número de proxies do RDS na sua conta da AWS.	Contagem

Métrica	Descrição	Unidades*
`ReadReplicasPerMaster`	O número de réplicas de leitura por instância de banco de dados na sua conta. O valor usado é o maior número de réplicas de leitura para uma instância de banco de dados na conta. Outras instâncias de banco de dados na conta podem ter um número menor de réplicas de leitura.	Contagem
`ReservedDBInstances`	O número de instâncias de banco de dados reservadas em sua Conta da AWS. A contagem exclui instâncias desativadas ou recusadas.	Contagem
`SubnetsPerDBSubnetGroup`	O número de sub-redes para por grupo de sub-redes de banco de dados na sua Conta da AWS. O número máximo de sub-redes para um grupo de sub-redes de banco de dados na conta. Outros grupos de sub-redes de banco de dados na conta podem ter um número menor de sub-redes.	Contagem

> **Note**
>
> O Amazon RDS não publica unidades para métricas de uso no CloudWatch. As unidades só aparecem na documentação.

Dimensões do Amazon CloudWatch para o Aurora

Você pode filtrar dados de métricas do Auroramétricas usando qualquer dimensão na tabela a seguir.

Dimensão	Filtra os dados solicitados para . . .
`DBInstanceIdentifier`	Uma instância específica de banco de dados.
`DBClusterIdentifier`	Um cluster específico de banco de dados do Aurora.
`DBClusterIdentifier, Role`	Um cluster específico de banco de dados Aurora, agregando a métrica por função de instância (WRITER/READER). Por

Dimensão	Filtra os dados solicitados para . . .
	exemplo, você pode agregar métricas para todas as instâncias de READER que pertençam a um cluster.
`DbClusterIdentifier, EngineName`	Uma combinação específica de nome de mecanismo e cluster de banco de dados do Aurora. Por exemplo, você pode visualizar a métrica `VolumeReadIOPs` para o cluster `ams1` e o mecanismo `aurora`.
`DatabaseClass`	Todas as instâncias em uma classe de banco de dados. Por exemplo, você pode agregar métricas para todas as instâncias que pertencem à classe do banco de dado `db.r5.large` .
`EngineName`	Apenas o nome do motor identificado. Por exemplo, você pode agrupar métricas para todas as instâncias que têm o nome de mecanismo `aurora-postgresql` .
`SourceRegion`	Apenas a região especificada. Por exemplo, você pode agregar métricas de todas as instâncias de Bancos de Dados na região `us-east-1` .

Disponibilidade de métricas Aurora no console Amazon RDS

Nem todas as métricas fornecidas pelo Amazon Aurora estão disponíveis no console do Amazon RDS. No entanto, você pode vê-las usando outras ferramentas, como a AWS CLI e a API do CloudWatch. Além disso, algumas métricas no console do Amazon RDS são mostradas apenas para classes de instância específicas ou com nomes e unidades de medida diferentes.

Tópicos

- Métricas do Aurora disponíveis na visualização Last Hour (Última hora)
- Métricas Aurora disponíveis em casos específicos
- Métricas do Aurora que não estão disponíveis no console

Métricas do Aurora disponíveis na visualização Last Hour (Última hora)

Você pode visualizar um subconjunto de métricas do Aurora categorizadas na visualização Última hora padrão no console do Amazon RDS. A tabela a seguir lista as categorias e métricas associadas exibidas no console do Amazon RDS para uma instância do Aurora.

Categoria	Métricas

SQL	`ActiveTransactions`
	`BlockedTransactions`
	`BufferCacheHitRatio`
	`CommitLatency`
	`CommitThroughput`
	`DatabaseConnections`
	`DDLLatency`
	`DDLThroughput`
	`Deadlocks`
	`DMLLatency`
	`DMLThroughput`
	`LoginFailures`
	`ResultSetCacheHitRatio`
	`SelectLatency`
	`SelectThroughput`

Sistema	`AuroraReplicaLag` `AuroraReplicaLagMaximum`

Categoria	Métricas

	`AuroraReplicaLagMinimum`
	`CPUCreditBalance`
	`CPUCreditUsage`
	`CPUUtilization`
	`FreeableMemory`
	`FreeLocalStorage` (Isso não se aplica ao Aurora Serverless v2.)
	`NetworkReceiveThroughput`

Implantação	`AuroraReplicaLag`
	`BufferCacheHitRatio`
	`ResultSetCacheHitRatio`
	`SelectThroughput`

Métricas Aurora disponíveis em casos específicos

Além disso, algumas das métricas do Aurora são mostradas apenas para classes de instâncias específicas, ou somente para instâncias de banco de dados, ou com nomes e unidades de medida diferentes:

- As métricas `CPUCreditBalance` e `CPUCreditUsage` são exibidas somente para classes de instância Aurora do `db.t2` MySQL e para classes de instância Aurora `db.t3` do PostgreSQL.
- As seguintes métricas são exibidas com nomes diferentes, conforme listado:

Métrica	Nome de exibição
`AuroraReplicaLagMaximum`	Máximo de atraso da réplica
`AuroraReplicaLagMinimum`	Mínimo de atraso da réplica

Métrica	Nome de exibição
DDLThroughput	DDL
NetworkReceiveThroughput	Taxa de transferência na rede
VolumeBytesUsed	[Faturado] Bytes usados do volume
VolumeReadIOPs	[Faturado] IOPs de leitura do volume
VolumeWriteIOPs	[Faturado] IOPs de gravação do volume

- As seguintes métricas se aplicam a um cluster de banco de dados inteiro do Aurora, mas são exibidas somente durante a visualização de instâncias de banco de dados de um cluster de bancos de dados Aurora no console do Amazon RDS:
  - VolumeBytesUsed
  - VolumeReadIOPs
  - VolumeWriteIOPs
- As métricas a seguir são exibidas em megabytes, em vez de bytes, no console do Amazon RDS:
  - FreeableMemory
  - FreeLocalStorage
  - NetworkReceiveThroughput
  - NetworkTransmitThroughput
- As métricas a seguir se aplicam a um cluster de banco de dados do Aurora PostgreSQL com leituras otimizadas do Aurora:
  - AuroraOptimizedReadsCacheHitRatio
  - FreeEphemeralStorage
  - ReadIOPSEphemeralStorage
  - ReadLatencyEphemeralStorage
  - ReadThroughputEphemeralStorage
  - WriteIOPSEphemeralStorage
  - WriteLatencyEphemeralStorage
  - WriteThroughputEphemeralStorage

Métricas do Aurora que não estão disponíveis no console

As métricas do Aurora a seguir não estão disponíveis no console do Amazon RDS:

- `AuroraBinlogReplicaLag`
- `DeleteLatency`
- `DeleteThroughput`
- `EngineUptime`
- `InsertLatency`
- `InsertThroughput`
- `NetworkThroughput`
- `Queries`
- `UpdateLatency`
- `UpdateThroughput`

Métricas do Amazon CloudWatch para Insights de Performance do Amazon RDS

O Insights de Performance publica automaticamente algumas métricas no Amazon CloudWatch. Os mesmos dados podem ser consultados do Performance Insights, mas ter as métricas no CloudWatch facilita a adição de alarmes do CloudWatch. Também facilita a adição de métricas aos painéis do CloudWatch existentes.

Métrica	Descrição
DBLoad	O número de sessões ativas para o banco de dados. Normalmente, você deseja os dados para o número médio de sessões ativas. No Performance Insights, esses dados são consultados como `db.load.avg` .
DBLoadCPU	O número de sessões ativas em que o tipo do evento de espera é CPU. No Performance Insights, esses dados são consultados como

Métrica	Descrição
	`db.load.avg` , filtrados pelo tipo de evento de espera `CPU`.
DBLoadNonCPU	O número de sessões ativas em que o tipo do evento de espera não é CPU.
DBLoadRelativeToNumVCPUs	A proporção entre a carga de banco de dados e o número de CPUs virtuais para o banco de dados.

> **Note**
>
> Essas métricas serão publicadas no CloudWatch somente se houver carga na instância de banco de dados.

Você pode examinar essas métricas usando o console do CloudWatch, a AWS CLI ou a API do CloudWatch. Você também pode examinar outras métricas de contador do Insights de Performance usando uma função matemática de métrica especial. Para obter mais informações, consulte Consultar outras métricas de contadores do Insights de Performance no CloudWatch.

Por exemplo, você pode obter as estatísticas da métrica `DBLoad` executando o comando get-metric-statistics.

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

Este exemplo gera uma saída semelhante à seguinte.

```
{
```

```
"Datapoints": [
{
"Timestamp": "2021-07-19T21:30:00Z",
"Unit": "None",
"Average": 2.1
},
{
"Timestamp": "2021-07-19T21:34:00Z",
"Unit": "None",
"Average": 1.7
},
{
"Timestamp": "2021-07-19T21:35:00Z",
"Unit": "None",
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


Para obter mais informações sobre o CloudWatch, consulte O que é o Amazon CloudWatch? no Guia do usuário do Amazon CloudWatch.

Consultar outras métricas de contadores do Insights de Performance no CloudWatch

> **Note**
>
> Se você habilitar o modo Avançado do Database Insights, o Amazon RDS publicará métricas de contador do Insights de Performance no Amazon CloudWatch. Com o Database Insights, você não precisa usar a função de matemática métrica `DB_PERF_INSIGHTS`. Você pode usar o painel do CloudWatch Database Insights para pesquisar, consultar e definir alarmes para métricas de contador do Insights de Performance.

É possível consultar, emitir alarmes e criar grafos sobre as métricas do Insights de Performance do RDS pelo CloudWatch. Você pode acessar informações sobre o cluster de banco de dados usando a função matemática de métrica `DB_PERF_INSIGHTS` do CloudWatch. Essa função permite que você use as métricas do Insights de Performance que não são diretamente informadas ao CloudWatch para criar uma série temporal.

É possível usar a nova função matemática de métrica clicando no menu suspenso Adicionar matemática na tela Selecionar métrica no console do CloudWatch. Você pode usá-lo para criar alarmes e grafos sobre as métricas do Insights de Performance ou sobre combinações das métricas do CloudWatch e do Insights de Performance, incluindo alarmes de alta resolução para métricas abaixo de um minuto. Também é possível usar a função programaticamente incluindo a expressão matemática de métrica em uma solicitação `get-metric-data`. Consulte mais informações em Metric math syntax and functions e Create an alarm on Performance Insights counter metrics from an AWS database.

Métricas de contadores do Performance Insights

Métricas de contador são métricas de performance do sistema operacional e do banco de dados no painel do Performance Insights. Para ajudar a identificar e analisar problemas de performance, é possível correlacionar métricas de contadores com a carga de banco de dados. Você deve adicionar uma função estatística à métrica para obter os valores métricos. Por exemplo, as funções compatíveis com a métrica `os.memory.active` são `.avg`, `.min`, `.max`, `.sum` e `.sample_count`.

As métricas do contador são coletadas uma vez por minuto. A coleta de métricas do sistema operacional depende do status do recurso Monitoramento Avançado. Se o recurso estiver desativado, as métricas do sistema operacional serão coletadas uma vez por minuto. Se o recurso estiver ativado, as métricas do sistema operacional serão coletadas para o período selecionado.

Para obter mais informações sobre como ativar ou desativar o recurso Monitoramento Avançado, consulte Ativar e desativar o monitoramento aprimorado.

Tópicos

- Contadores de sistema operacional do Performance Insights
- Contadores do Performance Insights para Aurora MySQL
- Contadores do Performance Insights para Aurora PostgreSQL

Contadores de sistema operacional do Performance Insights

Os contadores de sistema operacional a seguir, que são prefixados com `os`, estão disponíveis para o recurso Insights de Performance no Aurora PostgreSQL e no Aurora MySQL.

Você pode usar a API `ListAvailableResourceMetrics` para obter a lista de métricas de contador disponíveis para sua instância de banco de dados. Para obter mais informações, consulte ListAvailableResourceMetrics no Guia de referência de API do Insights de Performance do Amazon RDS.

Contador	Type	Unidade	Métrica	Descrição
Ativo	Memória	Kilobytes	os.memory .active	A quantidad e de memória atribuída, em kilobytes.
Buffers	Memória	Kilobytes	os.memory .buffers	A quantidade de memória usada para o buffer de solicitações de E/S antes de gravar no dispositivo de armazenamento, em kilobytes.
Em cache	Memória	Kilobytes	os.memory .cached	A quantidad e de memória

Contador	Type	Unidade	Métrica	Descrição
				utilizada para o armazenamento em cache da E/S baseada em sistema de arquivos, em quilobytes.
Cache de banco de dados	Memória	Bytes	os.memory.db.cache	A quantidade de memória usada para o cache de páginas por processo de banco de dados, incluindo tmpfs (shmem), em bytes.
Tamanho do conjunto residente de banco de dados	Memória	Bytes	os.memory.db.residentSetSize	A quantidade de memória usada para o cache anônimo e swap por processo de banco de dados, sem incluir tmpfs (shmem), em bytes.
Swap de banco de dados	Memória	Bytes	os.memory.db.swap	A quantidade de memória usada para swap por processo de banco de dados, em bytes.

Contador	Type	Unidade	Métrica	Descrição
Sujeira	Memória	Kilobytes	os.memory.dirty	A quantidade de páginas de memória na RAM que foram modificadas, mas não gravadas nos blocos de dados relacionados no armazenamento, em kilobytes.
Gratuito	Memória	Kilobytes	os.memory.free	A quantidade de memória não atribuída, em kilobytes.
Páginas enormes livres	Memória	Páginas	os.memory.hugePagesFree	O número de páginas enormes livres. Páginas enormes são um recurso do kernel do Linux.
Páginas enormes reservadas	Memória	Páginas	os.memory.hugePagesRsvd	O número de páginas enormes confirmadas.
Tamanho de páginas enormes	Memória	Kilobytes	os.memory.hugePagesSize	O tamanho de cada unidade de páginas enormes, em kilobytes.

Contador	Type	Unidade	Métrica	Descrição
Páginas enormes surp	Memória	Páginas	os.memory .hugePagesSurp	O número de páginas enormes excedentes disponíveis em comparação com o total.
Total de páginas enormes	Memória	Páginas	os.memory .hugePagesTotal	O número total de páginas enormes.
Inativo	Memória	Kilobytes	os.memory .inactive	A quantidade de páginas de memória usadas com menos frequência, em kilobytes.
Mapeamento	Memória	Kilobytes	os.memory .mapped	A quantidade total de conteúdo do sistema de arquivos que é mapeada na memória dentro de um espaço de endereçamento de processos, em kilobytes.

Contador	Type	Unidade	Métrica	Descrição
Contagem de eliminações por memória insuficiente	Memória	Eliminações	os.memory.outOfMemoryKillCount	O número de encerramentos OOM que aconteceram durante o último intervalo de coleta.
Tabelas de página	Memória	Kilobytes	os.memory.pageTables	A quantidade de memória usada por tabelas de página, em kilobytes.
Slab	Memória	Kilobytes	os.memory.slab	A quantidade de estruturas de dados reutilizáveis do kernel, em kilobytes.
Total	Memória	Kilobytes	os.memory.total	A quantidade total de memória, em kilobytes.
Writeback	Memória	Kilobytes	os.memory.writeback	A quantidade de páginas sujas na RAM que ainda estão sendo gravadas no armazenamento de suporte, em kilobytes.

Contador	Type	Unidade	Métrica	Descrição
Visitas	Utilização da CPU	Porcentagem	os.cpuUtilization.guest	A porcentagem de CPU em uso por programas de convidado.
Idle	Utilização da CPU	Porcentagem	os.cpuUtilization.idle	A porcentagem de CPU que está ociosa.
Irq	Utilização da CPU	Porcentagem	os.cpuUtilization.irq	A porcentagem de CPU em uso por interrupções de software.
Amigável	Utilização da CPU	Porcentagem	os.cpuUtilization.nice	A porcentagem de CPU em uso por programas em execução com a prioridade mais baixa.
Roubo	Utilização da CPU	Porcentagem	os.cpuUtilization.steal	A porcentag em de CPU em uso por outras máquinas virtuais.
Sistema	Utilização da CPU	Porcentagem	os.cpuUtilization.system	A porcentagem de CPU em uso pelo kernel.
Total	Utilização da CPU	Porcentagem	os.cpuUtilization.total	A porcentagem total da CPU em uso. Esse valor inclui o valor amigável.

Contador	Type	Unidade	Métrica	Descrição
Usuário	Utilização da CPU	Porcentagem	os.cpuUtilization.user	A porcentagem de CPU em uso por programas do usuário.
Wait	Utilização da CPU	Porcentagem	os.cpuUtilization.wait	A porcentagem de CPU fora de uso ao aguardar o acesso de E/S.
Bytes rx de armazenamento do Aurora Storage	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.auroraStorageBytesRx	O número de bytes recebidos para o armazenamento do Aurora por segundo.
Bytes tx de armazenamento do Aurora Storage	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.auroraStorageBytesTx	O número de bytes carregados no armazenamento do Aurora por segundo.
Profundidade da fila de discos do Aurora Storage	E/S de disco	Solicitações	os.diskIO.auroraStorage.diskQueueDepth	O tamanho da fila de discos do armazenamento do Aurora.
PS de E/S de leitura do Aurora Storage	E/S de disco	Solicitações por segundo	os.diskIO.auroraStorage.readIOsPS	O número de operações de leitura por segundo.

Contador	Type	Unidade	Métrica	Descrição
Latência de leitura do Aurora Storage	E/S de disco	Milissegundos	os.diskIO.auroraStorage.readLatency	A latência média de uma solicitação de E/S de leitura para o armazenamento do Aurora, em milissegundos.
Throughput de leitura do Aurora Storage	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.readThroughput	A quantidade de taxa de transferência da rede usada por solicitações para o cluster de banco de dados, em bytes por segundo.
PS de E/S de gravação do Aurora Storage	E/S de disco	Solicitações por segundo	os.diskIO.auroraStorage.writeIOsPS	O número de operações de gravação por segundo.
Latência de gravação do Aurora Storage	E/S de disco	Milissegundos	os.diskIO.auroraStorage.writeLatency	A latência média de uma solicitação de E/S de gravação para o armazenamento do Aurora, em milissegundos.

Contador	Type	Unidade	Métrica	Descrição
Throughput de gravação do Aurora Storage	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.writeThroughput	A quantidade de taxa de transferência da rede usada por respostas do cluster de banco de dados, em bytes por segundo.
Comprimento médio da fila de rdstemp	E/S de disco	Solicitações	os.diskIO.rdstemp.avgQueueLen	O número de solicitações que aguardam na fila do dispositivo de E/S.
Tamanho médio de solicitação de rdstemp	E/S de disco	Solicitações	os.diskIO.rdstemp.avgReqSz	O número de solicitações que aguardam na fila do dispositivo de E/S.
Espera de rdstemp	E/S de disco	Milissegundos	os.diskIO.rdstemp.await	O número de milissegundos necessários para responder a solicitações, incluindo o tempo na fila e o tempo de serviço.

Contador	Type	Unidade	Métrica	Descrição
PS de E/S de leitura de rdstemp	E/S de disco	Solicitações	os.diskIO .rdstemp. readIOsPS	O número de operações de leitura por segundo.
KB de leitura de rdstemp	E/S de disco	Kilobytes	os.diskIO .rdstemp.readKb	O número total de kilobytes lidos.
PS de KB de leitura de rdstemp	E/S de disco	Quilobytes/ segundo	os.diskIO .rdstemp. readKbPS	O número de kilobytes lidos por segundo.
PS de rrqm de rdstemp	E/S de disco	Solicitações por segundo	os.diskIO .rdstemp.rrqmPS	O número de solicitaç ões de leitura mescladas enfileiradas por segundo.
TPS de rdstemp	E/S de disco	Transações por segundo	os.diskIO .rdstemp.tps	O número de transações de E/ S por segundo.
Util de rdstemp	E/S de disco	Porcentagem	os.diskIO .rdstemp.util	A porcentagem de tempo de CPU durante o qual as solicitaç ões foram emitidas.
PS de E/S de gravação de rdstemp	E/S de disco	Solicitações por segundo	os.diskIO .rdstemp. writeIOsPS	O número de operações de gravação por segundo.

Contador	Type	Unidade	Métrica	Descrição
KB de gravação de rdstemp	E/S de disco	Kilobytes	os.diskIO .rdstemp.writeKb	O número total de kilobytes gravados.
PS de KB de gravação de rdstemp	E/S de disco	Quilobytes/ segundo	os.diskIO .rdstemp. writeKbPS	O número de kilobytes gravados por segundo.
PS de wrqm de rdstemp	E/S de disco	Solicitações por segundo	os.diskIO .rdstemp. wrqmPS	O número de solicitações de gravação mescladas enfileiradas por segundo.
Bloqueado	Tarefas	Tarefas	os.tasks.blocked	O número de tarefas que estão bloqueadas.
Executar	Tarefas	Tarefas	os.tasks.running	O número de tarefas que estão sendo executadas.
Sleeping	Tarefas	Tarefas	os.tasks.sleeping	O número de tarefas que estão em suspensão.
Interrompida	Tarefas	Tarefas	os.tasks.stopped	O número de tarefas que estão interrompidas.

Contador	Type	Unidade	Métrica	Descrição
Total	Tarefas	Tarefas	os.tasks.total	O número total de tarefas.
Zumbi	Tarefas	Tarefas	os.tasks.zombie	O número de tarefas filho que estão inativas com uma tarefa pai ativa.
Um	Carga média por minuto	Processos	os.loadAverageMinute.one	O número de processos que estão solicitando tempo de CPU no último minuto.
Quinze	Carga média por minuto	Processos	os.loadAverageMinute.fifteen	O número de processos que estão solicitando tempo de CPU nos últimos 15 minutos.
Cinco	Carga média por minuto	Processos	os.loadAverageMinute.five	O número de processos que estão solicitando tempo de CPU nos últimos 5 minutos.
Em cache	Troca	Kilobytes	os.swap.cached	A quantidade de memória de permuta, em kilobytes, usada como a memória cache.

Contador	Type	Unidade	Métrica	Descrição
Gratuito	Troca	Kilobytes	os.swap.free	A quantidade de memória de troca livre, em kilobytes.
Em	Troca	Kilobytes	os.swap.in	A quantidade de memória, em kilobytes , transferida temporariamente do disco.
Saída	Troca	Kilobytes	os.swap.out	A quantidade de memória, em kilobytes , transferida temporariamente para o disco.
Total	Troca	Kilobytes	os.swap.total	A quantidade de memória swap disponível, em kilobytes.
Máximo de arquivos	Sistema de arquivos	Arquivos	os.fileSy s.maxFiles	O número máximo de arquivos que podem ser criados para o sistema de arquivos em todos os volumes de armazenamento.

Contador	Type	Unidade	Métrica	Descrição
Arquivos usados	Sistema de arquivos	Arquivos	os.fileSys.usedFiles	O número de arquivos no sistema de arquivos em todos os volumes de armazenamento.
Porcentagem de arquivos usados	Sistema de arquivos	Arquivos	os.fileSys.usedFilePercent	A porcentagem de arquivos disponíveis em uso em todos os volumes de armazenamento.
Porcentagem usada	Sistema de arquivos	Porcentagem	os.fileSys.usedPercent	A porcentagem do espaço em disco do sistema de arquivos em uso em todos os volumes de armazenamento.
Usado	Sistema de arquivos	Kilobytes	os.fileSys.used	A quantidade de espaço em disco usada pelos arquivos no sistema de arquivos em todos os volumes de armazenamento, em kilobytes.

Contador	Type	Unidade	Métrica	Descrição
Total	Sistema de arquivos	Kilobytes	os.fileSys.total	O espaço total disponível em disco para o sistema de arquivos em todos os volumes de armazenamento, em kilobytes.
Máximo de arquivos	Sistema de arquivos	Arquivos	os.fileSys.<volumeName>.maxFiles	O número máximo de arquivos que podem ser criados para o sistema de armazenamento.
Arquivos usados	Sistema de arquivos	Arquivos	os.fileSys.<volumeName>.usedFiles	O número de arquivos no volume de armazenamento.
Porcentagem de arquivos usados	Sistema de arquivos	Arquivos	os.fileSys.<volumeName>.usedFilePercent	A porcentagem de arquivos disponíveis em uso no volume de armazenamento.
Porcentagem usada	Sistema de arquivos	Porcentagem	os.fileSys.<volumeName>.usedPercent	A porcentagem do espaço em disco do volume de armazenamento em uso.

Contador	Type	Unidade	Métrica	Descrição
Usado	Sistema de arquivos	Kilobytes	os.fileSys.<volumeName>.used	A quantidade de espaço em disco usada pelos arquivos no sistema de armazenamento, em kilobytes.
Total	Sistema de arquivos	Kilobytes	os.fileSys.<volumeName>.total	O espaço total disponível em disco no volume de armazenamento, em kilobytes.
Rx	Rede	Bytes por segundo	os.network.rx	O número de bytes recebidos por segundo.
Tx	Rede	Bytes por segundo	os.network.tx	O número de bytes carregados por segundo.
Utilização de acu	Geral	Porcentagem	os.general.acuUtilization	A porcentagem de capacidade atual da capacidade máxima configurada.

Contador	Type	Unidade	Métrica	Descrição
Configuração máxima de acu	Geral	ACUs	os.general.maxConfiguredAcu	A capacidade máxima configurada pelo usuário, em unidades de capacidade do Aurora (ACUs).
Configuração mínima de acu	Geral	ACUs	os.general.minConfiguredAcu	A capacidade mínima configurada pelo usuário, em ACUs.
Número de vCPUs	Geral	vCPUs	os.general.numVCPUs	O número de CPUs virtuais (vCPUs) para a instância de banco de dados.
Capacidade de banco de dados sem servidor	Geral	ACUs	os.general.serverlessDatabaseCapacity	A capacidade atual da instância, em ACUs.

Contadores do Performance Insights para Aurora MySQL

Os contadores de banco de dados a seguir estão disponíveis para o Performance Insights para o Aurora MySQL.

Tópicos

- Contadores nativos para o Aurora MySQL
- Contadores não nativos para o Aurora MySQL

Contadores nativos para o Aurora MySQL

As métricas nativas são definidas pelo mecanismo de banco de dados e não pelo Amazon Aurora. Você pode encontrar definições para essas métricas nativas em Variáveis de status do servidor, na documentação do MySQL.

Contador	Type	Unidade	Métrica
Com_analyze	SQL	Consultas por segundo	db.SQL.Com_analyze
Com_optimize	SQL	Consultas por segundo	db.SQL.Com_optimize
Com_select	SQL	Consultas por segundo	db.SQL.Com_select
Innodb_rows_deleted	SQL	Linhas por segundo	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	Linhas por segundo	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	Linhas por segundo	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	Linhas por segundo	db.SQL.Innodb_rows_updated
Consultas	SQL	Consultas por segundo	db.SQL.Queries
Perguntas	SQL	Consultas por segundo	db.SQL.Questions

Contador	Type	Unidade	Métrica
Select_full_join	SQL	Consultas por segundo	db.SQL.Select_full_join
Select_full_range_join	SQL	Consultas por segundo	db.SQL.Select_full_range_join
Select_range	SQL	Consultas por segundo	db.SQL.Select_range
Select_range_check	SQL	Consultas por segundo	db.SQL.Select_range_check
Select_scan	SQL	Consultas por segundo	db.SQL.Select_scan
Slow_queries	SQL	Consultas por segundo	db.SQL.Slow_queries
Sort_merge_passes	SQL	Consultas por segundo	db.SQL.Sort_merge_passes
Sort_range	SQL	Consultas por segundo	db.SQL.Sort_range
Sort_rows	SQL	Consultas por segundo	db.SQL.Sort_rows

Contador	Type	Unidade	Métrica
Sort_scan	SQL	Consultas por segundo	db.SQL.Sort_scan
Total_query_time	SQL	Milissegundos	db.SQL.Total_query_time
Table_locks_immediate	Travas	Solicitações por segundo	db.Locks.Table_locks_immediate
Table_locks_waited	Travas	Solicitações por segundo	db.Locks.Table_locks_waited
Innodb_row_lock_time	Travas	Milissegundos (média)	db.Locks.Innodb_row_lock_time
Aborted_clients	Usuários	Conexões	db.Users.Aborted_clients
Aborted_connects	Usuários	Conexões	db.Users.Aborted_connects
Conexões	Usuários	Conexões	db.Users.Connections
External_threads_connected	Usuários	Conexões	db.Users.External_threads_connected
max_connections	Usuários	Conexões	db.Users.max_connections
Threads_connected	Usuários	Conexões	db.Users.Threads_connected
Threads_created	Usuários	Conexões	db.Users.Threads_created
Threads_running	Usuários	Conexões	db.Users.Threads_running
Created_tmp_disk_tables	Temporário	Tabelas por segundo	db.Temp.Created_tmp_disk_tables

Contador	Type	Unidade	Métrica
Created_tmp_tables	Temporário	Tabelas por segundo	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Cache	Páginas	db.Cache.Innodb_buffer_pool_pages_data
Innodb_buffer_pool_pages_total	Cache	Páginas	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	Páginas por segundo	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Cache	Páginas por segundo	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	Tabelas	db.Cache.Opened_tables
Opened_table_definitions	Cache	Tabelas	db.Cache.Opened_table_definitions
Qcache_hits	Cache	Consultas	db.Cache.Qcache_hits

Contadores não nativos para o Aurora MySQL

Métricas de contador não nativas são contadores definidos pelo Amazon RDS. Uma métrica não nativa pode ser uma métrica obtida com uma consulta específica. Uma métrica não nativa também pode ser uma métrica derivada, em que dois ou mais contadores nativos são usados em cálculos para proporções, taxas de ocorrência ou latências.

Contador	Type	Unidad	Métrica	Descrição	Definição
active_transactions	Transa s	db.Trar ctions.a tive_tra saction	O total de transaçõe s ativas.	`SELECT COUNT(1) AS active_tr ansactions`	

Contador	Type	Unidad	Métrica	Descrição	Definição
				`FROM INFOR MATION_SC HEMA.INNO DB_TRX`	
innodb_buffer_pool _hit_rate	Cache	db.Cac innoDB ffer_poo _hit_rat	A porcentag em de leituras que o InnoDB pode atender no pool de buffer.	`100 * innodb_bu ffer_pool _read_req uests / (innodb_ buffer_po ol_read_r equests + innodb_bu ffer_pool _reads)`	
innodb_buffer_pool _hits	Cache	Página por segund	db.Cache. innoDB_bu ffer_pool _hits	O número de leituras que o InnoDB pode atender no pool de buffer.	`innodb_bu ffer_pool _read_requests - innodb_bu ffer_pool _reads`

Contador	Type	Unidad	Métrica	Descrição	Definição
innodb_buffer_pool_usage	Cache	Porcen em	db.Cache.innoDB_buffer_pool_usage	A porcentagem do pool de buffers do InnoDB que contém dados (páginas). **Note** Ao usar tabelas compactadas, esse valor pode variar. Para obter mais informações, consulte as informações sobre `Innodb_buffer_pool_pages_data` e `Innodb_buffer_pool_pages_total` em Variáveis de status do servidor, na documentação do MySQL.	`Innodb_buffer_pool_pages_data / Innodb_buffer_pool_pages_total * 100.0`

Contador	Type	Unidad	Métrica	Descrição	Definição
innodb_deadlocks	Travas	db.Lock innodb_ adlocks	O número total de deadlocks.	`SELECT COUNT AS innodb_de adlocks FROM INFORMATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_dea dlocks'`	
innodb_lo ck_timeouts	Travas	db.Lock innodb_ ck_time ts	O número total de deadlocks que expiraram.	`SELECT COUNT AS innodb_lo ck_timeouts FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_tim eouts'`	
innodb_ro w_lock_waits	Travas	db.Lock innodb_ w_lock_ its	O número total de bloqueios de linha que resultara m em uma espera.	`SELECT COUNT AS innodb_ro w_lock_waits FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME ='lock_ro w_lock_waits'`	

Contador	Type	Unidad	Métrica	Descrição	Definição
innodb_rows_changed	SQL	db.SQL nodb_r _chang	O total de operações de linhas do InnoDB.	`db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated`	
query_cache_hit_rate	Cache	Porcen em	db.Cache.query_cache_hit_rate	A taxa de acertos para o cache do conjunto de resultados do MySQL (cache de consulta).	`Qcache_hits / (QCache_hits + Com_select) * 100`
temp_disk_tables_percent	Tempo o	db.Tem emp_di tables_ rcent	A porcentagem de tabelas temporárias criadas no disco pelo servidor ao executar declarações.	`(db.Temp.Created_tmp_disk_tables / db.Temp.Created_tmp_tables) * 100`	

Contador	Type	Unidad	Métrica	Descrição	Definição
trx_rseg_history_len	Transa s	Nenhur	db.Transactions.trx_rseg_history_len	Uma lista das páginas de undo log de transações confirmadas que é mantida pelo sistema de transações InnoDB para implementar o controle de simultaneidade de várias versões. Para obter mais informações sobre os detalhes de registros de undo log, consulte https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html na documentação do MySQL.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Contadores do Performance Insights para Aurora PostgreSQL

Os contadores de banco de dados a seguir estão disponíveis para o Performance Insights para Aurora PostgreSQL.

Tópicos

- Contadores nativos para o Aurora PostgreSQL
- Contadores não nativos para o Aurora PostgreSQL

Contadores nativos para o Aurora PostgreSQL

As métricas nativas são definidas pelo mecanismo de banco de dados e não pelo Amazon Aurora. É possível encontrar definições dessas métricas em Visualizar estatísticas (em inglês) na documentação do PostgreSQL.

Contador	Type	Unidade	Métrica
tup_deleted	SQL	Tuplas por segundo	db.SQL.tup_deleted
tup_fetched	SQL	Tuplas por segundo	db.SQL.tup_fetched
tup_inserted	SQL	Tuplas por segundo	db.SQL.tup_inserted
tup_returned	SQL	Tuplas por segundo	db.SQL.tup_returned
tup_updated	SQL	Tuplas por segundo	db.SQL.tup_updated
blks_hit	Cache	Blocos por segundo	db.Cache.blks_hit
buffers_alloc	Cache	Blocos por segundo	db.Cache.buffers_alloc
buffers_checkpoint	Ponto de verificação	Blocos por segundo	db.Checkpoint.buffers_checkpoint
checkpoints_req	Ponto de verificação	Pontos de verificação por minuto	db.Checkpoint.checkpoints_req
checkpoint_sync_time	Ponto de verificação	Milissegundos por ponto de verificação	db.Checkpoint.checkpoint_sync_time
checkpoints_timed	Ponto de verificação	Pontos de verificação por minuto	db.Checkpoint.checkpoints_timed
checkpoint_write_time	Ponto de verificação	Milissegundos por ponto de verificação	db.Checkpoint.checkpoint_write_time

Contador	Type	Unidade	Métrica
maxwritten_clean	Ponto de verificação	Paradas de limpeza de Bgwriter por minuto	db.Checkpoint.maxwritten_clean
deadlocks	Simultaneidade	Deadlocks por minuto	db.Concurrency.deadlocks
blk_read_time	I/O	Milissegundos	db.IO.blk_read_time
blks_read	I/O	Blocos por segundo	db.IO.blks_read
buffers_backend	I/O	Blocos por segundo	db.IO.buffers_backend
buffers_backend_fsync	I/O	Blocos por segundo	db.IO.buffers_backend_fsync
buffers_clean	I/O	Blocos por segundo	db.IO.buffers_clean
temp_bytes	Temporário	Bytes por segundo	db.Temp.temp_bytes
temp_files	Temporário	Arquivos por minuto	db.Temp.temp_files
xact_commit	Transações	Confirmações por segundo	db.Transactions.xact_commit
xact_rollback	Transações	Reversões por segundo	db.Transactions.xact_rollback
numbackends	Usuário	Conexões	db.User.numbackends
archived_count	WAL	Arquivos por minuto	db.WAL.archived_count

Contadores não nativos para o Aurora PostgreSQL

Métricas de contadores não nativos são contadores definidos pelo Amazon Aurora. Uma métrica não nativa pode ser uma métrica obtida com uma consulta específica. Uma métrica não nativa também pode ser uma métrica derivada, em que dois ou mais contadores nativos são usados em cálculos para proporções, taxas de ocorrência ou latências.

Contador	Type	Unidade	Métrica	Descrição	Definição
checkpoint_sync_latency	Ponto de verificação	Milissegundos	db.Checkpoint.checkpoint_sync_latency	O tempo que foi gasto na parte do processamento de ponto de verificação em que os arquivos são sincronizados no disco.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Ponto de verificação	Milissegundos	db.Checkpoint.checkpoint_write_latency	O tempo que foi gasto na parte do processamento de ponto de verificação em que os arquivos são gravados no disco.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`
local_blks_read	E/S	Blocos	db.io.local_BLKS_read	Número total de blocos locais lidos.	Não aplicável
local_blk_read_time	E/S	Milissegundos	db.io.local_BLK_Read_Time	Se `track_io_timing` estiver ativado, ele rastreia o tempo total gasto lendo blocos de arquivos de dados locais, em milissegundos, caso contrário, o valor será zero.	Não aplicável

Contador	Type	Unidade	Métrica	Descrição	Definição
				Para obter mais informações, consulte track_io_timing.	
num_blocked_sessions	Travas	db.Locks.num_blocked_sessions	O número de sessões bloqueadas.	–	
orcache_blks_hit	E/S	Consultas	db.io.orcache_blks_hit	Número total de blocos compartilhados acessados pelo cache de leituras otimizado.	Não aplicável
ou cache_blk_read_time	E/S	Milissegundos	db.io.ORCache_BLK_Read_Time	Se `track_io_timing` estiver ativado, ele rastreia o tempo total gasto lendo blocos de arquivos de dados do cache de leituras otimizadas, em milissegundos, caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.	Não aplicável
read_latency	E/S	Milissegundos	db.IO.read_latency	O tempo gasto lendo blocos de arquivos de dados pelos backends nesta instância.	`blk_read_time / blks_read`

Contador	Type	Unidade	Métrica	Descrição	Definição
armazenamento_blocos_read	E/S	Blocos	db.io.STORAGE_BLK_READ	Número total de blocos compartilhados lidos do armazenamento aurora.	Não aplicável
tempo de leitura do bloco de armazenamento	E/S	Milissegundos	db.io.STORAGE_BLK READ_TIM	Se `track_io_timing` estiver ativado, ele rastreia o tempo total gasto lendo blocos de arquivos de dados do armazenamento do Aurora, em milissegundos, caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.	Não aplicável
num_blocked_sessions	Travas	db.Locks.num_blocked_sessions	O número de sessões bloqueadas.	–	
active_count	Estado	Sessões	db.state.active_count	O número de sessões no estado `active`.	Não aplicável
idle_count	Estado	Sessões	db.state.idle_count	O número de sessões no estado `idle`.	Não aplicável

Contador	Type	Unidade	Métrica	Descrição	Definição
idle_in_transaction_aborted_count	Estado	Sessões	db.state.idle_in_transaction_aborted_count	O número de sessões no estado `idle in transaction (aborted)` .	Não aplicável
idle_in_transaction_count	Estado	Sessões	db.state.idle_in_transaction_count	O número de sessões no estado `idle in transaction` .	Não aplicável
idle_in_transaction_max_time	Estado	Segundos	db.state.idle_in_transaction_max_time	A duração da transação mais longa no estado `idle in transaction` , em segundos.	Não aplicável
logical_reads	SQL	Blocos	db.SQL.logical_reads	O número total de blocos atingidos e lidos.	`blks_hit + blks_read`
queries_started	SQL	Consultas	db.SQL.queries	O número de consultas iniciadas.	Não aplicável
queries_finished	SQL	Consultas	db.SQL.queries	O número de consultas realizadas.	Não aplicável
total_query_time	SQL	Milissegundos	db.SQL.total_query_time	O tempo total decorrido na execução de declarações, em milissegundos.	Não aplicável

Contador	Type	Unidade	Métrica	Descrição	Definição
active_transactions	Transações	Transações	db.Transactions.active_transactions	O número de transações ativas.	Não aplicável
blocked_transactions	Transações	Transações	db.Transactions.blocked_transactions	O número de transações bloqueadas.	Não aplicável
commit_latency	Transações	Microssegundos	db.Transactions.commit_latency	A duração média das operações de commit.	`db.Transactions.duration_commits / db.Transactions.xact_commit`
duration_commits	Transações	Milissegundos	db.Transactions.duration_commits	O tempo total da transação gasto no último minuto, em milissegundos.	Não aplicável
max_used_xact_ids	Transações	Transações	db.Transactions.max_used_xact_ids	O número de transações que não foram apagadas.	Não aplicável
oldest_inactive_logical_replication_slot_xid_age	Transações	Tamanho	db.Transactions.oldest_inactive_logical_replication_slot_xid_age	A idade da transação mais antiga em um slot de replicação lógica inativo.	Não aplicável

Contador	Type	Unidade	Métrica	Descrição	Definição
oldest_active_logical_replication_slot_xid_age	Transações	Tamanho	db.Transactions.oldest_active_logical_replication_slot_xid_age	A idade da transação mais antiga em um slot de replicação lógica ativo.	Não aplicável
oldest_reader_feedback_xid_age	Transações	Tamanho	db.Transactions.oldest_reader_feedback_xid_age	A idade da transação mais antiga de uma transação de longa duração em uma instância de leitor do Aurora ou em uma instância de leitor de banco de dados global do Aurora.	Não aplicável
oldest_prepared_transaction_xid_age	Transações	Tamanho	db.transactions.oldest_prepared_transaction_xid_age	A idade da transação preparada mais antiga.	Não aplicável
oldest_running_transaction_xid_age	Transações	Tamanho	db.Transactions.oldest_running_transaction_xid_age	A idade da transação em execução mais antiga.	Não aplicável

Contador	Type	Unidade	Métrica	Descrição	Definição
max_connections	Usuários	Usuários	db.User.max_connections	O número máximo de conexões permitido para um banco de dados, conforme configurado no parâmetro `max_connections` .	Não aplicável
total_auth_attempts	Usuários	Usuários	db.User.total_auth_attempts	O número de tentativas de conexão com essa instância.	Não aplicável
archive_failed_count	WAL	Arquivos por minuto	db.WAL.archive_failed_count	O número de tentativas malsucedidas de arquivamento de arquivos WAL, em arquivos por minuto.	Não aplicável

Estatísticas SQL para Performance Insights

Estatísticas SQL são métricas de performance sobre consultas SQL coletadas pelo Performance Insights. O Performance Insights reúne estatísticas para cada segundo em que uma consulta está sendo executada e para cada chamada SQL. As estatísticas do SQL são uma média para o intervalo de tempo selecionado.

Um resumo SQL é um composto de todas as consultas com determinado padrão, mas não necessariamente com os mesmos valores literais. O resumo substitui valores literais por um ponto de interrogação. Por exemplo, `SELECT * FROM emp WHERE lname= ?`. Esse resumo pode consistir nas seguintes consultas subordinadas:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Todos os mecanismos são compatíveis com estatísticas de consultas de resumo.

Para receber informações sobre a compatibilidade da região, do mecanismo de banco de dados e da classe de instância com esse atributo, consulte Mecanismo de banco de dados do Amazon Aurora, região e classe de instância compatíveis com recursos do Insights de Performance.

Tópicos

- Estatísticas SQL para o Aurora MySQL
- Estatísticas SQL do Aurora PostgreSQL

Estatísticas SQL para o Aurora MySQL

O Aurora MySQL coleta estatísticas SQL somente em nível de resumo. Nenhuma estatística é mostrada em nível de instrução.

Tópicos

- Estatísticas de resumo para Aurora MySQL
- Estatísticas por segundo para o Aurora MySQL
- Estatísticas por chamada para o Aurora MySQL
- Principais estatísticas para Aurora MySQL

Estatísticas de resumo para Aurora MySQL

O Performance Insights coleta estatísticas de resumo do SQL da tabela `events_statements_summary_by_digest`. A tabela `events_statements_summary_by_digest` é gerenciada pelo seu banco de dados.

Ela não tem uma política de evicção. Quando a tabela estiver cheia, o Console de gerenciamento da AWS mostrará a seguinte mensagem:

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

Nessa situação, o Aurora MySQL não rastreia consultas SQL. Para resolver esse problema, o Performance Insights trunca automaticamente a tabela de resumo quando ambas as condições são satisfeitas:

- A tabela está cheia.

- O Performance Insights gerencia o Performance Schema automaticamente.

  Para gerenciamento automático, o parâmetro `performance_schema` deve ser definido como `0` e Source (Origem) não deve ser definido como `user`. Se o Performance Insights não estiver gerenciando o Performance Schema automaticamente, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Na AWS CLI, verifique a origem de um valor de parâmetro executando o comando describe-db-parameters.

Estatísticas por segundo para o Aurora MySQL

As seguintes estatísticas do SQL estão disponíveis para clusters de banco de dados do Aurora MySQL

Métrica	Unidade
db.sql_tokenized.stats.count_star_per_sec	Chamadas por segundo
db.sql_tokenized.stats.sum_timer_wait_per_sec	Latência média por segundo (em ms)
db.sql_tokenized.stats.sum_select_full_join_per_sec	Selecionar junção completa por segundo
db.sql_tokenized.stats.sum_select_range_check_per_sec	Selecionar verificação de intervalo por segundo
db.sql_tokenized.stats.sum_select_scan_per_sec	Selecionar verificação por segundo
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	Classificar passagens de mesclagem por segundo
db.sql_tokenized.stats.sum_sort_scan_per_sec	Classificar verificações por segundo
db.sql_tokenized.stats.sum_sort_range_per_sec	Classificar intervalos por segundo
db.sql_tokenized.stats.sum_sort_rows_per_sec	Classificar linhas por segundo

Métrica	Unidade
db.sql_tokenized.stats.sum_rows_affected_per_sec	Linhas afetadas por segundo
db.sql_tokenized.stats.sum_rows_examined_per_sec	Linhas examinadas por segundo
db.sql_tokenized.stats.sum_rows_sent_per_sec	Linhas enviadas por segundo
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	Tabelas de disco temporárias criadas por segundo
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	Tabelas temporárias criadas por segundo
db.sql_tokenized.stats.sum_lock_time_per_sec	Tempo de bloqueio por segundo (em ms)

Estatísticas por chamada para o Aurora MySQL

As métricas a seguir fornecem estatísticas por chamada para uma instrução SQL.

Métrica	Unidade
db.sql_tokenized.stats.sum_timer_wait_per_call	Latência média por chamada (em ms)
db.sql_tokenized.stats.sum_select_full_join_per_call	Selecionar junções completas por chamada
db.sql_tokenized.stats.sum_select_range_check_per_call	Selecionar verificação de intervalo por chamada
db.sql_tokenized.stats.sum_select_scan_per_call	Selecionar verificações por chamada
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	Classificar passagens de mesclagem por chamada
db.sql_tokenized.stats.sum_sort_scan_per_call	Classificar verificações por chamada

Métrica	Unidade
db.sql_tokenized.stats.sum_sort_range_per_call	Classificar intervalos por chamada
db.sql_tokenized.stats.sum_sort_rows_per_call	Classificar linhas por chamada
db.sql_tokenized.stats.sum_rows_affected_per_call	Linhas afetadas por chamada
db.sql_tokenized.stats.sum_rows_examined_per_call	Linhas examinadas por chamada
db.sql_tokenized.stats.sum_rows_sent_per_call	Linhas enviadas por chamada
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	Tabelas de disco temporárias criadas por chamada
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	Tabelas temporárias criadas por chamada
db.sql_tokenized.stats.sum_lock_time_per_call	Tempo de leitura por chamada (em ms)

Principais estatísticas para Aurora MySQL

As seguintes estatísticas do SQL estão disponíveis para clusters de banco de dados do Aurora MySQL

Métrica	Unidade
db.sql_tokenized.stats.count_star	Calls
db.sql_tokenized.stats.sum_timer_wait	Tempo de espera (em ms)
db.sql_tokenized.stats.sum_select_full_join	Selecionar junção completa
db.sql_tokenized.stats.sum_select_range_check	Selecionar verificações de intervalo
db.sql_tokenized.stats.sum_select_scan	Selecionar verificações

Métrica	Unidade
db.sql_tokenized.stats.sum_sort_merge_passes	Classificar passagens de mesclagem
db.sql_tokenized.stats.sum_sort_scan	Classificar verificações
db.sql_tokenized.stats.sum_sort_range	Classificar intervalos
db.sql_tokenized.stats.sum_sort_rows	Classificar linhas
db.sql_tokenized.stats.sum_rows_affected	Linhas afetadas
db.sql_tokenized.stats.sum_rows_examined	Linhas examinadas
db.sql_tokenized.stats.sum_rows_sent	Linhas enviadas
db.sql_tokenized.stats.sum_created_tmp_disk_tables	Tabelas de disco temporárias criadas
db.sql_tokenized.stats.sum_created_tmp_tables	Tabelas temporárias criadas
db.sql_tokenized.stats.sum_lock_time	Tempo de bloqueio (em ms)

Estatísticas SQL do Aurora PostgreSQL

Para cada chamada SQL e para cada segundo em que uma consulta é executada, o Performance Insights coleta estatísticas SQL. Todos os mecanismos do Aurora coletam estatísticas somente em nível de resumo.

A seguir, você pode encontrar informações sobre estatísticas no nível do resumo para Aurora PostgreSQL.

Tópicos

- Estatísticas de resumo do AuroraPostgreSQL
- Estatísticas de resumo por segundo para o Aurora PostgreSQL
- Estatísticas de resumo por chamada para o Aurora PostgreSQL
- Principais estatísticas do Aurora PostgreSQL

Estatísticas de resumo do AuroraPostgreSQL

Para visualizar estatísticas do SQL Digest, a biblioteca `pg_stat_statements` deve estar carregada. Essa biblioteca é carregada por padrão para clusters de banco de dados Aurora PostgreSQL que são compatíveis com PostgreSQL 10. Você deve habilitar essa biblioteca manualmente para clusters de banco de dados Aurora PostgreSQL compatíveis com o PostgreSQL 9.6. Para ativá-la manualmente, adicione `pg_stat_statements` a `shared_preload_libraries` no grupo de parâmetros de banco de dados associado à instância de banco de dados. Em seguida, reinicialize sua instância de banco de dados. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.

> **Note**
>
> O Performance Insights só pode coletar estatísticas em relação a consultas não truncadas em `pg_stat_activity`. Por padrão, os bancos de dados PostgreSQL truncam consultas com mais de 1.024 bytes. Para aumentar o tamanho das consultas, altere o parâmetro `track_activity_query_size` no grupo de parâmetros de banco de dados associado à sua instância de banco de dados. Ao alterar esse parâmetro, uma reinicialização da instância de banco de dados é necessária.

Estatísticas de resumo por segundo para o Aurora PostgreSQL

As seguintes estatísticas de resumo SQL estão disponíveis para instância de banco de dados Aurora PostgreSQL.

Métrica	Unidade
db.sql_tokenized.stats.calls_per_sec	Chamadas por segundo
db.sql_tokenized.stats.rows_per_sec	Linhas por segundo
db.sql_tokenized.stats.total_time_per_sec	Média de execuções ativas por segundo (AAE)
db.sql_tokenized.stats.shared_blks_hit_per_sec	Acertos de bloco por segundo
db.sql_tokenized.stats.shared_blks_read_per_sec	Leituras de bloco por segundo

Métrica	Unidade
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	Blocos sujos por segundo
db.sql_tokenized.stats.shared_blks_written_per_sec	Gravações de bloco por segundo
db.sql_tokenized.stats.local_blks_hit_per_sec	Acertos de bloco local por segundo
db.sql_tokenized.stats.local_blks_read_per_sec	Leituras de bloco local por segundo
db.sql_tokenized.stats.local_blks_dirtied_per_sec	Bloco local sujo por segundo
db.sql_tokenized.stats.local_blks_written_per_sec	Gravações de bloco local por segundo
db.sql_tokenized.stats.temp_blks_written_per_sec	Gravações temporárias por segundo
db.sql_tokenized.stats.temp_blks_read_per_sec	Leituras temporárias por segundo
db.sql_tokenized.stats.blk_read_time_per_sec	Leituras simultâneas médias por segundo
db.sql_tokenized.stats.blk_write_time_per_sec	Gravações simultâneas médias por segundo

Estatísticas de resumo por chamada para o Aurora PostgreSQL

As métricas a seguir fornecem estatísticas por chamada para uma instrução SQL.

Métrica	Unidade
db.sql_tokenized.stats.rows_per_call	Linhas por chamada
db.sql_tokenized.stats.avg_latency_per_call	Latência média por chamada (em ms)
db.sql_tokenized.stats.shared_blks_hit_per_call	Acertos de bloco por chamada

Métrica	Unidade
db.sql_tokenized.stats.shared_blks_read_per_call	Leituras de bloco por chamada
db.sql_tokenized.stats.shared_blks_written_per_call	Gravações de bloco por chamada
db.sql_tokenized.stats.shared_blks_dirtied_per_call	Blocos sujos por chamada
db.sql_tokenized.stats.local_blks_hit_per_call	Acertos de bloco local por chamada
db.sql_tokenized.stats.local_blks_read_per_call	Leituras de bloco local por chamada
db.sql_tokenized.stats.local_blks_dirtied_per_call	Bloco local sujo por chamada
db.sql_tokenized.stats.local_blks_written_per_call	Gravações de bloco local por chamada
db.sql_tokenized.stats.temp_blks_written_per_call	Gravações temporárias de bloco por chamada
db.sql_tokenized.stats.temp_blks_read_per_call	Leituras temporárias de bloco por chamada
db.sql_tokenized.stats.blk_read_time_per_call	Tempo de leitura por chamada (em ms)
db.sql_tokenized.stats.blk_write_time_per_call	Tempo de gravação por chamada (em ms)

Principais estatísticas do Aurora PostgreSQL

As estatísticas de SQL a seguir estão disponíveis para instâncias de banco de dados do Aurora PostgreSQL.

Métrica	Unidade
db.sql_tokenized.stats.calls	Calls

Métrica	Unidade
db.sql_tokenized.stats.rows	Linhas
db.sql_tokenized.stats.total_time	Tempo total (em ms)
db.sql_tokenized.stats.shared_blks_hit	Acertos de bloco
db.sql_tokenized.stats.shared_blks_read	Leituras de bloco
db.sql_tokenized.stats.shared_blks_dirtied	Blocos sujos
db.sql_tokenized.stats.shared_blks_written	Gravações de bloco
db.sql_tokenized.stats.local_blks_hit	Acertos de bloco local
db.sql_tokenized.stats.local_blks_read	Leituras de bloco local
db.sql_tokenized.stats.local_blks_dirtied	Blocos locais sujos
db.sql_tokenized.stats.local_blks_written	Gravações de bloco local
db.sql_tokenized.stats.temp_blks_written	Gravações temporárias
db.sql_tokenized.stats.temp_blks_read	Leituras temporárias
db.sql_tokenized.stats.blk_read_time	Média de leituras simultâneas (em ms)
db.sql_tokenized.stats.blk_write_time	Média de gravações simultâneas (em ms)

Para obter mais informações sobre essas métricas, consulte pg_stat_statements na documentação do PostgreSQL.

Métricas do sistema operacional no monitoramento avançado

O Amazon Aurora fornece métricas em tempo real para o sistema operacional (SO) no qual seu cluster de de banco de dados é executado. O Aurora fornece as métricas do monitoramento avançado à sua conta do Amazon CloudWatch Logs. As tabelas a seguir listam métricas de SO disponíveis usando o Amazon CloudWatch Logs.

Tópicos

- Métricas do SO para Aurora

Métricas do SO para Aurora

Grupo	Métrica	Nome do console	Descrição
General	engine	Não aplicável	O mecanismo de banco de dados para a instância de banco de dados.
	instanceID	Não aplicável	O DB instance identifier.
	instanceResourceID	Não aplicável	Um identificador imutável para a instância de Bancos de Dados que é exclusivo para uma região da AWS, também usado como o identificador de stream de logs.
	numVCPUs	Não aplicável	O número de CPUs virtuais para a instância de banco de dados.
	timestamp	Não aplicável	A hora em que as métricas foram obtidas.
	uptime	Não aplicável	Por quanto tempo a instância de banco de dados esteve ativa.
	version	Não aplicável	A versão do formato JSON do stream de métricas do sistema operacional.
cpuUtilization	guest	Convidado da CPU	A porcentagem de CPU em uso por programas de convidado.
	idle	CPU ociosa	A porcentagem de CPU que está ociosa.
	irq	IRQ da CPU	A porcentagem de CPU em uso por interrupções de software.

Grupo	Métrica	Nome do console	Descrição
	`nice`	CPU boa	A porcentagem de CPU em uso por programas em execução com a prioridade mais baixa.
	`steal`	Roubo de CPU	A porcentagem de CPU em uso por outras máquinas virtuais.
	`system`	Sistema de CPU	A porcentagem de CPU em uso pelo kernel.
	`total`	Total da CPU	A porcentagem total da CPU em uso. Esse valor inclui o valor de `nice`.
	`user`	Usuário da CPU	A porcentagem de CPU em uso por programas do usuário.
	`wait`	Espera da CPU	A porcentagem de CPU fora de uso ao aguardar o acesso de E/S.
`diskIO`	`avgQueueLen`	Tamanho médio da fila	O número de solicitações que aguardam na fila do dispositivo de E/S.
	`avgReqSz`	Tamanho médio da solicitação	O tamanho médio da solicitação, em kilobytes.
	`await`	Espera de E/S de disco	O número de milissegundos necessários para responder a solicitações, incluindo o tempo na fila e o tempo de serviço.
	`device`	Não aplicável	O identificador do dispositivo de disco em uso.
	`readIOsPS`	E/Ss de leitura	O número de operações de leitura por segundo.

Grupo	Métrica	Nome do console	Descrição
	`readKb`	Total de leitura	O número total de kilobytes lidos.
	`readKbPS`	Kb/s de leitura	O número de kilobytes lidos por segundo.
	`readLatency`	Latência de leitura	O tempo decorrido entre o envio de uma solicitação de E/S de leitura e sua conclusão, em milissegundos. Essa métrica só está disponível para o Amazon Aurora.
	`readThroughput`	Taxa de transferência de leitura	A quantidade de taxa de transferência da rede usada por solicitações para o cluster de banco de dados, em bytes por segundo. Essa métrica só está disponível para o Amazon Aurora.
	`rrqmPS`	Rrqms	O número de solicitações de leitura mescladas enfileiradas por segundo.
	`tps`	TPS	O número de transações de E/S por segundo.
	`util`	Utilização de E/S de disco	A porcentagem de tempo de CPU durante o qual as solicitações foram emitidas.
	`writeIOsPS`	E/Ss de gravação	O número de operações de gravação por segundo.
	`writeKb`	Total de gravação	O número total de kilobytes gravados.
	`writeKbPS`	Kb/s de gravação	O número de kilobytes gravados por segundo.

Grupo	Métrica	Nome do console	Descrição
	writeLatency	Latência de gravação	O tempo médio decorrido entre o envio de uma solicitação de E/S de gravação e sua conclusão, em milissegundos. Essa métrica só está disponível para o Amazon Aurora.
	writeThroughput	Taxa de transferência de gravação	A quantidade de taxa de transferência da rede usada por respostas do cluster de banco de dados, em bytes por segundo. Essa métrica só está disponível para o Amazon Aurora.
	wrqmPS	Wrqms	O número de solicitações de gravação mescladas enfileiradas por segundo.
mountPoint	Não aplicável	O caminho para o sistema de arquivos.	
fileSys	maxFiles	Máximo de Inodes	O número máximo de arquivos que podem ser criados para o sistema de arquivos.
	mountPoint	Não aplicável	O caminho para o sistema de arquivos. Quando isso é `/rdsdbdata*` , representa o agregado de todos os volumes de armazenamento.
	name	Não aplicável	O nome do sistema de arquivos.
	total	Total do sistema de arquivos	O número total de espaço disponível em disco para o sistema de arquivos, em kilobytes.

Grupo	Métrica	Nome do console	Descrição
	`used`	Sistema de arquivos usado	A quantidade de espaço em disco usada pelos arquivos no sistema de arquivos, em kilobytes.
	`usedFilePercent`	Inodes usados	A porcentagem de arquivos disponíveis em uso.
	`usedFiles`	% de utilização	O número de arquivos no sistema de arquivos.
	`usedPercent`	Sistema de arquivos usado	A porcentagem do espaço em disco do sistema de arquivos em uso.
`loadAverageMinute`	`fifteen`	Carga média de 15 min	O número de processos que estão solicitando tempo de CPU nos últimos 15 minutos.
	`five`	Carga média de 5 min	O número de processos que estão solicitando tempo de CPU nos últimos 5 minutos.
	`one`	Carga média de 1 min	O número de processos que estão solicitando tempo de CPU no último minuto.
`memory`	`active`	Memória ativa	A quantidade de memória atribuída, em kilobytes.
	`buffers`	Memória armazenada em buffer	A quantidade de memória usada para o buffer de solicitações de E/S antes de gravar no dispositivo de armazenamento, em kilobytes.
	`cached`	Memória em cache	A quantidade de memória utilizada para o armazenamento em cache da E/S baseada em sistema de arquivos.

Grupo	Métrica	Nome do console	Descrição
	`dirty`	Memória suja	A quantidade de páginas de memória na RAM que foram modificadas, mas não gravadas nos blocos de dados relacionados no armazenamento, em kilobytes.
	`free`	Memória livre	A quantidade de memória não atribuída, em kilobytes.
	`hugePagesFree`	Páginas enormes livres	O número de páginas enormes livres. Páginas enormes são um recurso do kernel do Linux.
	`hugePagesRsvd`	Páginas enormes reservadas	O número de páginas enormes confirmadas.
	`hugePagesSize`	Tamanho de páginas enormes	O tamanho de cada unidade de páginas enormes, em kilobytes.
	`hugePagesSurp`	Páginas enormes surp	O número de páginas enormes excedentes disponíveis em comparação com o total.
	`hugePagesTotal`	Total de páginas enormes	O número total de páginas enormes.
	`inactive`	Memória inativa	A quantidade de páginas de memória usadas com menos frequência, em kilobytes.
	`mapped`	Memória mapeada	A quantidade total de conteúdo do sistema de arquivos que é mapeada na memória dentro de um espaço de endereçamento de processos, em kilobytes.
	`pageTables`	Tabelas de página	A quantidade de memória usada por tabelas de página, em kilobytes.

Grupo	Métrica	Nome do console	Descrição
	`slab`	Memória dividida	A quantidade de estruturas de dados reutilizáveis do kernel, em kilobytes.
	`total`	Memória total	A quantidade total de memória, em kilobytes.
	`writeback`	Memória de gravação	A quantidade de páginas sujas na RAM que ainda estão sendo gravadas no armazenamento de suporte, em kilobytes.
`network`	`interface`	Não aplicável	O identificador da interface de rede que está sendo usada para a instância de banco de dados.
	`rx`	RX	O número de bytes recebidos por segundo.
	`tx`	TX	O número de bytes carregados por segundo.
`processList`	`cpuUsedPc`	da CPU %	A porcentagem de CPU usada pelo processo.
	`id`	Não aplicável	O identificador do processo.
	`memoryUsedPc`	MEM%	A porcentagem da memória usada pelo processo.
	`name`	Não aplicável	O nome do processo.
	`parentID`	Não aplicável	O identificador do processo para o processo pai do processo.
	`rss`	RES	A quantidade de RAM alocada ao processo, em kilobytes.

Grupo	Métrica	Nome do console	Descrição
	`tgid`	Não aplicável	O identificador do grupo de threads, que é um número que representa o ID do processo ao qual um thread pertence. Esse identificador é usado para agrupar threads do mesmo processo.
	`vss`	VIRT	A quantidade de memória virtual alocada ao processo, em kilobytes.
`swap`	`total`	Troca	A quantidade de memória de permuta disponível, em kilobytes.
	`in`	Trocas feitas	A quantidade de memória, em kilobytes, transferida temporariamente do disco.
	`out`	Trocas recebidas	A quantidade de memória, em kilobytes, transferida temporariamente para o disco.
	`free`	Troca livre	A quantidade de memória de troca livre, em kilobytes.
	`cached`	Troca confirmada	A quantidade de memória de permuta, em kilobytes, usada como a memória cache.
`tasks`	`blocked`	Tarefas bloqueadas	O número de tarefas que estão bloqueadas.
	`running`	Tarefas em execução	O número de tarefas que estão sendo executadas.
	`sleeping`	Tarefas em espera	O número de tarefas que estão em suspensão.
	`stopped`	Tarefas paradas	O número de tarefas que estão interrompidas.

Grupo	Métrica	Nome do console	Descrição
	`total`	Total de tarefas	O número total de tarefas.
	`zombie`	Tarefas Zombie	O número de tarefas filho que estão inativas com uma tarefa pai ativa.

Monitorar eventos, logs e fluxos em um cluster de banco de dados do Amazon Aurora

Quando você monitora seus bancos de dados Amazon Aurorae outras soluções da AWS, seu objetivo é manter o seguinte:

- Confiabilidade
- Disponibilidade
- desempenho
- Segurança

Monitorar métricas em um cluster do Amazon Aurora explica como monitorar seu cluster usando métricas. Uma solução completa também deve monitorar eventos de banco de dados, arquivos de log e transmissões de atividades. A AWS fornece as seguintes ferramentas de monitoramento:

- O Amazon EventBridge é um serviço de barramento de eventos sem servidor que facilita a conexão de aplicações a dados de diversas origens. O EventBridge fornece uma fluxo de dados em tempo real de suas próprias aplicações, de aplicações de software como serviço (SaaS) e de serviços da AWS. O EventBridge encaminha esses dados para destinos como AWS Lambda. Dessa forma, você pode monitorar eventos que ocorrem em serviços e criar arquiteturas orientadas a eventos. Para saber mais, consulte o Guia do Usuário do Amazon EventBridge.
- O Amazon CloudWatch Logs oferece uma forma de monitorar, armazenar e acessar os arquivos de log de instâncias do Amazon Aurora, do AWS CloudTrail e de outras fontes. O Amazon CloudWatch Logs pode monitorar informações nos arquivos de log e notificar você quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para saber mais, consulte o Guia do usuário do Amazon CloudWatch Logs.
- AWS CloudTrail O captura chamadas de API e eventos relacionados feitos por sua conta da ou em nome dela Conta da AWS. O CloudTrail fornece os arquivos de log para um bucket do Amazon S3 especificado por você. Você pode identificar quais usuários e contas chamaram a AWS, o endereço IP de origem no qual as chamadas foram feitas e quando elas ocorreram. Para saber mais, consulte o Guia do usuário do AWS CloudTrail.
- O Database Activity Streams é um recurso do Amazon Aurora que fornece um fluxo quase em tempo real da atividade em seu cluster de banco de dados do Oracle. O Amazon Aurora

envia atividades para um fluxo de dados do Amazon Kinesis. O fluxo do Kinesis é criado automaticamente. No Kinesis, é possível configurar serviços da AWS, como o Amazon Data Firehose e o AWS Lambda, para consumir o fluxo e armazenar os dados.

Tópicos

- Visualizar logs, eventos e transmissões no console do Amazon RDS
- Monitorar eventos do Amazon Aurora
- Monitorar arquivos de log do Amazon Aurora
- Monitorar chamadas de API do Amazon Aurorano AWS CloudTrail
- Monitorar o Amazon Aurora com o recurso Database Activity Streams
- Monitorar ameaças com o Amazon GuardDuty RDS Protection para Amazon Aurora

Visualizar logs, eventos e transmissões no console do Amazon RDS

O Amazon RDS se integra aos Serviços da AWS para mostrar informações sobre logs, eventos e transmissões de atividades de banco de dados no console do RDS.

A guia Logs e eventos do cluster de banco de dados do Aurora mostra as seguintes informações:

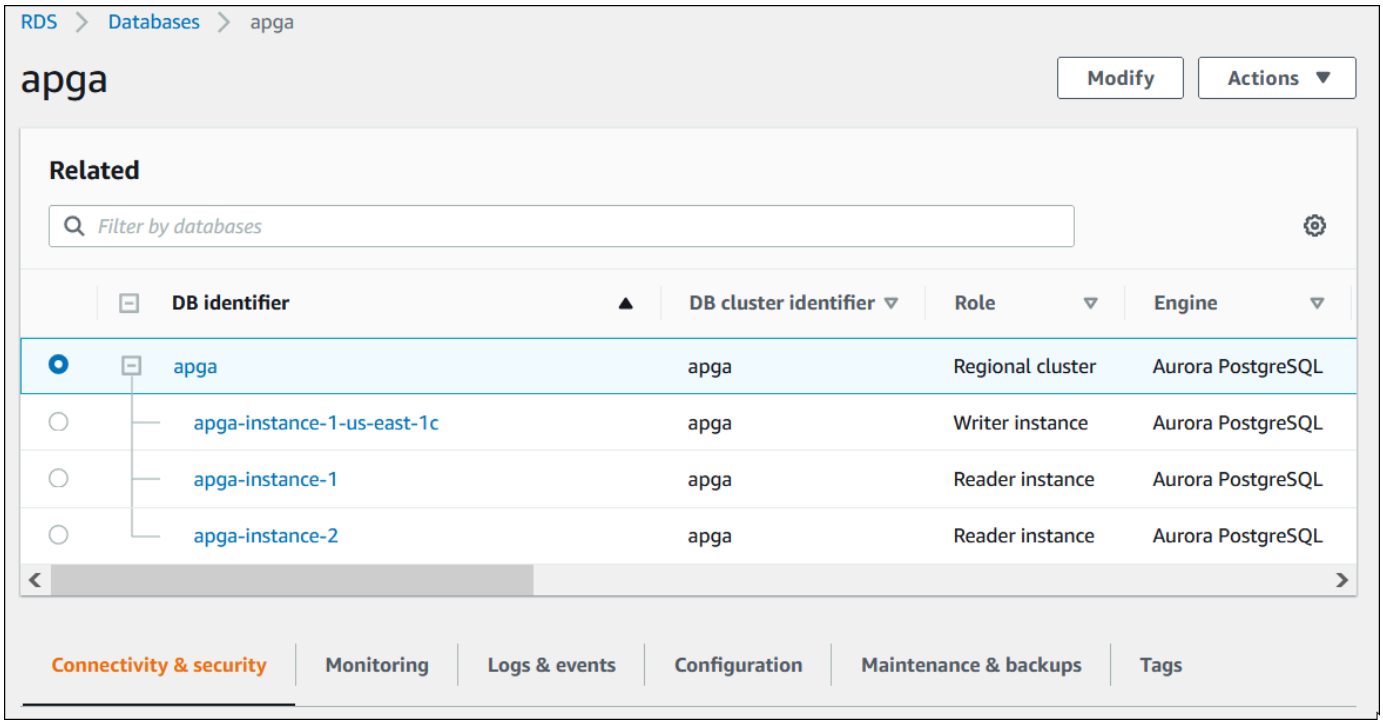
- Políticas e atividades de autoescalabilidade: mostra políticas e atividades relacionadas ao recurso de autoescalabilidade do Aurora. Essas informações só aparecem na guia Logs & events (Logs e eventos) no nível do cluster.
- Alertas do Amazon CloudWatch: mostra todos os alarmes de métricas configurados para a instância de banco de dados em seu cluster do Aurora. Se você ainda não configurou alarmes, poderá criá-los no console do RDS.
- Eventos recentes: mostra um resumo dos eventos (alterações no ambiente) relacionados à sua instância ou cluster de banco de dados Aurora. Para obter mais informações, consulte Visualizar eventos do Amazon RDS.
- Logs: mostra os arquivos de log do banco de dados gerados por uma instância de banco de dados em seu cluster do Aurora. Para obter mais informações, consulte Monitorar arquivos de log do Amazon Aurora.

A guia Configuration (Configuração) exibe informações sobre transmissões de atividades de banco de dados.

Como visualizar logs, eventos e fluxos do cluster de banco de dados do Aurora no console do RDS

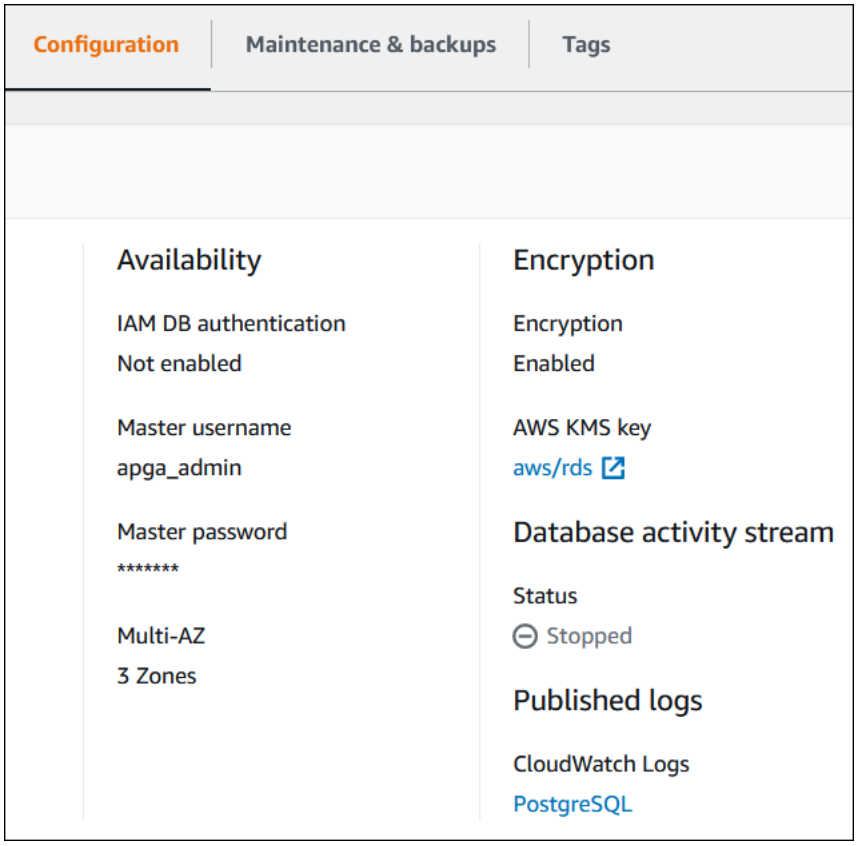
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o nome do cluster de banco de dados Aurora que deseja monitorar.

   A página “Databases” (Bancos de dados) é exibida. O exemplo a seguir mostra um cluster de banco de dados Amazon Aurora PostgreSQL chamado `apga`.



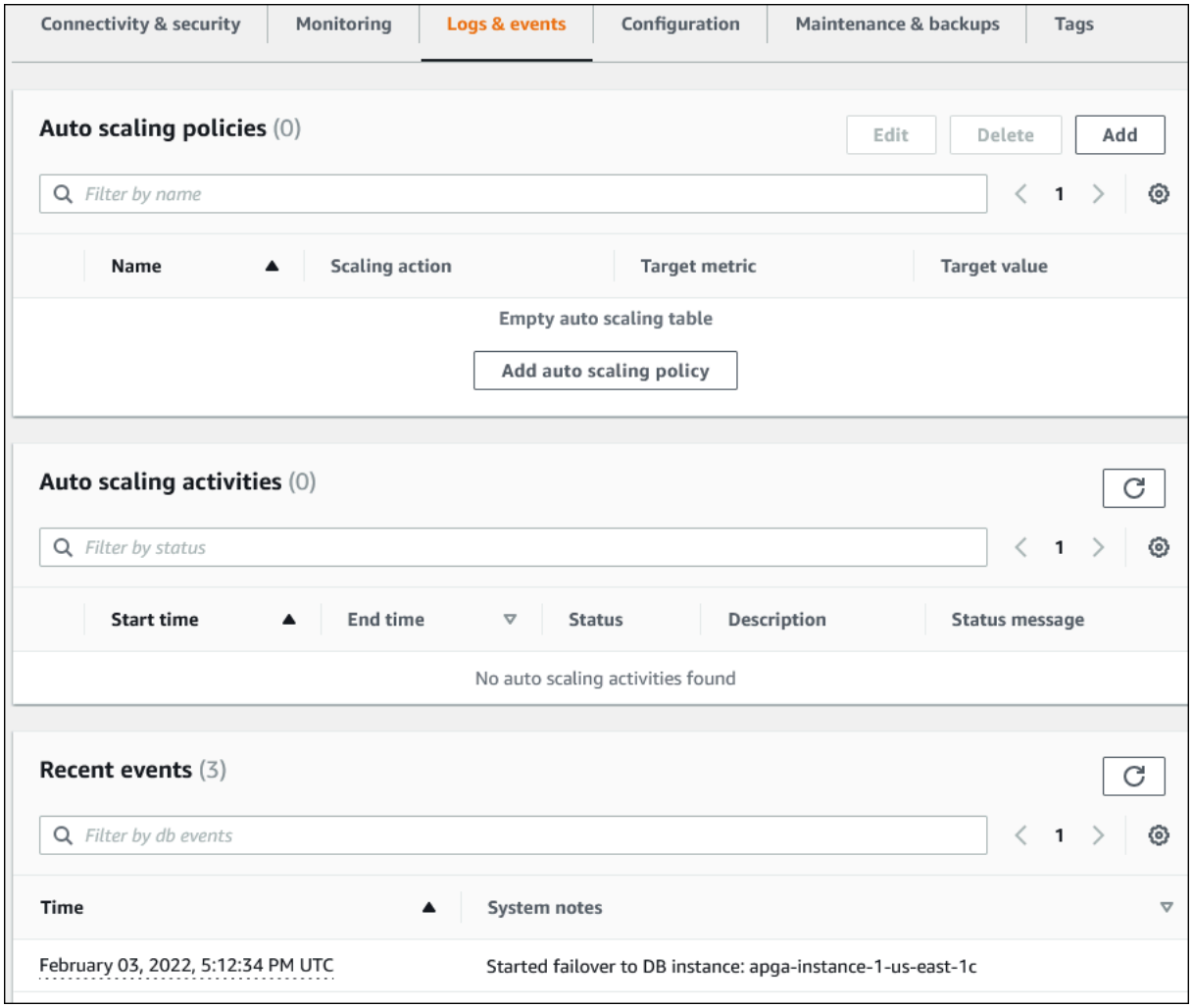
4. Role para baixo e escolha Configuration (Configuração).

   O exemplo a seguir mostra o status das transmissões de atividades de banco de dados do cluster.



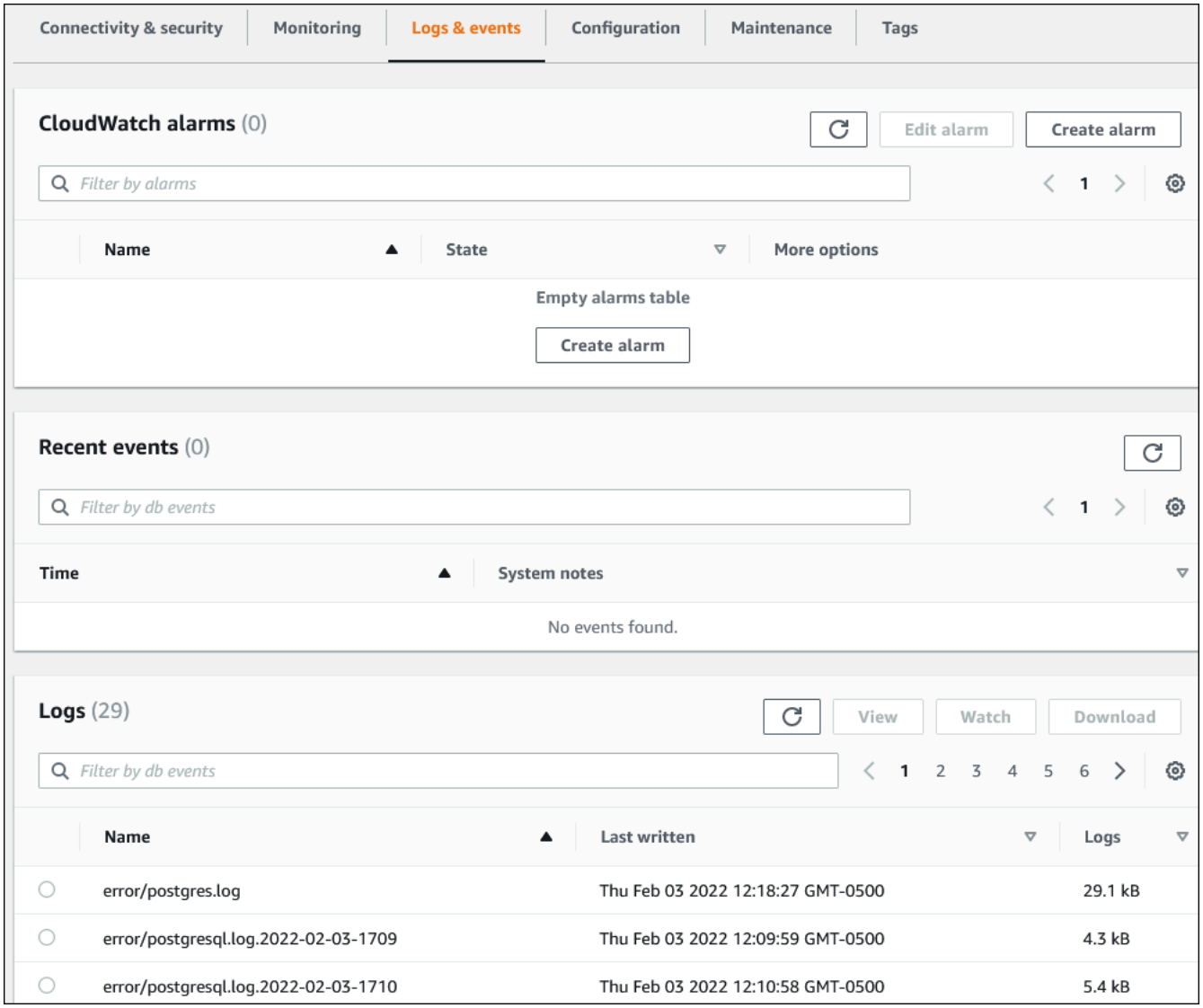
5. Escolha Logs & events (Logs e eventos).

   A seção “Logs & events” (Logs e eventos) é exibida.



6. Escolha uma instância de banco de dados em seu cluster do Aurora e, em seguida, escolha a guia Logs & eventos (Logs e eventos) da instância.

   O exemplo a seguir mostra que o conteúdo é diferente entre a página da instância de banco de dados e a página do cluster de banco de dados. A página da instância de banco de dados mostra logs e alarmes.


Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance
Tags
CloudWatch alarms (0)
Edit alarm
Create alarm
Filter by alarms
1
Name
State
More options
Empty alarms table
Create alarm
Recent events (0)
Filter by db events
1
Time
System notes
No events found.
Logs (29)
View
Watch
Download
Filter by db events
1 2 3 4 5 6
Name
Last written
Logs
error/postgres.log
Thu Feb 03 2022 12:18:27 GMT-0500
29.1 kB
error/postgresql.log.2022-02-03-1709
Thu Feb 03 2022 12:09:59 GMT-0500
4.3 kB
error/postgresql.log.2022-02-03-1710
Thu Feb 03 2022 12:10:58 GMT-0500
5.4 kB

Monitorar eventos do Amazon Aurora

Um evento indica uma alteração em um ambiente. Isso pode ser um ambiente da AWS, um serviço ou uma aplicação de parceiro de SaaS ou uma das suas próprios aplicações ou serviços personalizados. Para obter descrições de eventos do Aurora, consulte Categorias de eventos e mensagens de eventos para o Amazon RDS.

Tópicos

- Visão geral dos eventos para Aurora
- Visualizar eventos do Amazon RDS
- Trabalhar com a notificação de eventos do Amazon RDS
- Criar uma regra que é acionada em um evento do Amazon Aurora
- Categorias de eventos e mensagens de eventos para o Amazon RDS

Visão geral dos eventos para Aurora

Um evento do RDS indica uma alteração no ambiente do Aurora. Por exemplo, o Amazon Aurora gera um evento quando um cluster de banco de dados recebe patch. O Amazon Aurora entrega eventos ao EventBridge quase em tempo real.

> **Note**
>
> O Amazon RDS emite eventos em uma base de melhor esforço. Recomendamos que você evite gravar programas que dependam da ordem ou da existência de eventos de notificação, pois eles podem estar fora de sequência ou ausentes.

O Amazon RDS registra eventos relacionados aos seguintes recursos:

- Clusters do banco de dados

  Para obter uma lista de eventos de cluster, consulte Eventos de cluster de banco de dados.
- Instâncias de banco de dados

  Para obter uma lista de eventos de instância de banco de dados, consulte Eventos de instância de banco de dados.
- Grupos de parâmetros do banco de dados

Para obter uma lista de eventos de grupo de parâmetros de banco de dados, consulte Eventos de grupo de parâmetros de banco de dados.

- Grupos de segurança de banco de dados

  Para obter uma lista de eventos de grupo de segurança do banco de dados, consulte Eventos de grupos de segurança de banco de dados.

- Snapshots de cluster de banco de dados

  Para obter uma lista de eventos de snapshot de cluster de banco de dados, consulte Eventos de snapshot de cluster de banco de dados.

- Eventos do RDS Proxy

  Para obter uma lista de eventos do proxy do RDS, consulte Eventos do RDS Proxy.

- Eventos de implantação azul/verde

  Para obter uma lista de eventos de implantação azul/verde, consulte Eventos de implantação azul/verde.

Essas informações incluem:

- A data e a hora do evento.
- O nome da origem e o tipo de origem do evento
- A mensagem associada ao evento
- As notificações de eventos incluem tags de quando a mensagem foi enviada e podem não refletir as tags no momento em que o evento ocorreu

Visualizar eventos do Amazon RDS

É possível recuperar as seguintes informações do evento para seus recursos do Amazon Aurora:

- Nome do recurso
- Tipo de recurso
- Horário do evento
- Resumo de mensagens do evento

É possível acessar eventos nas seguintes partes do Console de gerenciamento da AWS:

- A guia Eventos, que mostra eventos das últimas 24 horas.
- A tabela Eventos recentes na seção Logs e eventos na guia Bancos de dados, que pode mostrar eventos das últimas duas semanas.

Também é possível recuperar eventos usando o comando describe-events da AWS CLI ou a operação DescribeEvents da API do RDS. Se você usar a AWS CLI ou a API do RDS para visualizar eventos, poderá recuperar eventos até os últimos 14 dias.

> **Note**
>
> Se precisar armazenar eventos por períodos mais longos, você poderá enviar eventos do Amazon RDS ao EventBridge. Para ter mais informações, consulte Criar uma regra que é acionada em um evento do Amazon Aurora.

Para obter descrições de eventos do Amazon Aurora, consulte Categorias de eventos e mensagens de eventos para o Amazon RDS.

Para acessar informações detalhadas sobre eventos usando o AWS CloudTrail, incluindo parâmetros de solicitação, consulte Eventos do CloudTrail.

Console

Como visualizar todos os eventos do Amazon RDS nas últimas 24 horas

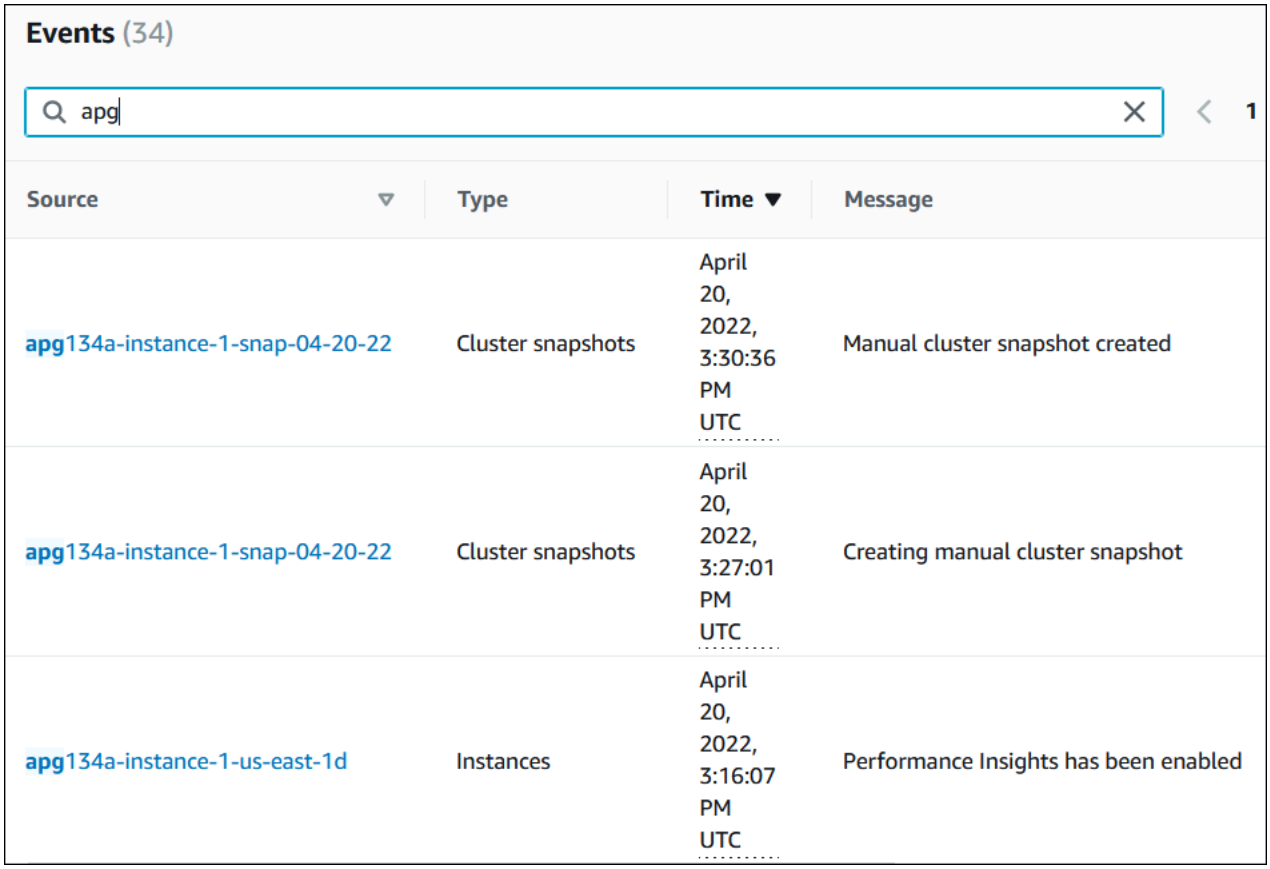
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

2. No painel de navegação, escolha Events.

   Os eventos disponíveis aparecem em uma lista.

3. (Opcional) Insira um termo de pesquisa para filtrar seus resultados.

   O exemplo a seguir mostra uma lista de eventos filtrados pelos caracteres **apg**.



AWS CLI

Para visualizar todos os eventos gerados na última hora, chame describe-events sem parâmetros.

```
aws rds describe-events
```

O exemplo de saída a seguir mostra que uma instância de cluster de banco de dados iniciou a recuperação.

```
{
    "Events": [
        {
            "EventCategories": [
```

```
                "recovery"
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mycluster-instance-1",
            "Date": "2022-04-20T15:02:38.416Z",
            "Message": "Recovery of the DB instance has started. Recovery time will
 vary with the amount of data to be recovered.",
            "SourceIdentifier": "mycluster-instance-1"
        }, ...
```

Para visualizar todos os eventos do Amazon RDS dos últimos 10.080 minutos (sete dias), chame o comando describe-events AWS CLI e defina o parâmetro `--duration` como `10080`.

```
aws rds describe-events --duration 10080
```

O exemplo a seguir mostra os eventos no intervalo de tempo especificado para a instância de banco de dados *`test-instance`*.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

O exemplo de saída a seguir mostra o status de um backup.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
```

```
                "backup"
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

É possível visualizar todos os eventos de instâncias do Amazon RDS dos últimos 14 dias, chamando a operação da API do RDS DescribeEvents e definindo o parâmetro `Duration` como `20160`.

Trabalhar com a notificação de eventos do Amazon RDS

O Amazon RDS usa o Amazon Simple Notification Service (Amazon SNS) para fornecer uma notificação quando um evento do Amazon RDS ocorre. Essas notificações podem estar em qualquer formato de notificação compatível com o Amazon SNS para uma região da AWS, como um e-mail, uma mensagem de texto ou uma chamada para um endpoint HTTP.

Tópicos

- Visão geral das notificações de eventos do Amazon RDS
- Conceder permissões para publicar notificações em um tópico do Amazon SNS
- Inscrever-se em notificações de eventos do Amazon RDS
- Tags e atributos de notificação de eventos do Amazon RDS
- Listar assinaturas de notificação de evento do Amazon RDS
- Modificar uma assinatura de notificação de evento do Amazon RDS
- Adicionar um identificador de origem a uma assinatura de notificação de evento do Amazon RDS
- Remover um identificador de origem de uma assinatura de notificação de eventos do Amazon RDS
- Listar as categorias de notificação de evento do Amazon RDS
- Excluir uma assinatura de notificação de evento do Amazon RDS

Visão geral das notificações de eventos do Amazon RDS

O Amazon RDS agrupa eventos em categorias em que você pode se inscrever para receber notificações quando um evento ocorrer na categoria.

Tópicos

- Recursos do RDS elegíveis para assinatura de eventos
- Processo básico de assinatura de notificações de eventos do Amazon RDS
- Entrega de notificações de evento do RDS
- Faturamento de notificações de eventos do Amazon RDS
- Exemplos de eventos do Aurora usando o Amazon EventBridge

Recursos do RDS elegíveis para assinatura de eventos

No Amazon Aurora, os eventos ocorrem tanto no nível do cluster de banco de dados como no da instância de banco de dados. Você pode se inscrever em uma categoria de evento para os seguintes recursos:

- DB instance (Instância de banco de dados)
- Cluster de banco de dados
- Snapshot de cluster de banco de dados
- Grupo de parâmetros de banco de dados
- DB security group (grupo de segurança de banco de dados)
- RDS Proxy
- Versões de mecanismo personalizadas

Por exemplo, se você se inscrever na categoria de backup para uma determinada instância de banco de dados, receberá notificações sempre que houver um evento relacionado ao backup que afete a instância de banco de dados. Se você se inscrever em uma categoria de alteração de configuração para uma instância de banco de dados, será notificado quando a instância de banco de dados for alterada. Você também recebe uma notificação quando uma assinatura de notificação de evento é alterada.

Você pode querer criar várias assinaturas diferentes. Por exemplo, você pode criar uma assinatura que receba todas as notificações de eventos de todas as instâncias de banco de dados e outra que inclua somente eventos críticos de um subconjunto das instâncias de bancos de dados. Para a segunda assinatura, especifique uma ou mais instâncias de banco de dados no filtro.

Processo básico de assinatura de notificações de eventos do Amazon RDS

O processo de inscrição na notificação de evento do Amazon RDS é o seguinte:

1. Você cria uma assinatura de notificação de eventos do Amazon RDS usando o console do Amazon RDS, a AWS CLI ou a API.

   O Amazon RDS usa o ARN de um tópico do Amazon SNS para identificar cada assinatura. O console do Amazon RDS cria o ARN para você quando cria a assinatura. Crie o ARN usando o console do Amazon SNS, a AWS CLI ou a API do Amazon SNS.
2. O Amazon RDS envia um e-mail ou mensagem SMS de aprovação para os endereços que você submeteu com a assinatura.

3. Você confirma a assinatura escolhendo o link na notificação que recebeu.
4. O console do Amazon RDS atualiza a seção My Event Subscriptions (Minhas assinaturas de eventos) com o status de sua assinatura.
5. O Amazon RDS começa a enviar notificações aos endereços que você forneceu ao criar a assinatura.

Para saber mais sobre o Gerenciamento de Identidade e Acesso ao usar o Amazon SNS, consulte Gerenciamento de Identidade e Acesso no Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.

É possível usar o AWS Lambda para processar notificações de eventos de uma instância de banco de dados. Para obter mais informações, consulte Uso do AWS Lambda com o Amazon RDS no Guia do desenvolvedor do AWS Lambda.

Entrega de notificações de evento do RDS

O Amazon RDS envia notificações aos endereços que você fornece ao criar a assinatura. A notificação pode incluir atributos de mensagem que fornecem metadados estruturados sobre a mensagem. Para obter mais informações sobre os atributos de mensagem, consulte Categorias de eventos e mensagens de eventos para o Amazon RDS.

As notificações de eventos podem levar até cinco minutos para serem entregues.

> **Important**
>
> O Amazon RDS não garante a ordem dos eventos enviados em um fluxo de eventos. A ordem do evento está sujeita a alterações.

Quando o Amazon SNS envia uma notificação para um endpoint HTTP ou HTTPS inscrito, a mensagem POST enviada ao endpoint tem um corpo de mensagem que contém um documento JSON. Para obter mais informações, consulte Mensagens do Amazon SNS e formatos JSON no Guia do desenvolvedor do Amazon Simple Notification Service.

É possível configurar o SNS para notificar você com mensagens de texto. Para obter mais informações, consulte Mensagens de texto móveis (SMS) no Guia do desenvolvedor do Amazon Simple Notification Service.

Para desativar as notificações sem excluir uma assinatura, escolha No (Não) para Enabled (Habilitado) no console do Amazon RDS. Ou você pode definir o parâmetro `Enabled` para `false` usando a AWS CLI ou a API do Amazon RDS.

Faturamento de notificações de eventos do Amazon RDS

O faturamento da notificação de eventos do Amazon RDS é feito por meio do Amazon SNS. As taxas do Amazon SNS se aplicam durante o uso da notificação de eventos. Para obter mais informações sobre o faturamento do Amazon SNS, consulte Preço do Amazon Simple Notification Service.

Exemplos de eventos do Aurora usando o Amazon EventBridge

Os exemplos a seguir mostram diferentes tipos de eventos do Aurora no formato JSON. Para acessar um tutorial que mostre como capturar e visualizar eventos no formato JSON, consulte Tutorial: Registro de alterações de estado de uma instância de banco de dados usando o Amazon EventBridge.

Tópicos

- Exemplo de um evento de cluster de banco de dados
- Exemplo de um evento de grupo de parâmetros de banco de dados
- Exemplo de um evento de snapshot de cluster de banco de dados

Exemplo de um evento de cluster de banco de dados

Veja a seguir um exemplo de um evento de cluster de banco de dados no formato JSON. O evento mostra que o cluster chamado `my-db-cluster` foi corrigido. O ID do evento é `RDS-EVENT-0173`.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster"
  ],
  "detail": {
```

```
    "EventCategories": [
      "notification"
    ],
    "SourceType": "CLUSTER",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Database cluster has been patched",
    "SourceIdentifier": "my-db-cluster",
    "EventID": "RDS-EVENT-0173"
  }
}
```

Exemplo de um evento de grupo de parâmetros de banco de dados

Veja a seguir um exemplo de um evento de grupo de parâmetros de banco de dados no formato JSON. O evento mostra que o parâmetro `time_zone` foi atualizado no grupo de parâmetros `my-db-param-group`. O ID do evento é RDS-EVENT-0037.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

Exemplo de um evento de snapshot de cluster de banco de dados

Veja a seguir um exemplo de um evento de snapshot de cluster de banco de dados no formato JSON. O evento mostra a criação do snapshot chamado `my-db-cluster-snapshot`. O ID do evento é RDS-EVENT-0074.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-cluster-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "backup"
    ],
    "SourceType": "CLUSTER_SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-
cluster-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "SourceIdentifier": "my-db-cluster-snapshot",
    "Message": "Creating manual cluster snapshot",
    "EventID": "RDS-EVENT-0074"
  }
}
```

Conceder permissões para publicar notificações em um tópico do Amazon SNS

Para conceder permissões do Amazon RDS a fim de publicar notificações em um tópico do Amazon Simple Notification Service (Amazon SNS), anexe uma política do AWS Identity and Access Management (IAM) ao tópico de destino. Para obter mais informações sobre permissões, consulte Exemplos de casos de controle de acesso do Amazon Simple Notification Service no Guia do desenvolvedor do Amazon Simple Notification Service.

Por padrão, um tópico do Amazon SNS tem uma política que permite que todos os recursos do Amazon RDS na mesma conta publiquem notificações nele. Você pode anexar uma política personalizada para possibilitar notificações entre contas ou restringir o acesso a determinados recursos.

Veja a seguir um exemplo de uma política do IAM anexada ao tópico de destino do Amazon SNS. Ela restringe o tópico a instâncias de banco de dados com nomes correspondentes ao prefixo especificado. Para usar essa política, especifique os seguintes valores:

- `Resource`: o nome do recurso da Amazon (ARN) do tópico do Amazon SNS
- `SourceARN`: o ARN de recursos do RDS
- `SourceAccount`: o ID de sua Conta da AWS

Para ver uma lista dos tipos de recursos e seus ARNs, consulte Recursos definidos pelo Amazon RDS na Referência de autorização do serviço.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Inscrever-se em notificações de eventos do Amazon RDS

A forma mais fácil de criar uma assinatura é com o console do RDS. Se você preferir criar assinaturas de notificações de eventos usando a CLI ou API, é necessário criar um tópico do Amazon Simple Notification Service e fazer a assinatura desse tópico com o console do Amazon SNS ou a API do Amazon SNS. Também será necessário reter o nome de recurso da Amazon (ARN) do tópico, pois ele é usado ao enviar comandos da CLI ou operações da API. Para obter informações sobre como criar um tópico do SNS e assiná-lo, consulte Conceitos básicos do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.

Você pode especificar o tipo de origem sobre o qual deseja ser notificado e a origem do Amazon RDS que aciona o evento:

Source type (Tipo de origem)

O tipo de fonte. Por exemplo, Source type (Tipo de origem) pode ser Instances (Instâncias). Você deve escolher um tipo de origem.

Resources to include (Recursos para incluir)

Os recursos do Amazon RDS que estão gerando os eventos. Por exemplo, você pode escolher Select specific instances (Selecionar instâncias específicas), depois myDBInstance1.

A tabela a seguir explica o resultado quando você especifica ou não ***Resources*** to include (Recursos para incluir).

Recursos para incluir	Descrição	Exemplo
Especificado	O RDS notifica você sobre todos os eventos somente do recurso especificado.	Se o seu Source type (Tipo de origem) for Instances (Instâncias) e seu recurso for myDBInstance1, o RDS notificará você sobre todos os eventos somente para myDBInstance1.
Não especificado	O RDS notifica você sobre os eventos do tipo de fonte especificado para todos os recursos do Amazon RDS.	Se o Source type (Tipo de origem) for Instances (Instâncias), o RDS notificará você sobre

Recursos para incluir	Descrição	Exemplo
		todos os eventos relacionados à instância em sua conta.

Por padrão, um assinante de tópico do Amazon SNS recebe todas as mensagens publicadas no tópico. Para receber apenas um subconjunto das mensagens, o assinante deve atribuir uma política de filtro à assinatura do tópico. Para obter mais informações sobre filtragem de mensagens do SNS, consulte Filtragem de mensagens do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.

Console

Para assinar a notificação de eventos do RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Event subscriptions (Assinaturas de eventos).
3. No painel Event subscriptions (Assinaturas de eventos), escolha Create event subscription (Criar assinatura de evento).
4. Insira os detalhes da sua assinatura da seguinte forma:

   a. Em Name (Nome), insira um nome para a assinatura de notificação de eventos.

   b. Para Send notifications to (Enviar notificação para), utilize um dos seguintes procedimentos:

      - Escolha New email topic (Novo tópico de e-mail). Insira um nome para o tópico do seu e-mail e uma lista de destinatários. Recomendamos que você configure as assinaturas de eventos com o mesmo endereço de e-mail do contato primário da sua conta. As mensagens de recomendações, eventos de serviço e integridade pessoal são enviadas por meio de canais diferentes. As assinaturas com o mesmo endereço de e-mail garantem que todas as mensagens sejam consolidadas em um único local.
      - Escolha Amazon Resource Name (ARN) [Nome do recurso da Amazon (ARN)]. Depois, escolha o ARN do Amazon SNS existente para um tópico do Amazon SNS.

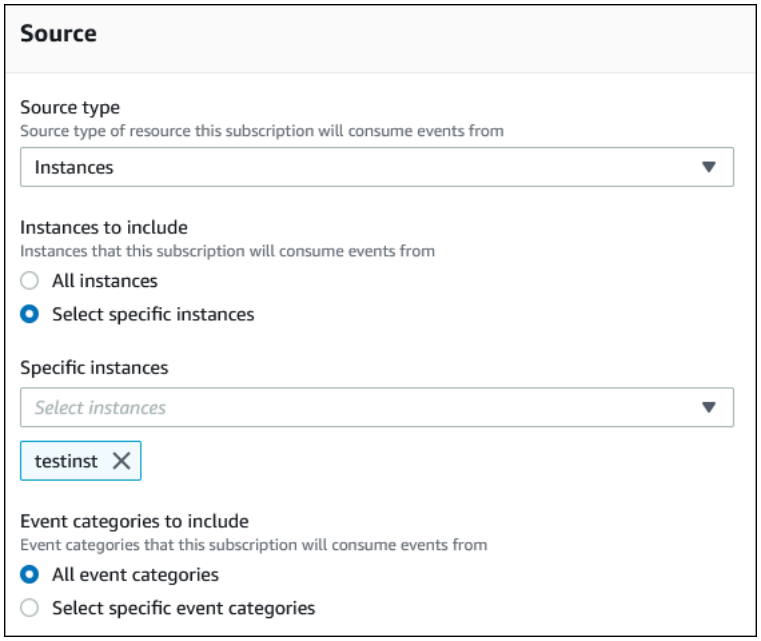
        Se você quiser usar um tópico que tenha sido ativado para criptografia do lado do servidor (SSE), conceda ao Amazon RDS as permissões necessárias para acessar a

AWS KMS key. Para obter mais informações, consulte Ativar a compatibilidade entre as fontes de eventos de tópicos criptografados e serviços da AWS no Guia do desenvolvedor do Amazon Simple Notification Service.

c. Em Source type (Tipo de origem), escolha um tipo de origem. Por exemplo, escolha Clusters ou Cluster snapshots (Snapshots de cluster).

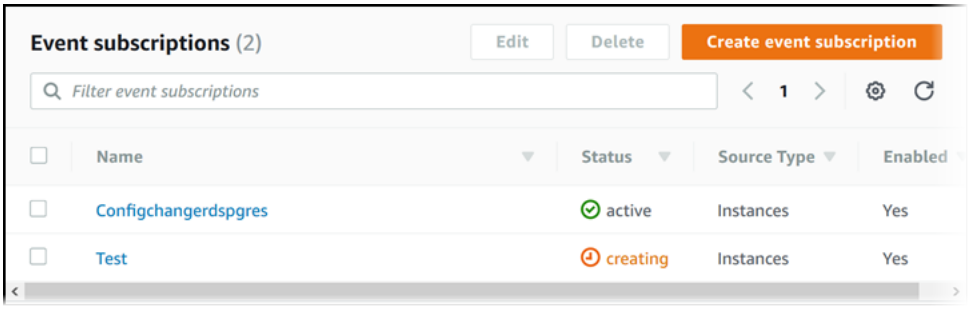
d. Escolha as categorias e recursos de eventos para os quais quer receber notificações de eventos.

   O exemplo a seguir configura notificações de eventos para a instância de banco de dados denominada `testinst`.



e. Escolha Criar.

O console do Amazon RDS indica que a assinatura está sendo criada.



AWS CLI

Para assinar a Notificação de eventos do RDS, use o comando AWS CLI da `create-event-subscription`. Inclua os seguintes parâmetros necessários:

- `--subscription-name`
- `--sns-topic-arn`

Example

Para Linux, macOS ou Unix:

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Para Windows:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Para assinar a notificação de eventos do Amazon RDS, chame a função da API `CreateEventSubscription` do Amazon RDS. Inclua os seguintes parâmetros necessários:

- `SubscriptionName`
- `SnsTopicArn`

Tags e atributos de notificação de eventos do Amazon RDS

Quando o Amazon RDS envia uma notificação de evento ao Amazon Simple Notification Service (SNS) ou ao Amazon EventBridge, a notificação contém atributos de mensagem e tags de evento. O RDS envia os atributos da mensagem separadamente com a mensagem, enquanto as tags de evento estão no corpo da mensagem. Use os atributos de mensagem e as tags do Amazon RDS para adicionar metadados aos seus recursos. Você pode modificar essas tags com suas próprias notações sobre as instâncias de banco de dados, clusters do Aurora, etc. Para ter mais informações sobre recursos de marcação do Amazon RDS, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.

Por padrão, o Amazon SNS e o Amazon EventBridge recebem todas as mensagens enviadas a eles. O SNS e o EventBridge podem filtrar a mensagem e enviar as notificações ao modo de comunicação preferencial, como um e-mail, uma mensagem de texto ou uma chamada para um endpoint HTTP.

> **Note**
>
> A notificação enviada em um e-mail ou uma mensagem de texto não terá tags de evento.

A tabela a seguir mostra os atributos da mensagem de eventos do RDS enviada ao assinante do tópico.

Atributo de evento do Amazon RDS	Descrição
EventID	Identificador da mensagem de eventos do RDS, por exemplo, RDS-EVENT-0006.
Recurso	O identificador do ARN do recurso que emite o evento, por exemplo, `arn:aws:rds:ap-southeast-2:123456789012:db:database-1` .

As tags do RDS fornecem dados sobre o recurso que foi afetado pelo evento do serviço. O RDS adiciona o estado atual das tags ao corpo da mensagem quando a notificação é enviada ao SNS ou ao EventBridge.

Para ter mais informações sobre filtragem de atributos de mensagens do SNS, consulte Filtragem de mensagens do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.

Para obter mais informações sobre a filtragem de tags de evento para o EventBridge, consulte Operadores de comparação para uso em padrões de eventos do Amazon EventBridge no Guia do usuário do Amazon EventBridge.

Para obter mais informações sobre a filtragem de tags baseadas em carga útil para o SNS, consulte Introdução à filtragem de mensagens baseadas em carga útil para o Amazon SNS

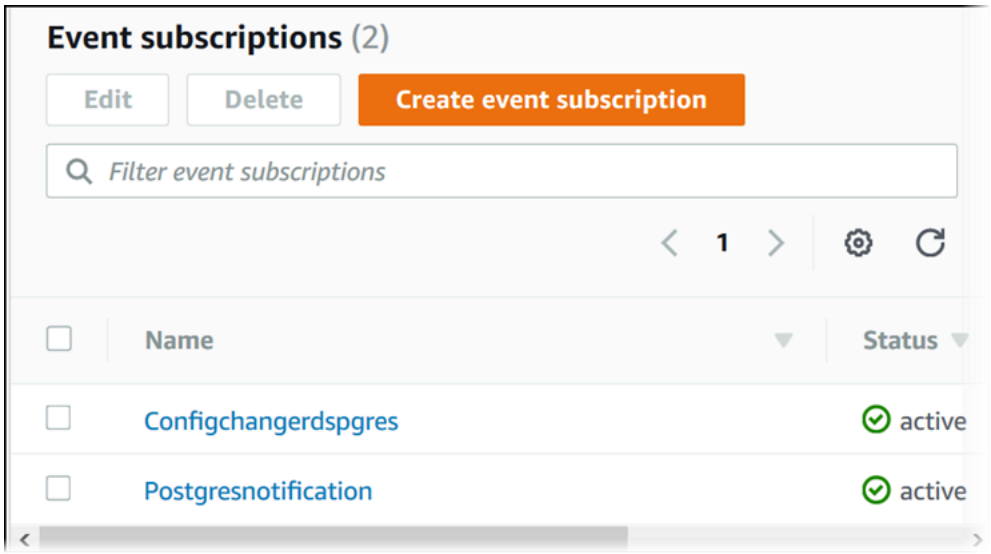
Listar assinaturas de notificação de evento do Amazon RDS

Você pode listar suas atuais assinaturas de notificações de eventos do Amazon RDS.

Console

Para listar suas atuais assinaturas de notificações de eventos do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Event subscriptions (Assinaturas de eventos). O painel Event subscriptions (Assinaturas de eventos) exibirá todas as suas assinaturas de notificação de eventos.



AWS CLI

Para listar suas atuais assinaturas de notificações de eventos do Amazon RDS, use o comando da AWS CLI `describe-event-subscriptions`l.

Example

O exemplo a seguir descreve todas as assinaturas de eventos.

```
aws rds describe-event-subscriptions
```

O exemplo a seguir descreve a pilha `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

Para listar suas atuais assinaturas de notificações de eventos do Amazon RDS, chame a ação da API do Amazon RDS `DescribeEventSubscriptions`.

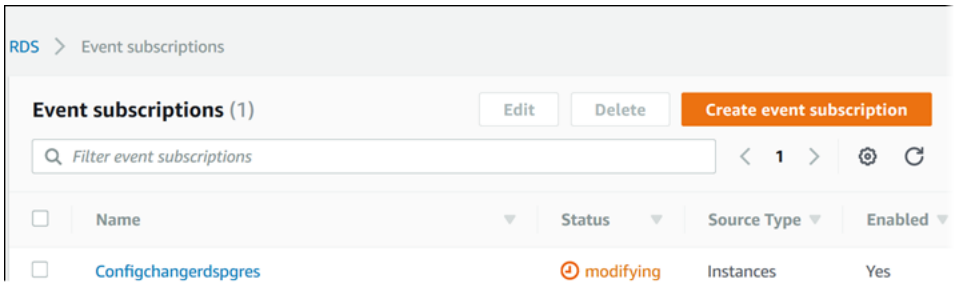
Modificar uma assinatura de notificação de evento do Amazon RDS

Depois que você criar uma assinatura, é possível alterar o nome, identificador de origem, categorias ou ARN do tópico da assinatura.

Console

Para modificar uma assinatura de notificação de evento do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Event subscriptions (Assinaturas de eventos).
3. No painel Event subscriptions (Assinaturas de eventos), escolha a assinatura que deseja modificar e escolha Edit (Editar).
4. Faça as alterações na assinatura usando as seções Target (Alvo) ou Source (Origem).
5. Selecione a opção Editar. O console do Amazon RDS indica que a assinatura está sendo modificada.



AWS CLI

Para modificar uma assinatura de notificação de evento do Amazon RDS, use o comando da AWS CLI `modify-event-subscription`. Inclua o seguinte parâmetro necessário:

- `--subscription-name`

Example

O código a seguir habilita `myeventsubscription`.

Para Linux, macOS ou Unix:

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

Para Windows:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Para modificar um evento do Amazon RDS, chame a operação da API do Amazon RDS `ModifyEventSubscription`. Inclua o seguinte parâmetro necessário:

- `SubscriptionName`

Adicionar um identificador de origem a uma assinatura de notificação de evento do Amazon RDS

Você pode adicionar um identificador de origem (a origem do Amazon RDS que gera o evento) à uma assinatura existente.

Console

Você pode facilmente adicionar ou remover identificadores de origem usando o console do Amazon RDS selecionando-os ou desmarcando-os ao modificar uma assinatura. Para obter mais informações, consulte Modificar uma assinatura de notificação de evento do Amazon RDS.

AWS CLI

Para adicionar um identificador de origem a uma assinatura de notificação de eventos do Amazon RDS, use o comando da AWS CLI `add-source-identifier-to-subscription`. Inclua os seguintes parâmetros necessários:

- `--subscription-name`
- `--source-identifier`

Example

O exemplo a seguir adiciona o identificador de origem `mysqldb` à assinatura `myrdseventsubscription`

Para Linux, macOS ou Unix:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Para Windows:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

solicitações de

Para adicionar um identificador de origem a uma assinatura de notificação de evento do Amazon RDS, chame a API do Amazon RDS `AddSourceIdentifierToSubscription`. Inclua os seguintes parâmetros necessários:

- `SubscriptionName`
- `SourceIdentifier`

Remover um identificador de origem de uma assinatura de notificação de eventos do Amazon RDS

Você pode remover um identificador de origem (a origem do Amazon RDS que gera o evento) de uma assinatura se você não quiser mais ser notificado sobre eventos para aquela origem.

Console

Você pode facilmente adicionar ou remover identificadores de origem usando o console do Amazon RDS selecionando-os ou desmarcando-os ao modificar uma assinatura. Para obter mais informações, consulte Modificar uma assinatura de notificação de evento do Amazon RDS.

AWS CLI

Para remover um identificador de origem de uma assinatura de notificação de eventos do Amazon RDS, use o comando da AWS CLI `remove-source-identifier-from-subscription`. Inclua os seguintes parâmetros necessários:

- `--subscription-name`
- `--source-identifier`

Example

O exemplo a seguir remove o identificador de origem `mysqldb` da assinatura `myrdseventsubscription`.

Para Linux, macOS ou Unix:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Para Windows:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

solicitações de

Para remover um identificador de origem de uma assinatura de notificação de eventos do Amazon RDS, use o comando `RemoveSourceIdentifierFromSubscription` da API do Amazon RDS. Inclua os seguintes parâmetros necessários:

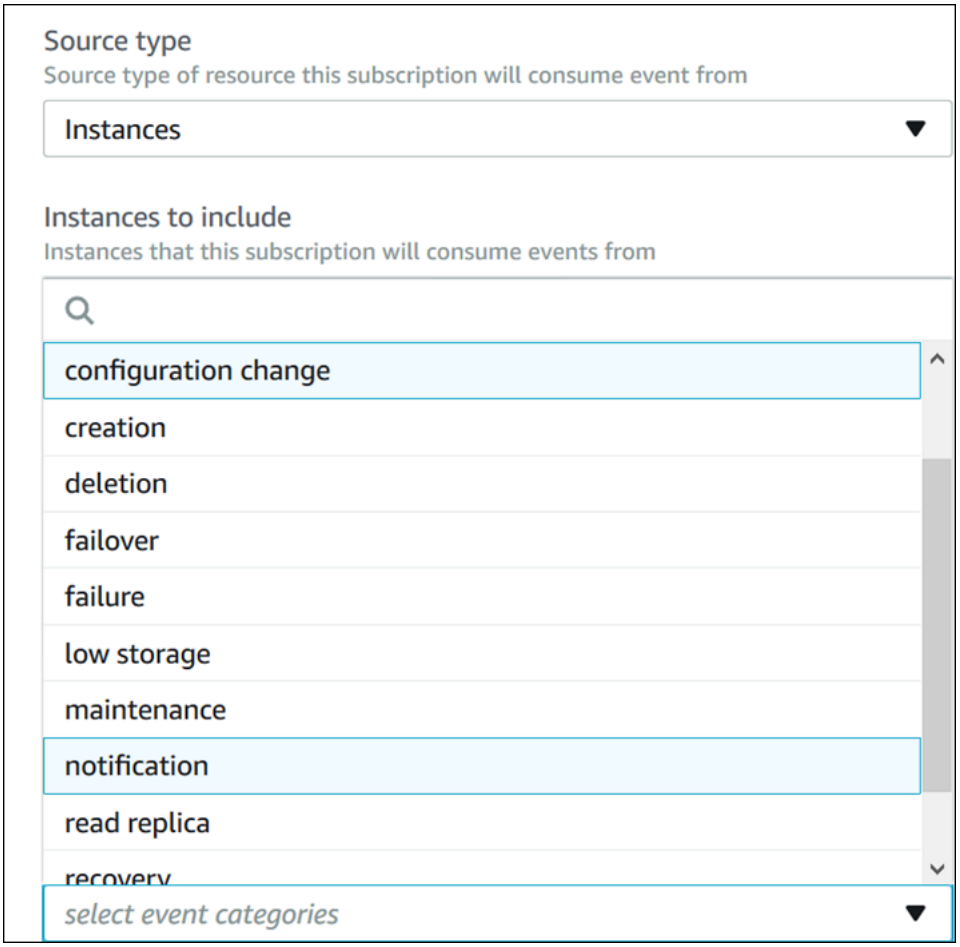
- `SubscriptionName`
- `SourceIdentifier`

Listar as categorias de notificação de evento do Amazon RDS

Todos os eventos para um tipo de recurso são agrupados em categorias. Para visualizar a lista de categorias disponíveis, use os seguintes procedimentos.

Console

Quando você cria ou modifica uma assinatura de notificação de evento, as categorias de eventos são exibidas no console do Amazon RDS. Para obter mais informações, consulte Modificar uma assinatura de notificação de evento do Amazon RDS.



AWS CLI

Para listar as categorias de notificação de evento do Amazon RDS, use o comando da AWS CLI `describe-event-categories`. Esse comando não possui parâmetros necessários.

Example

```
aws rds describe-event-categories
```

API

Para listar as categorias de notificação de evento do Amazon RDS, use o comando `DescribeEventCategories` da API do Amazon RDS. Esse comando não possui parâmetros necessários.

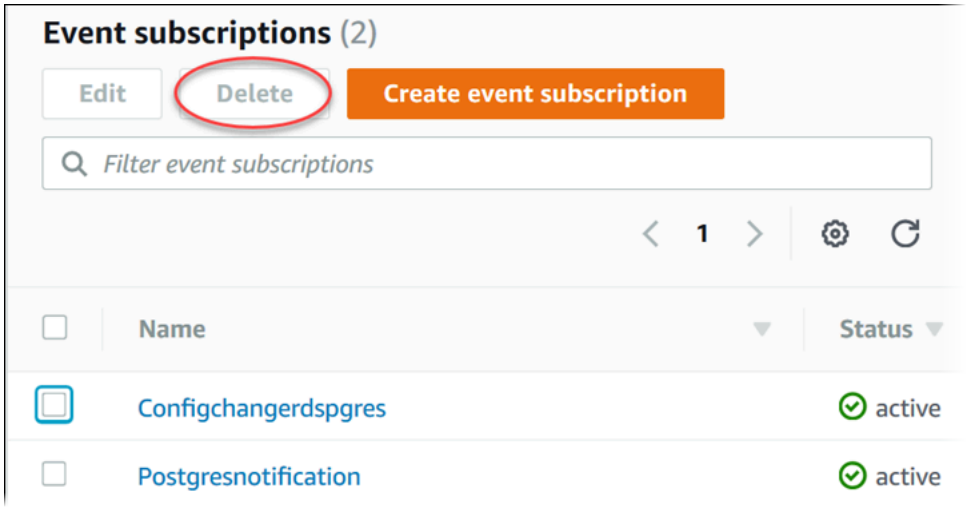
Excluir uma assinatura de notificação de evento do Amazon RDS

É possível excluir uma assinatura quando não precisar mais dela. Todos os assinantes do tópico não receberão mais notificações de evento especificadas pela assinatura.

Console

Para excluir uma assinatura de notificação de evento do Amazon RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha DB Event Subscriptions (Assinaturas de eventos de banco de dados).
3. No painel My DB Event Subscriptions (Minhas assinaturas de eventos de banco de dados), escolha a assinatura que deseja excluir.
4. Escolha Excluir.
5. O console do Amazon RDS indica que a assinatura está sendo excluída.



AWS CLI

Para excluir uma assinatura de notificação de evento do Amazon RDS, use o comando da AWS CLI `delete-event-subscription`. Inclua o seguinte parâmetro necessário:

- `--subscription-name`

Example

O exemplo a seguir exclui a assinatura `myrdssubscription`.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Para excluir uma assinatura de notificação de evento do Amazon RDS, use o comando `DeleteEventSubscription` da API do RDS. Inclua o seguinte parâmetro necessário:

- `SubscriptionName`

Criar uma regra que é acionada em um evento do Amazon Aurora

Usando o Amazon EventBridge, é possível automatizar serviços da AWS e responder aos eventos do sistema, como problemas de disponibilidade da aplicação ou alterações de recursos.

Tópicos

- Tutorial: Registro de alterações de estado de uma instância de banco de dados usando o Amazon EventBridge

Tutorial: Registro de alterações de estado de uma instância de banco de dados usando o Amazon EventBridge

Neste tutorial, você pode criar uma função do AWS Lambda que registra as alterações de estado para uma instância do . Depois, crie uma regra que execute a função sempre que houver uma alteração de estado de uma instância de banco de dados do RDS existente. O tutorial pressupõe que você tem uma pequena instância de teste em execução que você pode desligar temporariamente.

> **Important**
>
> Não execute este tutorial em uma instância de banco de dados de produção em execução.

Tópicos

- Etapa 1: Criar uma função do AWS Lambda
- Etapa 2: Criar uma regra
- Etapa 3: Testar a regra

Etapa 1: Criar uma função do AWS Lambda

Crie uma função Lambda para registrar em log os eventos de alteração de estado. Você especifica essa função quando cria sua regra.

Como criar uma função do Lambda

1. Abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/.
2. Se você estiver começando a usar o Lambda, verá uma página de boas-vindas. Escolha Get Started Now (Começar agora). Do contrário, escolha Create function (Criar função).

3. Escolha Author from scratch.
4. Na página Create function (Criar função), faça o seguinte:
   a. Digite um nome e uma descrição para a função Lambda. Por exemplo, atribua à função o nome **`RDSInstanceStateChange`**.
   b. Em Runtime (Tempo de execução), selecione Node.js 14x.
   c. Em Architecture (Arquitetura), escolha x86_64.
   d. Em Execution role (Perfil de execução), realize um dos seguintes procedimentos:
      - Escolha Create a new role with basic Lambda permissions (Criar uma nova função com permissões básicas do Lambda).
      - Em Existing role (Perfil existente), escolha Use an existing role (Usar um perfil existente). Escolha o perfil que deseja usar.
   e. Escolha Create function.
5. Na página RDSinstAncestateChange, faça o seguinte:
   a. Em Code source (Fonte do código), selecione index.js.
   b. No painel de index.js, exclua o código existente.
   c. Insira o seguinte código:

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. Escolha Deploy (Implantar).

Etapa 2: Criar uma regra

Crie uma regra para executar sua função do Lambda sempre que executar uma instância do Amazon RDS.

Como criar a regra do EventBridge

1. Abra o console do Amazon EventBridge em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Regras.

3. Escolha Create rule.
4. Insira um nome e uma descrição para a regra. Por exemplo, digite **`RDSInstanceStateChangeRule`**.
5. Escolha Rule with an event pattern (Regra com padrão de eventos), depois selecione Next (Próximo).
6. Em Origem do evento, escolha Eventos da AWS ou eventos de parceiro do EventBridge.
7. Role para baixo até a seção Event pattern (Padrão de eventos).
8. Para Origem do evento, escolha Serviços da AWS.
9. Em Serviço da AWS, escolha Relational Database Service (RDS).
10. Em Event type (Tipo de evento), escolha RDS DB Instance Event (Evento de instância de banco de dados do RDS).
11. Mantenha o padrão de eventos predefinido. Escolha Próximo.
12. Em Target types (Tipos de destino), escolha AWS service (Serviço da AWS).
13. Para Selecionar um destino, escolha Função do Lambda.
14. Em Function (Função), selecione a função do Lambda que você criou. Em seguida, escolha Próximo.
15. Em Configure tags (Configurar etiquetas), escolha Next (Próximo).
16. Revise as etapas da sua regra. Em seguida, escolha Create rule (Criar regra).

Etapa 3: Testar a regra

Para testar sua regra, desligue uma instância de banco de dados do RDS. Depois de esperar alguns minutos para a instância ser inicializada e executada, verifique se a sua função do Lambda foi chamada.

Como testar a regra ao interromper uma instância de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Interrupção de uma instância de banco de dados do RDS.
3. Abra o console do Amazon EventBridge em https://console.aws.amazon.com/events/.
4. No painel de navegação, escolha Rules (Regras) e escolha o nome da regra criada por você.
5. Em Detalhes da regra, escolha Monitoramento.

O sistema redireciona você para o console do Amazon CloudWatch. Se você não for redirecionado, clique em Visualizar as métricas no CloudWatch.

6. Em All metrics (Todas as métricas), escolha o nome da regra que você criou.

   O gráfico deve indicar que a regra foi invocada.

7. No painel de navegação, escolha Log groups (Grupos de logs).
8. Escolha o nome do grupo de logs para a sua função do Lambda (/aws/lambda/***nome-da-função***).
9. Escolha o nome do fluxo de logs para visualizar os dados fornecidos pela função para a instância que você iniciou. Será exibido um resultado semelhante ao seguinte:

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

   Para obter mais exemplos de eventos do RDS no formato JSON, consulte Visão geral dos eventos para Aurora.

10. (Opcional) Ao terminar, você poderá abrir o console do Amazon RDS e iniciar a instância interrompida.

Categorias de eventos e mensagens de eventos para o Amazon RDS

O Amazon RDS gera um número significativo de eventos em categorias nas quais você pode fazer uma assinatura usando o console do Amazon RDS, a AWS CLI ou a API.

Tópicos

- Eventos de cluster de banco de dados
- Eventos de snapshot de cluster de banco de dados
- Eventos de instância de banco de dados
- Eventos de grupo de parâmetros de banco de dados
- Eventos de grupos de segurança de banco de dados
- Eventos de grupo de fragmentos de banco de dados
- Eventos do RDS Proxy
- Eventos de implantação azul/verde

Eventos de cluster de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um cluster de banco de dados é o tipo de origem.

> **Note**
>
> Nenhuma categoria de evento existe para Aurora Serverless no tipo de evento do cluster de banco de dados. Os eventos Aurora sem servidores variam de RDS-EVENT-0141 a RDS-EVENT-0149.

Categoria	ID do evento do RDS	Mensagem	Observações
alteração de configuração	RDS-EVENT-0016	Redefina as credenciais principais.	Nenhum
alteração de configuração	RDS-EVENT-0179	Os fluxos de atividades do banco de dados são	Para ter mais informaçõ es, consulte Monitorar o Amazon Aurora com o

Categoria	ID do evento do RDS	Mensagem	Observações
		iniciados no cluster de banco de dados.	recurso Database Activity Streams.
alteração de configuração	RDS-EVENT-0180	Os fluxos de atividades do banco de dados são interrompidos no cluster de banco de dados.	Para ter mais informaçõ es, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.
criação	RDS-EVENT-0170	Cluster de banco de dados criado.	Nenhum
exclusão	RDS-EVENT-0171	Cluster de banco de dados excluído.	Nenhum
failover	RDS-EVENT-0069	Falha no failover do cluster, verifique a integridade das instâncias do cluster e tente novamente.	Nenhum
failover	RDS-EVENT-0070	Promover novamente a primária anterior: *nome*.	Nenhum
failover	RDS-EVENT-0071	Failover concluído para a instância de banco de dados: *nome*.	Nenhum
failover	RDS-EVENT-0072	Iniciou o mesmo failover da AZ para a instância de banco de dados: *nome*.	Nenhum
failover	RDS-EVENT-0073	Iniciou o failover cruzado da AZ para a instância de banco de dados: *nome*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0083	O Amazon RDS não conseguiu criar credencia is para acessar o bucket do Amazon S3 para o cluster de banco de dados*nome*. Isso ocorre porque o perfil do IAM de ingestão de snapshots do S3 não está configurado corretame nte na conta ou porque o bucket especificado do Amazon S3 não pode ser encontrado. Consulte a seção de solução de problemas na documenta ção do Amazon RDS para obter mais detalhes.	Para ter mais informações, consulte Migração física do MySQL usando o Percona XtraBackup e o Amazon S3 .
falha	RDS-EVENT-0143	O cluster de banco de dados não escalou de *unidades* para *unidades* por este motivo: *motivo*.	Falha na escalabilidade do cluster de banco de dados do Aurora Serverless.
falha	RDS-EVENT-0354	Não é possível criar o cluster de banco de dados devido a recursos incompatíveis. *mensagem*.	A *mensagem* inclui detalhes sobre a falha.
falha	RDS-EVENT-0355	O cluster de banco de dados não pode ser criado devido a limites insuficientes de recursos. *mensagem*.	A *mensagem* inclui detalhes sobre a falha.

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0387	Falha ao particionar instâncias de banco de dados no cluster de banco de dados *name* para aplicação de patches.	As atualizações do sistema operacional para as instâncias de banco de dados no cluster de banco de dados falharam.
Failover global	RDS-EVENT-0181	A transição global para o cluster de banco de dados *nome* na região *nome* foi iniciada.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). O processo pode ser atrasado devido à execução de outras operações no cluster de banco de dados.
Failover global	RDS-EVENT-0182	O antigo cluster de banco de dados primário *nome* na região *nome* foi encerrado com sucesso.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). A instância primária antiga no banco de dados global não aceita gravações. Todos os volumes são sincronizados.

Categoria	ID do evento do RDS	Mensagem	Observações
Failover global	RDS-EVENT-0183	Aguardar a sincroniz ação de dados entre os membros do cluster global. O cluster atual está atrasado em relação ao cluster de banco de dados primário: *reason*.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). Está ocorrendo um atraso de replicação durante a fase de sincronização do failover de banco de dados global.
Failover global	RDS-EVENT-0184	O novo do cluster de banco de dados primário *nome* na região *nome* foi promovido com sucesso.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). A topologia de volumes do banco de dados global é restabelecida com o novo volume primário.

Categoria	ID do evento do RDS	Mensagem	Observações
Failover global	RDS-EVENT-0185	A transição global para o cluster de banco de dados *nome* na região *nome* foi concluída.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). A transição de banco de dados global está concluída no cluster de banco de dados primário. Réplicas podem demorar muito para ficarem on-line após a conclusão do failover.
Failover global	RDS-EVENT-0186	A transição global para o cluster de banco de dados *nome* na região *nome* foi cancelada.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”).
Failover global	RDS-EVENT-0187	A transição global para o cluster de banco de dados *nome* na região *nome* apresentou falha.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”).
Failover global	RDS-EVENT-0238	O failover global para o cluster de banco de dados *nome* na região *nome* foi concluído.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
Failover global	RDS-EVENT-0239	O failover global para o cluster de banco de dados *nome* na região *nome* falhou.	Nenhum
Failover global	RDS-EVENT-0240	Começou a ressincronizar membros do cluster de banco de dados *nome* na região *nome* após o failover global.	Nenhum
Failover global	RDS-EVENT-0241	Terminou de ressincronizar membros do cluster de banco de dados *nome* na região *nome* após o failover global.	Nenhum
Failover global	RDS-EVENT-0397	O Aurora terminou de alterar o nome DNS para o qual o endpoint global do gravador é resolvido.	Nenhum
Failover global	RDS-EVENT-0423	Aguardando a sincroniz ação de dados com o cluster de banco de dados de destino. O cluster de banco de dados de destino atual está atrasado em relação ao cluster de banco de dados primário: `%s`.	Esse evento destina-se a uma operação de transição (anteriormente chamada de “failover planejado gerenciado”). Está ocorrendo um atraso de replicação durante a fase de sincronização do failover de banco de dados global.

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0156	O cluster de banco de dados tem uma atualizaç ão de versão secundária do mecanismo de banco de dados disponível.	Nenhum
manutenção	RDS-EVENT-0173	A versão do mecanismo do cluster de banco de dados foi atualizada.	A aplicação de patches do cluster de banco de dados foi concluída.
manutenção	RDS-EVENT-0174	O cluster do banco de dados está em um estado que não pode ser atualizad o.	Nenhum
manutenção	RDS-EVENT-0176	A versão principal do mecanismo do cluster de banco de dados foi atualizada.	Nenhum
manutenção	RDS-EVENT-0177	A atualização do cluster de banco de dados está em andamento.	Nenhum
manutenção	RDS-EVENT-0286	A atualização da versão *version_number* do mecanismo de cluster de banco de dados foi iniciada. O cluster permanece online.	Nenhum
manutenção	RDS-EVENT-0287	Requisito de atualização do sistema operacional detectado.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0288	Início da atualização do sistema operacional do cluster.	Nenhum
manutenção	RDS-EVENT-0289	Atualização do sistema operacional de cluster concluída.	Nenhum
manutenção	RDS-EVENT-0290	O cluster de banco de dados foi corrigido: versão de origem *número_da_versão* => *número_da_nova_versão* .	Nenhum
manutenção	RDS-EVENT-0363	Preparação do upgrade em andamento: *cluster_name*	As pré-verificações do upgrade foram iniciadas para o cluster de banco de dados.
manutenção	RDS-EVENT-0388	Iniciando patches offline para instâncias de banco de dados na partição *name*/*name* para o cluster de banco de dados *name*: *partition_n* .	Iniciando as atualizações do sistema operacional para as instâncias de banco de dados no cluster de banco de dados.
manutenção	RDS-EVENT-0389	Não foi possível atualizar o sistema operacional do cluster de banco de dados. Você pode aguardar a próxima janela de manutenção ou atualizar o sistema operacional do cluster de banco de dados manualmente.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0424	O cluster de banco de dados *nome* está executando a versão *versão*, que é superior à versão de atualizaç ão de destino *versão* para o cluster global. Não recomendamos ter um cluster secundário em uma versão posterior à do cluster global, pois isso pode causar problemas durante o failover ou a transição. Considere a possibilidade de atualizar o cluster global para que haja correspondência.	Nenhum
notificação	RDS-EVENT-0076	Falha ao migrar de *nome* para *nome*. Motivo: *motivo*.	Falha na migração para um cluster de bancos de dados Aurora.
notificação	RDS-EVENT-0077	Falha ao converter *nome*.*nome* para InnoDB. Motivo: *motivo*.	Uma tentativa de converter uma tabela do banco de dados de origem para o InnoDB falhou durante a migração para um cluster de bancos de dados Aurora.

Categoria	ID do evento do RDS	Mensagem	Observações
notificação	RDS-EVENT-0085	Não foi possível atualizar o cluster de banco de dados *nome* porque a instância *nome* tem um status de *nome*. Resolva o problema ou exclua a instância e tente novamente.	Ocorreu um erro ao tentar aplicar um patch no cluster de Aurora banco de dados. Verifique o status da instância, resolva o problema e tente novamente. Para ter mais informações, consulte Manutenção de um cluster de banco de dados do Amazon Aurora.
notificação	RDS-EVENT-0141	Ajustar a escala do cluster de banco de dados de *unidades* para *unidades* por este motivo: *motivo*.	Escalonamento iniciado para o cluster de banco de dados do Aurora Serverles s.
notificação	RDS-EVENT-0142	O cluster de banco de dados foi escalado de *unidades* para *unidades*.	Escalonamento concluído para o cluster de banco de dados do Aurora Serverles s.
notificação	RDS-EVENT-0144	O cluster de banco de dados está sendo pausado.	Uma pausa automática foi iniciada no cluster de banco de dados do Aurora Sem Servidor.
notificação	RDS-EVENT-0145	O cluster de banco de dados está pausado.	O cluster de banco de dados do Aurora Serverles s foi pausado.
notificação	RDS-EVENT-0146	A pausa cancelada para o cluster de banco de dados.	A pausa foi cancelada para o cluster de banco de dados Aurora Serverless.

Categoria	ID do evento do RDS	Mensagem	Observações
notificação	RDS-EVENT-0147	O cluster de banco de dados está sendo retomado.	Uma operação de retomada foi iniciada para o cluster de banco de dados do Aurora Serverless.
notificação	RDS-EVENT-0148	O cluster de banco de dados foi retomado.	A operação retomada foi concluída para o cluster de banco de dados do Aurora Serverless.
notificação	RDS-EVENT-0149	O cluster de banco de dados foi escalado de *unidades* para *unidades*, mas a o ajuste de escala não foi perfeito por este motivo: *motivo*.	Escalabilidade simples concluída com a opção forçada para o cluster de banco de dados do Aurora Serverless. Conexões podem ser interrompidas conforme necessário.
notificação	RDS-EVENT-0150	O cluster de banco de dados parou.	Nenhum
notificação	RDS-EVENT-0151	O cluster de banco de dados iniciou.	Nenhum
notificação	RDS-EVENT-0152	Falha ao interromper o cluster de banco de dados.	Nenhum
notificação	RDS-EVENT-0153	O cluster de banco de dados está sendo iniciado porque excede o tempo máximo permitido para permanecer parado.	Nenhum
notificação	RDS-EVENT-0172	Cluster renomeado de *nome* para *nome*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
notificação	RDS-EVENT-0234	Falha na tarefa de exportação.	Falha na tarefa de exportação do cluster de banco de dados.
notificação	RDS-EVENT-0235	Tarefa de exportação cancelada.	A tarefa de exportação de cluster de banco de dados foi cancelada.
notificação	RDS-EVENT-0236	Tarefa de exportação concluída.	Tarefa de exportação de cluster de banco de dados concluída.
notificação	RDS-EVENT-0386	As instâncias de banco de dados no cluster de banco de dados *name* foram particionadas: *list_of_partitions* . O cluster de banco de dados está online.	As atualizações do sistema operacional para as instâncias de banco de dados no cluster de banco de dados foram bem-sucedidas.
notificação	RDS-EVENT-0426	O RDS não pode configurar o cluster de banco de dados *nome* como origem de replicação devido a transações ociosas ou de longa execução. Aguarde a conclusão das transações ou cancele-as e tente novamente.	Nenhum

Eventos de snapshot de cluster de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um snapshot de cluster de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Mensagem	Observações
backup	RDS-EVENT-0074	Criação de um snapshot manual do cluster.	Nenhum
backup	RDS-EVENT-0075	Snapshot de cluster manual criado.	Nenhum
notificação	RDS-EVENT-0162	Falha na tarefa de exportação do snapshot do cluster.	Nenhum
notificação	RDS-EVENT-0163	A tarefa de exportação do snapshot do cluster foi cancelada.	Nenhum
notificação	RDS-EVENT-0164	A tarefa de exportação do snapshot do cluster foi concluída.	Nenhum
backup	RDS-EVENT-0168	Criando snapshot de cluster automatizado.	Nenhum
backup	RDS-EVENT-0169	Snapshot de cluster automatizado criado.	Nenhum
falha	RDS-EVENT-0489	Tarefa de exportação *nome*: falha na extração da tabela *nome* devido à *mensagem*.	Esse evento não é aceito pela exportação mais rápida.
notificação	RDS-EVENT-0491	Tarefa de exportação *nome*: a tabela *nome* será exportada em um modo de segmento único mais lento. Aguarde um atraso	Esse evento não é aceito pela exportação mais rápida.

Categoria	ID do evento do RDS	Mensagem	Observações
		na conclusão da exportaçã o.	
notificação	RDS-EVENT-0492	Tarefa de exportação *nome*: a tabela *nome* está em andamento e o tamanho exportado é *número* GB.	Esse evento será enviado apenas para tabelas que levem mais de uma hora para serem concluídas. Esse evento não é aceito pela exportação mais rápida.
notificação	RDS-EVENT-0493	A tarefa de exportaçã o *nome* está no estágio *nome*.	Nenhum
		Tarefa de exportaçã o *nome*: o andamento da exportação está em *número*% com *número* tabela(s) em andamento e *número* tabela(s) concluída (s). O tamanho total das tabelas finalizadas é *número* GB.	Esse evento não é aceito pela exportação mais rápida.

Eventos de instância de banco de dados

As tabelas a seguir mostram a categoria de evento e uma lista de eventos quando uma instância de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Mensagem	Observações
disponibilidade	RDS-EVENT-0004	Desligamento da instância de banco de dados.	Nenhum
disponibilidade	RDS-EVENT-0006	A instância de banco de dados foi reiniciada.	Nenhum
disponibilidade	RDS-EVENT-0022	Erro ao reiniciar o mysql: *mensagem*.	Ocorreu um erro ao reiniciar o Aurora MySQL ou o RDS para MariaDB.
disponibilidade	RDS-EVENT-0419	O Amazon RDS não pôde acessar a chave de criptografia KMS para a instância de banco de dados *nome*. Esse banco de dados será colocado em um estado inacessível. Consulte a seção de solução de problemas na documentação do Amazon RDS para obter mais detalhes.	Nenhum
retrocesso	RDS-EVENT-0131	A janela real de retrocesso é menor que a janela de retrocesso de destino que você especificou. Considere reduzir o número de horas na sua janela de retrocesso de destino.	Para ter mais informações sobre retrocesso, consulte Retroceder um cluster de banco de dados Aurora.

Categoria	ID do evento do RDS	Mensagem	Observações
retrocesso	RDS-EVENT-0132	A janela real de retrocesso é igual à janela de retrocesso de destino.	Nenhum
alteração de configuração	RDS-EVENT-0011	Atualização para usar DBParameterGroup *name*.	Nenhum
alteração de configuração	RDS-EVENT-0012	Aplicando modificação à classe de instância de banco de dados.	Nenhum
alteração de configuração	RDS-EVENT-0014	Conclusão da aplicação de modificação à classe de instância de banco de dados.	Nenhum
alteração de configuração	RDS-EVENT-0017	Concluída aplicação de modificação ao armazenamento alocado.	Nenhum
alteração de configuração	RDS-EVENT-0025	Concluída a aplicação de modificação para converter em uma instância de banco de dados multi-AZ.	Nenhum
alteração de configuração	RDS-EVENT-0029	Concluída a aplicação da modificação para converter em uma instância de banco de dados padrão (single-AZ).	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
alteração de configuração	RDS-EVENT-0033	Há *`número`* usuários que correspondem ao nome de usuário principal; apenas redefinindo aquele que não está vinculado a um host específico.	Nenhum
alteração de configuração	RDS-EVENT-0067	Não foi possível redefinir sua senha. Informações de erro: *`mensagem`*.	Nenhum
alteração de configuração	RDS-EVENT-0078	Intervalo de monitoramento alterado para *`número`*.	A configuração de Monitoramento avançado foi alterada.
alteração de configuração	RDS-EVENT-0092	Conclusão da atualização do grupo de parâmetros do banco de dados.	Nenhum
criação	RDS-EVENT-0005	Instância de banco de dados criada.	Nenhum
exclusão	RDS-EVENT-0003	A instância de banco de dados foi excluída.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0035	Instância de banco de dados colocada no *estado*. *mensagem*.	A instância de banco de dados tem parâmetros inválidos. Por exemplo, se a instância de banco de dados não pôde ser iniciada porque um parâmetro relacionado à memória está definido como um valor muito alto para essa classe de instância, a ação seria modificar o parâmetro da memória e reinicializar a instância de banco de dados.
falha	RDS-EVENT-0036	Instância do banco de dados em *estado*. *mensagem*.	A instância de banco de dados está em uma rede incompatível. Alguns dos IDs de sub-rede especific ados são inválidos ou não existem.

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0079	O Amazon RDS não conseguiu criar credencia is para o monitoram ento avançado e esse recurso foi desabilitado. Provavelmente, isso ocorre porque rds-monitoring-rol e não está presente e foi configurado corretamente em sua conta. Consulte a seção de solução de problemas na documenta ção do Amazon RDS para obter mais detalhes.	O Monitoramento avançado não pode ser habilitado sem o perfil do IAM de monitoramento avançado. Para obter informações sobre como criar o perfil do IAM, consulte Como criar uma função do IAM para o monitoramento avançado do Amazon RDS.
falha	RDS-EVENT-0080	O Amazon RDS não conseguiu configurar o monitoramento avançado em sua instância: *nome* e esse recurso foi desabilit ado. Provavelmente, isso ocorre porque rds-monitoring-role não está presente e foi configura do corretamente em sua conta. Consulte a seção de solução de problemas na documentação do Amazon RDS para obter mais detalhes.	O Monitoramento avançado foi desabilitado porque ocorreu um erro durante a alteração da configura ção. É provável que o perfil do IAM de monitoramento avançado esteja configura do incorretamente. Para obter informações sobre como criar o perfil do IAM de monitoramento avançado, consulte Como criar uma função do IAM para o monitoramento avançado do Amazon RDS.

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0082	O Amazon RDS não conseguiu criar credencia is para acessar o bucket do Amazon S3 para a instância de banco de dados *nome*. Isso ocorre porque o perfil do IAM de ingestão de snapshots do S3 não está configurado corretamente na conta ou porque o bucket especific ado do Amazon S3 não pode ser encontrado. Consulte a seção de solução de problemas na documentação do Amazon RDS para obter mais detalhes.	O Aurora não pôde copiar dados de backup de um bucket do Amazon S3. É provável que as permissõe s para o Aurora acessar o bucket do Amazon S3 estejam configuradas incorretamente. Para ter mais informações, consulte Migração física do MySQL usando o Percona XtraBackup e o Amazon S3 .
falha	RDS-EVENT-0353	A instância de banco de dados não pode ser criada devido a limites insuficientes de recursos. *mensagem*.	A *mensagem* inclui detalhes sobre a falha.

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0418	O Amazon RDS não consegue acessar a chave de criptografia KMS para a instância de banco de dados *nome*. Provavelmente, a chave está desabilitada ou o Amazon RDS não pode acessá-la. Se isso continuar, o banco de dados será colocado em um estado inacessível. Consulte a seção de solução de problemas na documentação do Amazon RDS para obter mais detalhes.	Nenhum
falha	RDS-EVENT-0420	Agora o Amazon RDS pode acessar com êxito a chave de criptografia KMS para a instância de banco de dados *nome*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
armazenamento baixo	RDS-EVENT-0007	O armazenamento alocado foi esgotado. Aloque armazenamento adicional para resolver.	O armazenamento alocado para a instância de banco de dados foi consumido. Para resolver esse problema, aloque armazenamento adicional para a instância de banco de dados. Para ter mais informações, consulte Perguntas frequentes do RDS. Você pode monitorar o espaço de armazenamento para uma instância de banco de dados usando a métrica Free Storage Space (Espaço de armazenamento livre).
armazenamento baixo	RDS-EVENT-0089	A capacidade de armazenamento livre para instância de banco de dados: *nome* está baixa em *porcentagem* do armazenamento provisionado [Armazenamento provisionado: *tamanho*, armazenamento livre: *tamanho*]. Talvez você queira aumentar o armazenamento provisionado para resolver esse problema.	A instância de banco de dados consumiu mais de 90% do armazenamento alocado. Você pode monitorar o espaço de armazenamento para uma instância de banco de dados usando a métrica Free Storage Space (Espaço de armazenamento livre).

Categoria	ID do evento do RDS	Mensagem	Observações
armazenamento baixo	RDS-EVENT-0227	O armazenamento do cluster do Aurora está arriscadamente baixo, com apenas *quantidade* terabytes restantes. Tome medidas para reduzir a carga de armazenamento no cluster.	O subsistema de armazenamento do Aurora está com pouco espaço.
manutenção	RDS-EVENT-0026	Aplicação de patches off-line à instância de banco de dados.	Está ocorrendo a manutenção offline da instância de banco de dados. Atualmente, a instância do banco de dados está indisponível.
manutenção	RDS-EVENT-0027	Concluída a aplicação de patches off-line na instância de banco de dados.	A manutenção offline da instância de banco de dados está completa. A instância de banco de dados já está disponível.
manutenção	RDS-EVENT-0047	Instância de banco de dados corrigida.	Nenhum
manutenção	RDS-EVENT-0155	A instância de banco de dados tem uma atualização de versão secundária do mecanismo de banco de dados disponível.	Nenhum
manutenção	RDS-EVENT-0178	A atualização da instância do banco de dados está em andamento.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0422	O RDS substituirá o host da instância de banco de dados *nome* devido a uma ação de manutenção pendente.	Nenhum
notificação	RDS-EVENT-0044	*message*	Essa é uma notificação emitida pelo operador. Para obter mais informações, consulte a mensagem do evento.
notificação	RDS-EVENT-0048	Atraso na atualização do mecanismo de banco de dados, pois essa instância tem réplicas de leitura que precisam ser atualizadas primeiro.	O patch da instância de banco de dados foi atrasado.
notificação	RDS-EVENT-0087	Instância de banco de dados foi interrompida.	Nenhum
notificação	RDS-EVENT-0088	A instância de banco de dados iniciou.	Nenhum
notificação	RDS-EVENT-0365	Os arquivos de fuso horário foram atualizados. Reinicie a instância do RDS para que as alterações entrem em vigor.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
notificação, tecnologia sem servidor	RDS-EVENT-0370	Pausa iniciada para a instância de banco de dados.	Uma nova tentativa de pausar uma instância de banco de dados do Aurora Serverless v2 ociosa foi iniciada.
notificação, tecnologia sem servidor	RDS-EVENT-0371	A pausa foi cancelada para a instância de banco de dados.	Uma tentativa de pausar uma instância de banco de dados do Aurora Serverless v2 ociosa não teve êxito devido à workload.
notificação, tecnologia sem servidor	RDS-EVENT-0372	Instância de banco de dados pausada com êxito.	A instância de banco de dados do Aurora Serverless v2 foi pausada.
notificação, tecnologia sem servidor	RDS-EVENT-0373	Retomada iniciada para a instância de banco de dados.	A instância de banco de dados do Aurora Serverless v2 começou a ser retomada devido a uma nova workload, atividade administrativa ou de manutenção.
notificação, tecnologia sem servidor	RDS-EVENT-0374	Instância de banco de dados retomada com êxito.	A instância de banco de dados do Aurora Serverless v2 foi retomada.

Categoria	ID do evento do RDS	Mensagem	Observações
notificação	RDS-EVENT-0385	A topologia do cluster é atualizada.	Há alterações de DNS no cluster de banco de dados para a instância de banco de dados. Isso inclui quando novas instância s de banco de dados são adicionadas ou excluídas, ou quando há um failover.
notificação, banco de dados global	RDS-EVENT-0390	A tentativa de bloquear gravações para o cluster de banco de dados *cluster_id* na região *region_id* foi bem-sucedida.	O Aurora começou a bloquear as gravações na camada de armazenam ento em preparação para a transição ou o failover de um Aurora Global Database.
notificação, banco de dados global	RDS-EVENT-0391	A tentativa de bloquear gravações para o cluster de banco de dados *cluster_id* na região *region_id* expirou.	O Aurora não conseguiu bloquear as gravações na camada de armazenam ento em preparação para a transição ou o failover de um Aurora Global Database. A transição ou o failover continuarão, mas talvez seja necessário recuperar dados gravados recentemente do snapshot do cluster primário original.

Categoria	ID do evento do RDS	Mensagem	Observações
réplica de leitura	RDS-EVENT-0045	A replicação foi interromp ida.	Essa mensagem é exibida quando ocorre um erro durante a replicação. Para determinar o tipo de erro, consulte Solucionar problemas de uma réplica de leitura do MySQL.
réplica de leitura	RDS-EVENT-0046	A réplica de leitura foi retomada.	Essa mensagem aparece quando você cria uma réplica de leitura pela primeira vez, ou como uma mensagem de monitoram ento confirmando que a replicação está funcionan do corretamente. Se essa mensagem vem depois de uma notificação `RDS-EVENT-0045` , a replicaçã o foi retomada após um erro ou depois que a replicação foi interrompida.
réplica de leitura	RDS-EVENT-0057	O streaming de replicação foi encerrado.	Nenhum
recuperação	RDS-EVENT-0020	A recuperação da instância de banco de dados começou. O tempo de recuperação variará dependendo da quantidad e de dados a serem recuperados.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
recuperação	RDS-EVENT-0021	A recuperação da instância de banco de dados está completa.	Nenhum
recuperação	RDS-EVENT-0023	Solicitação de snapshot emergente: *mensagem*.	Um backup manual foi solicitado, mas o Amazon RDS está no processo de criar um snapshot de banco de dados. Envie o pedido novamente depois que o Amazon RDS tiver concluído o snapshot de banco de dados.
recuperação	RDS-EVENT-0052	A recuperação da instância multi-AZ foi iniciada.	O tempo de recuperação variará dependendo da quantidade de dados a serem recuperados.
recuperação	RDS-EVENT-0053	A recuperação da instância multi-AZ foi concluída . Failover ou ativação pendentes.	Essa mensagem indica que o Amazon RDS preparou a instância de banco de dados para iniciar um failover para a instância secundária, se necessário.
recuperação	RDS-EVENT-0361	A recuperação da instância de banco de dados em espera começou.	A instância de banco de dados em espera é recriada durante o processo de recuperação. O desempenho do banco de dados é afetado durante o processo de recuperaç ão.

Categoria	ID do evento do RDS	Mensagem	Observações
recuperação	RDS-EVENT-0362	A recuperação da instância de banco de dados em espera foi concluída.	A instância de banco de dados em espera é recriada durante o processo de recuperação. O desempenho do banco de dados é afetado durante o processo de recuperação.
restauração	RDS-EVENT-0019	Restauração da instância de banco de dados *nome* para *nome*.	A instância de banco de dados foi restaurada a partir de um backup point-in-time.
patches de segurança	RDS-EVENT-0230	A atualização do sistema está disponível para a instância de banco de dados. Para obter informações sobre como aplicar atualizações, consulte “Como manter uma instância de banco de dados” no Guia do usuário do RDS.	Um novo patch de versão secundária do sistema operacional está disponível para sua instância de banco de dados. Para obter informações sobre a aplicação de atualizações, consulte Atualizações do sistema operacional para clusters de banco de dados do Aurora.

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0425	O Amazon RDS não pode realizar a atualização do sistema operacional porque não há endereços IP disponíveis nas sub-redes especificadas. Escolha sub-redes com endereços IP disponíveis e tente novamente.	Nenhum
manutenção	RDS-EVENT-0429	O Amazon RDS não pode realizar a atualização do sistema operacional porque não há capacidad e suficiente disponível para o tipo de instância *type* instância na zona de disponibilidade *zone*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0501	O certificado de servidor da instância de banco de dados do Amazon RDS exige a alternância por meio de uma ação de manutenção pendente.	O certificado de servidor da instância de banco de dados exige a alternânc ia por meio de uma ação de manutenção pendente. O Amazon RDS reinicializa seu banco de dados durante essa manutenção para concluir a alternância do certifica do. Para programar essa manutenção, vá até a guia Manutenção e backups e escolha Aplicar agora ou Agendar para a próxima janela de manutenção. Se a alteração não for programada, o Amazon RDS a aplicará automatic amente em sua janela de manutenção na data de aplicação automática mostrada em sua ação de manutenção.

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0502	O Amazon RDS programou uma alternância de certificados de servidor para a instância de banco de dados durante a próxima janela de manutenção. Essa manutenção exigirá a reinicialização do banco de dados.	Nenhum

Eventos de grupo de parâmetros de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um grupo de parâmetros de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Mensagem	Observações
alteração de configuração	RDS-EVENT-0037	Atualização do parâmetro *name* para *value* pelo método de aplicação *method*.	Nenhum

Eventos de grupos de segurança de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um grupo de segurança de banco de dados é o tipo de origem.

> **Note**
>
> Grupos de segurança de banco de dados são recursos do EC2-Classic. O EC2-Classic foi removido em 15 de agosto de 2022. Se ainda não migrou do EC2-Classic para uma VPC, recomendamos que você migre o mais rápido possível. Para ter mais informações,

consulte Migrate from EC2-Classic to a VPC (Migrar do EC2-Classic para uma VPC) no Guia do usuário do Amazon EC2 e o blog EC2-Classic Networking is Retiring – Here's How to Prepare (O EC2-Classic Networking será descontinuado. Veja como se preparar).

Categoria	ID do evento do RDS	Mensagem	Observações
alteração de configuração	RDS-EVENT-0038	Alteração aplicada ao grupo de segurança.	Nenhum
falha	RDS-EVENT-0039	Revogando a autorização como *usuário*.	O grupo de segurança de propriedade de *usuário* não existe. A autorização para o grupo de segurança foi revogada porque é inválida.

Eventos de grupo de fragmentos de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um grupo de fragmentos de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Mensagem	Observações
falha	RDS-EVENT-0324	Falha no trabalho de carregamento de dados.	O trabalho de carregamento de dados falhou pelo motivo que aparece na mensagem de erro.
manutenção	RDS-EVENT-0302	Sua ação é necessária para finalizar um trabalho pendente de divisão de fragmento *job_ID* para o ID de subcluster *number*	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
		no grupo de fragmentos de banco de dados *name*. Para concluir a operação, invoque o SQL: `SELECT rds_aurora.limitless_finalize_split_shard( job_ID);`	
manutenção	RDS-EVENT-0303	A finalização do trabalho de divisão de fragmento *job_ID* foi iniciada para o ID de subcluster *number* no grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0304	O trabalho de divisão de fragmento *job_ID* foi concluído para o ID de subcluster *number* no grupo de fragmentos de banco de dados *name*. Um novo fragmento com ID de subcluster *number* foi adicionado ao grupo de fragmentos de banco de dados *name*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0305	O trabalho de divisão de fragmento *job_ID* falhou para o ID de subcluster *number* no grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0366	O trabalho de divisão de fragmento *job_ID* foi iniciado para o ID de subcluster *number* no grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0367	O trabalho de adição de roteador *job_ID* foi iniciado no grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0368	O trabalho de adição de roteador *job_ID* foi concluído com êxito. Um novo fragmento com ID de subcluster *number* foi adicionado ao grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0369	O trabalho de adição de roteador *job_ID* falhou no grupo de fragmentos de banco de dados *name*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
manutenção	RDS-EVENT-0394	O trabalho de adição de roteador *job_ID* foi cancelado no grupo de fragmentos de banco de dados *name*.	Nenhum
manutenção	RDS-EVENT-0395	O trabalho de divisão de fragmento *job_ID* foi cancelado no grupo de fragmentos de banco de dados *name*.	Nenhum
notificação	RDS-EVENT-0321	Inicializando a infraestrutura para o trabalho de carregamento de dados.	Nenhum
notificação	RDS-EVENT-0322	O trabalho de carregamento de dados está em andamento.	Nenhum
notificação	RDS-EVENT-0323	Trabalho de carregamento de dados concluído.	Nenhum
notificação	RDS-EVENT-0325	Cancelando o trabalho de carregamento de dados conforme solicitação do cliente.	Nenhum
notificação	RDS-EVENT-0326	Trabalho de carregamento de dados cancelado.	Nenhum

Eventos do RDS Proxy

As tabelas a seguir mostram a categoria de evento e uma lista de eventos quando um proxy do RDS é o tipo de fonte.

Categoria	ID do evento do RDS	Mensagem	Observações
alteração de configuração	RDS-EVENT-0204	Proxy de banco de dados *nome* modificado pelo RDS.	Nenhum
alteração de configuração	RDS-EVENT-0207	O RDS modificou o endpoint do proxy de banco de dados *nome*.	Nenhum
alteração de configuração	RDS-EVENT-0213	O RDS detectou a adição da instância de banco de dados e a adicionou automaticamente ao grupo de destino do *nome* do proxy de banco de dados.	Nenhum
alteração de configuração	RDS-EVENT-0214	O RDS detectou a exclusão da instância de banco de dados *nome* e a removeu automaticamente do grupo de destino *nome* do proxy de banco de dados *nome*.	Nenhum
alteração de configuração	RDS-EVENT-0215	O RDS detectou a exclusão do cluster de banco de dados *nome* e o removeu automaticamente do grupo de destino *nome* do proxy de banco de dados *nome*.	Nenhum
criação	RDS-EVENT-0203	O RDS criou o proxy de banco de dados *nome*.	Nenhum
criação	RDS-EVENT-0206	O RDS criou o endpoint para *nome* para o proxy do banco de dados *nome*.	Nenhum

Categoria	ID do evento do RDS	Mensagem	Observações
exclusão	RDS-EVENT-0205	O RDS excluiu o proxy do banco de dados *nome*.	Nenhum
exclusão	RDS-EVENT-0208	O RDS excluiu o endpoint *nome* para o proxy do banco de dados *nome*.	Nenhum
falha	RDS-EVENT-0243	O RDS não conseguiu provisionar capacidad e para o *nome* do proxy porque não há endereços IP suficientes disponíveis em suas sub-redes: *nome*. Para resolver o problema, as sub-redes devem ter o número mínimo de endereços IP não usados, conforme recomendado na documentação do proxy do RDS.	Para determinar o número recomendado para sua classe de instância , consulte Planejar a capacidade de endereços IP.
falha	RDS-EVENT-0275	O RDS limitou algumas conexões com o proxy de banco de dados *nome*. O número de solicitações de conexão simultâneas do cliente para o proxy excedeu o limite.	Nenhum

Eventos de implantação azul/verde

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando uma implantação azul/verde é o tipo de origem.

Para ter mais informações sobre implantações azul/verde, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Categoria	ID do evento do Amazon RDS	Mensagem	Observações
criação	RDS-EVENT-0244	Tarefas de implantação azul/verde concluídas. Você pode fazer outras modificações nos bancos de dados no ambiente verde ou fazer a transição da implantação.	Nenhum
falha	RDS-EVENT-0245	Falha na criação da implantação azul/verde devido a *reason*.	Nenhum
exclusão	RDS-EVENT-0246	Implantação azul/verde excluída.	Nenhum
notificação	RDS-EVENT-0247	A transição do *azul* para o *verde* começou.	Nenhum
notificação	RDS-EVENT-0248	Transição concluída na implantação azul/verde.	Nenhum
falha	RDS-EVENT-0249	Transição cancelada na implantação azul/verde.	Nenhum
notificação	RDS-EVENT-0259	A transição do cluster de banco de dados *azul* para *verde* foi iniciada.	Nenhum
notificação	RDS-EVENT-0260	A transição do cluster de banco de dados *azul* para *verde* foi concluída . *Renomeado de azul*	Nenhum

Categoria	ID do evento do Amazon RDS	Mensagem	Observações
		para azul-antigo e verde para azul.	
falha	RDS-EVENT-0261	*A transição do cluster de banco de dados de azul para verde foi cancelada por um motivo.*	Nenhum
notificação	RDS-EVENT-0311	A sincronização de sequência para alternância do cluster de banco de dados *azul* para *verde foi iniciada*. A alternância ao usar sequências pode levar a um tempo de inatividade prolongado.	Nenhum
notificação	RDS-EVENT-0312	A sincronização de sequência para alternância do cluster de banco de dados *azul* para *verde foi concluída* .	Nenhum
falha	RDS-EVENT-0314	A sincronização de sequência para alternância do cluster de banco de dados *azul* para *verde* foi cancelada porque as sequências falharam na sincronização.	Nenhum
notificação	RDS-EVENT-0409	*message*	Nenhum

Monitorar arquivos de log do Amazon Aurora

Cada mecanismo de banco de dados do RDS gera logs que você pode acessar para auditoria e solução de problemas. O tipo dos logs depende do mecanismo do banco de dados.

É possível acessar os logs de banco de dados para instâncias de banco de dados usando o Console de gerenciamento da AWS, a AWS Command Line Interface (AWS CLI) ou a API do Amazon RDS. Você não pode visualizar, nem monitorar, nem baixar logs de transações.

> **Note**
>
> Em alguns casos, os logs contêm dados ocultos. Portanto, o Console de gerenciamento da AWS pode mostrar o conteúdo em um arquivo de log, mas o arquivo de log pode estar vazio quando você o baixa.

Tópicos

- Como visualizar e listar arquivos de log do banco de dados
- Como baixar um arquivo de log de banco de dados
- Como observar um arquivo de log de banco de dados
- Publicação de logs de banco de dados no Amazon CloudWatch Logs
- Leitura do conteúdo de arquivos de log usando REST
- Arquivos de log do banco de dados AuroraMySQL
- Arquivos de log do banco de dados Aurora PostgreSQL

Como visualizar e listar arquivos de log do banco de dados

É possível visualizar arquivos de log de banco de dados do mecanismo de banco de dados do Amazon Aurora usando o Console de gerenciamento da AWS. Você pode listar quais arquivos de log estão disponíveis para download ou monitoramento usando a AWS CLI ou a API do Amazon RDS.

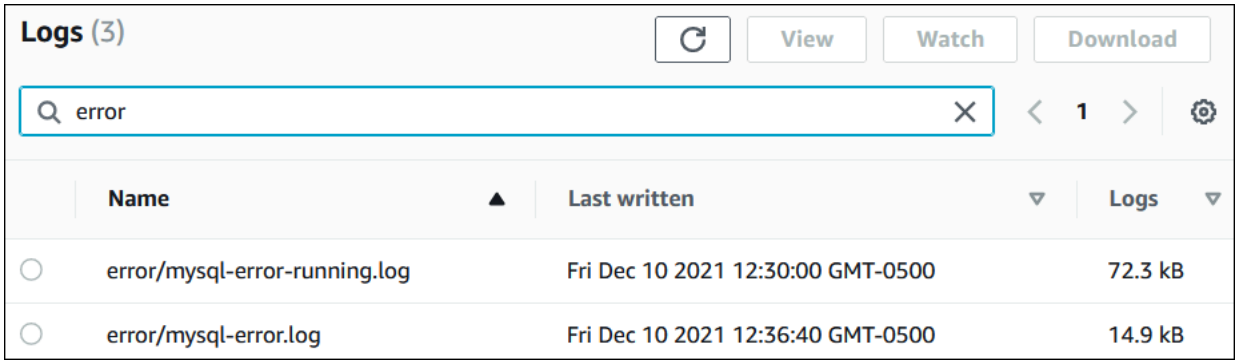
> **Note**
>
> Não é possível visualizar os arquivos de log dos clusters de banco de dados do Aurora Serverless v1 no console do RDS. No entanto, você pode visualizá-los no console do Amazon CloudWatch em https://console.aws.amazon.com/cloudwatch/.

Console

Para visualizar um arquivo de log de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o nome da instância de banco de dados que contém o arquivo de log que você deseja visualizar.
4. Escolha a guia Logs & events (Logs e eventos).
5. Role para baixo até a seção Logs.
6. (Opcional) Insira um termo de pesquisa para filtrar seus resultados.

   O exemplo a seguir lista os logs filtrados pelo texto **`error`**.



7. Escolha o log que você deseja visualizar e, depois, View (Visualizar).

AWS CLI

Para listar os arquivos de log do banco de dados disponíveis para uma instância de banco de dados, use o comando AWS CLI da `describe-db-log-files`.

O exemplo a seguir retorna uma lista de arquivos de log para uma instância de banco de dados chamada `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

API do RDS

Para listar os arquivos de log disponíveis do banco de dados para uma instância de banco de dados, use a ação `DescribeDBLogFiles` da API do Amazon RDS.

Como baixar um arquivo de log de banco de dados

É possível usar o Console de gerenciamento da AWS, a AWS CLI ou a API para baixar um arquivo de log de banco de dados.

Console

Para baixar um arquivo de log de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o nome da instância de banco de dados que contém o arquivo de log que você deseja visualizar.
4. Escolha a guia Logs & events (Logs e eventos).
5. Role para baixo até a seção Logs.
6. Na seção Logs, escolha o botão próximo ao log do qual você deseja baixar e escolha Download.
7. Abra o menu de contexto (clique com o botão direito do mouse) para o link fornecido e escolha Save Link As (Salvar link como). Informe o local onde você deseja salvar o arquivo de log e escolha Save (Salvar).

AWS CLI

Para baixar um arquivo de log de banco de dados, use o comando AWS CLI da `download-db-log-file-portion`. Por padrão, esse comando baixa apenas da parte mais recente de um arquivo de log. No entanto, baixe um arquivo inteiro especificando o parâmetro `--starting-token 0`.

O exemplo a seguir mostra como baixar o conteúdo inteiro de um arquivo de log denominado log/ERROR.4 e armazená-lo em um arquivo local denominado errorlog.txt.

Example

Para Linux, macOS ou Unix:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Para Windows:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

API do RDS

Para baixar um arquivo de log de banco de dados, use a ação `DownloadDBLogFilePortion` da API do Amazon RDS.

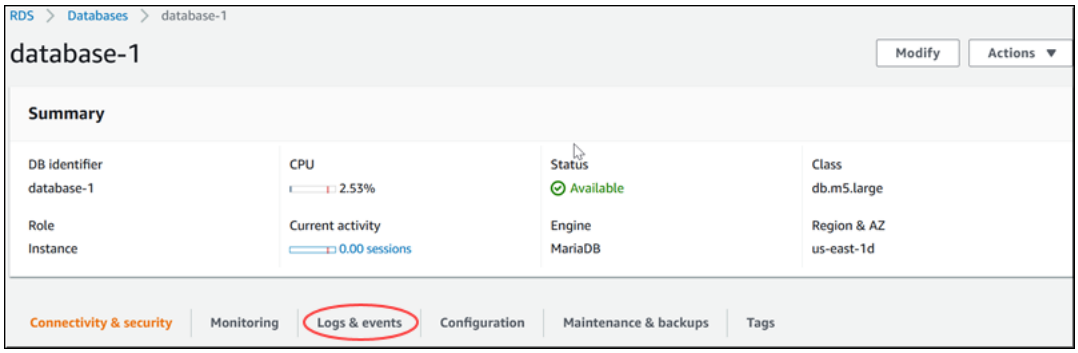
Como observar um arquivo de log de banco de dados

Observar um arquivo de log do banco de dados é equivalente a seguir o arquivo em um sistema UNIX ou Linux. É possível monitorar um arquivo de log usando o Console de gerenciamento da AWS. O RDS atualiza o final do log a cada 5 segundos.

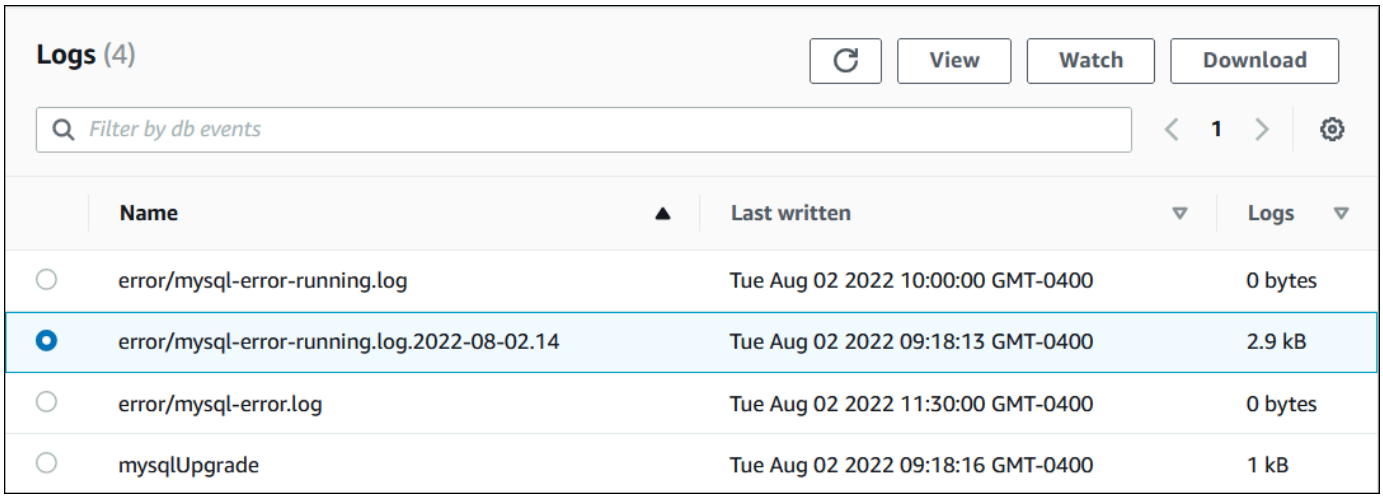
Para observar um arquivo de log de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).

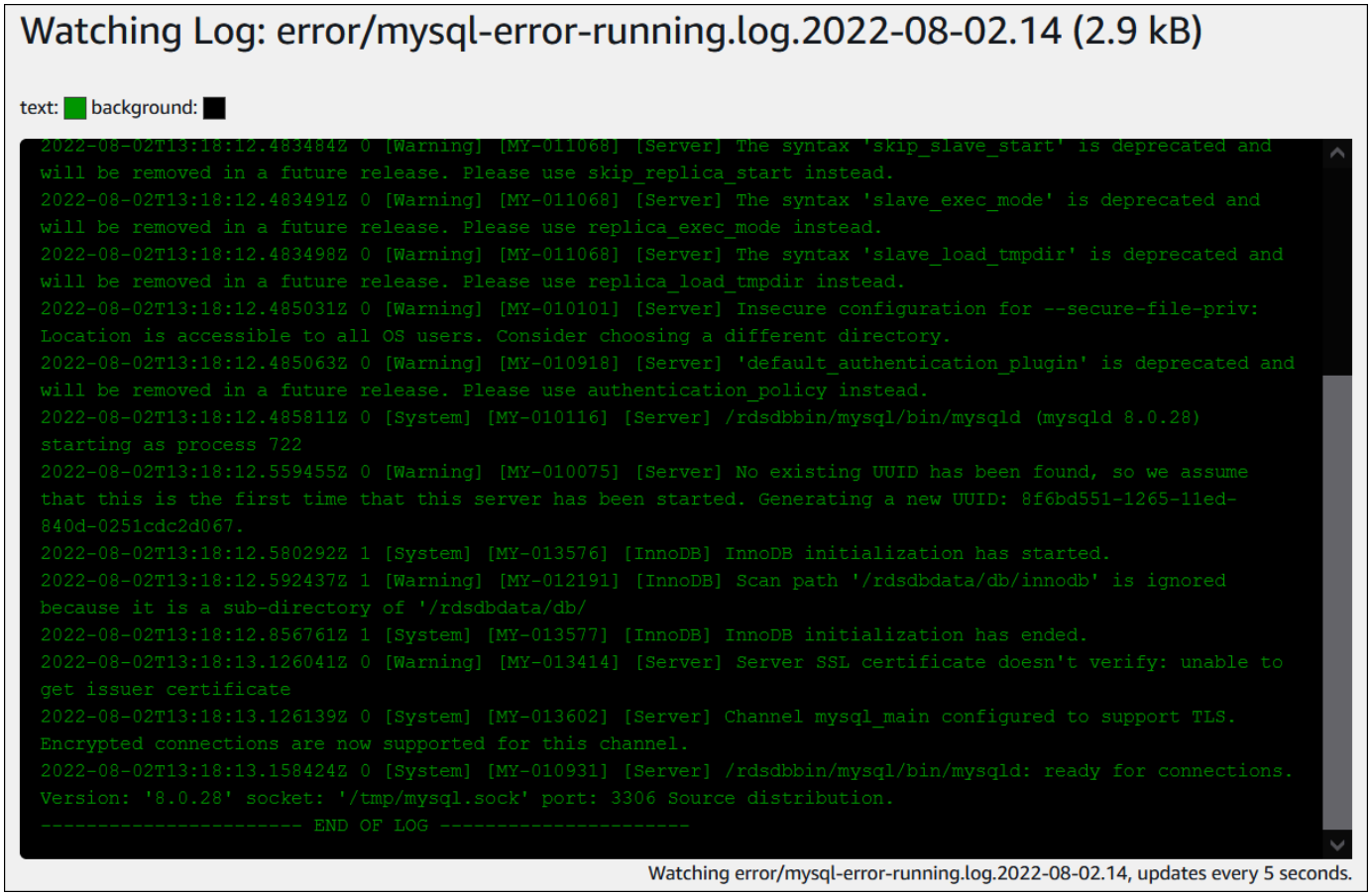
3. Escolha o nome da instância de banco de dados que contém o arquivo de log que você deseja visualizar.
4. Escolha a guia Logs & events (Logs e eventos).



5. Na seção Logs, escolha um arquivo de log e Watch (Observar).



O RDS mostra o final do log, como no exemplo do MySQL a seguir.



Publicação de logs de banco de dados no Amazon CloudWatch Logs

Em um banco de dados local, os registros do banco de dados residem no sistema de arquivos. O Amazon RDS não fornece acesso ao host para os logs de banco de dados no sistema de arquivos de seu cluster de banco de dados. Por esse motivo, o Amazon RDS permite exportar logs de banco de dados para o Amazon CloudWatch Logs. Com o CloudWatch Logs, você pode realizar análise em tempo real de dados de log. Você também pode armazenar os dados em um armazenamento resiliente e gerenciar os dados com o agente do CloudWatch Logs.

Tópicos

- Visão geral da integração do RDS com o CloudWatch Logs
- Decidir quais logs publicar no CloudWatch Logs
- Especificar logs para publicar no CloudWatch Logs
- Pesquisar e filtrar logs no CloudWatch Logs

Visão geral da integração do RDS com o CloudWatch Logs

No CloudWatch Logs, um fluxo de logs é uma sequência de eventos de logs que compartilham a mesma origem. Cada origem separada de logs no CloudWatch Logs compõe um fluxo de logs separado. Um grupo de logs é um grupo de fluxos de log que compartilham as mesmas configurações de retenção, monitoramento e controle de acesso.

O Amazon Aurora transmite continuamente os registros de logs de seu cluster de banco de dados para um grupo de logs. Por exemplo, suponhamos que você tem um grupo de logs `/aws/rds/cluster/`*`cluster_name`*`/`*`log_type`* para cada tipo de log que publica. Esse grupo de logs está na mesma região da AWS que a instância de banco de dados que gera o log.

A AWS retém os dados de log publicados no CloudWatch Logs por um período indefinido, a menos que você especifique um período de retenção. Para obter mais informações, consulte Alterar a retenção de dados de log no CloudWatch Logs.

Decidir quais logs publicar no CloudWatch Logs

Cada mecanismo de banco de dados do RDS oferece suporte ao seu próprio conjunto de logs. Para saber mais sobre as opções do seu mecanismo de banco de dados, consulte os seguintes tópicos:

- the section called “Publicar logs do Aurora MySQL no CloudWatch Logs”
- the section called “Publicar logs do Aurora PostgreSQL no CloudWatch Logs”

Especificar logs para publicar no CloudWatch Logs

Você especifica quais logs deseja publicar no console. Verifique se você tem um perfil vinculado ao serviço no AWS Identity and Access Management (IAM). Para obter mais informações sobre funções vinculadas ao serviço, consulte Usar funções vinculadas ao serviço do Amazon Aurora.

Como especificar os logs que deseja publicar

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Realize um dos procedimentos a seguir:
   - Selecione Criar banco de dados.
   - Escolha um banco de dados da lista e selecione Modify (Modificar).

4. Em Logs exports (Exportações de logs), escolha os logs para publicar.

   O exemplo a seguir especifica o log de auditoria, os logs de erros, o log geral, o log da instância, o log de erros de autenticação de banco de dados do IAM e o log de consultas lentas de um cluster de banco de dados do Aurora MySQL.

Pesquisar e filtrar logs no CloudWatch Logs

Você pode procurar entradas de log que atendam a critérios especificados usando o console do CloudWatch Logs. Você pode acessar os logs por meio do console do RDS, que leva você ao console do CloudWatch Logs, ou diretamente do console do CloudWatch Logs.

Como pesquisar logs do RDS usando o console do RDS

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha um cluster de banco de dados ou uma instância de banco de dados.
4. Escolher configuração.
5. Em Published logs (Logs publicados), escolha o log de banco de dados que deseja exibir.

Como pesquisar logs do RDS usando o console do CloudWatch Logs

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Grupos de logs.
3. Na caixa de filtro, insira **`/aws/rds`**.
4. Em Grupos de logs, escolha o nome do grupo de logs que contém o fluxo de log a ser pesquisado.
5. Em Fluxos de log, escolha o nome do fluxo de log para pesquisa.
6. Em Eventos de log, insira a sintaxe do filtro a ser usada.

Para obter mais informações, consulte Pesquisar e filtrar dados de logs no Guia do usuário do Amazon CloudWatch Logs. Para acessar um tutorial de blog explicando como monitorar logs do RDS, consulte Criar monitoramento de banco de dados proativo para o Amazon RDS com o Amazon CloudWatch Logs, o AWS Lambda e o Amazon SNS.

Leitura do conteúdo de arquivos de log usando REST

O Amazon RDS fornece um endpoint REST que permite acesso aos arquivos de log de instâncias de banco de dados. Isso é útil para gravar uma aplicação para transmitir o conteúdo do arquivo de log do Amazon RDS.

A sintaxe é:

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

Os seguintes parâmetros são obrigatórios:

- *DBInstanceIdentifier*: o nome da instância de banco de dados que contém o arquivo de log do qual você deseja baixar.
- *LogFileName*: o nome do arquivo de log que será baixado.

A resposta contém o conteúdo do arquivo de log solicitado, como um fluxo.

O exemplo a seguir baixa o arquivo de log chamado log/ERROR.6 para a instância de banco de dados chamada sample-sql na região us-west-2.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

Se você especificar uma instância de banco de dados não existente, a resposta consistirá no erro a seguir:

- `DBInstanceNotFound`: *DBInstanceIdentifier* não se refere a uma instância de banco de dados existente. (Código de status HTTP: 404)

Arquivos de log do banco de dados AuroraMySQL

Você pode monitorar os logs do Aurora MySQL diretamente por meio do console do Amazon RDS, da API do Amazon RDS, da AWS CLI ou dos SDKs da AWS. Você também pode acessar os logs do MySQL direcionando os logs para uma tabela de banco de dados no banco de dados primário e consultando essa tabela. Você pode usar o utilitário mysqlbinlog para baixar um log de binários.

Para mais informações sobre a visualização, o download e os logs de bancos de dados baseados no monitoramento de arquivos, consulte Monitorar arquivos de log do Amazon Aurora.

Tópicos

- Visão geral dos logs de banco de dados do Aurora MySQL
- Enviar a saída de log do AuroraMySQL para tabelas
- Configurar o registro em log binário do Aurora MySQL para bancos de dados single-AZ
- Acessar logs binários do MySQL

Visão geral dos logs de banco de dados do Aurora MySQL

Você pode monitorar os seguintes tipos de arquivos de log do Aurora MySQL:

- Log de erros
- Log de consultas lentas
- Log geral
- Log de auditoria
- Log de instância
- Log de erros de autenticação de banco de dados do IAM

O log de erros do Aurora MySQL é gerado por padrão. Você pode gerar a consulta lenta e os logs gerais definindo parâmetros no seu grupo de parâmetros do banco de dados.

Tópicos

- Logs de erro do Aurora MySQL
- Logs gerais e de consultas lentas do Aurora MySQL
- Log de auditoria do Aurora MySQL

- Log de instâncias do Aurora MySQL
- Alternância e retenção de logs do Aurora MySQL
- Publicar logs do Aurora MySQL no Amazon CloudWatch Logs

Logs de erro do Aurora MySQL

O Aurora MySQL grava erros no arquivo `mysql-error.log`. Cada arquivo de log tem a hora em que foi gerado (em UTC) anexada ao seu nome. Os arquivos de log também possuem um carimbo de data/hora que ajuda você a determinar quando as entradas de log foram gravadas.

O Aurora MySQL grava no log de erros apenas na inicialização, no desligamento e quando encontra erros. Uma instância de banco de dados pode passar horas ou dias sem novas entradas gravadas no log de erros. Se você não vir nenhuma entrada recente, é porque o servidor não encontrou nenhum erro que tenha gerado uma entrada de log.

Por padrão, os logs de erros são filtrados para que apenas eventos inesperados, como erros, sejam exibidos. No entanto, os logs de erros também contêm algumas informações adicionais do banco de dados, por exemplo, o andamento da consulta, que não são mostradas. Portanto, mesmo sem erros reais, o tamanho dos logs de erros pode aumentar devido às atividades em andamento do banco de dados. Embora você possa ver determinado tamanho em bytes ou quilobytes para os logs de erros no Console de gerenciamento da AWS, eles poderão ter 0 byte quando você fizer download deles.

O Aurora MySQL grava o `mysql-error.log` no disco a cada cinco minutos. Ele acrescenta o conteúdo do log ao `mysql-error-running.log`.

O Aurora MySQL alterna o arquivo `mysql-error-running.log` de hora em hora.

> **Note**
>
> Observe que o período de retenção é diferente entre o Amazon RDS e o Aurora.

Logs gerais e de consultas lentas do Aurora MySQL

É possível gravar o log de consultas lentas e o log geral do Aurora MySQL em um arquivo ou em uma tabela de banco de dados. Para isso, defina parâmetros em seu grupo de parâmetros de banco de dados. Para obter informações sobre como criar e modificar um grupo de parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora. Você deve definir esses parâmetros

antes de visualizar o log de consultas lentas ou o log geral no console do Amazon RDS ou usando a API do Amazon RDS, a CLI do Amazon RDS ou os SDKs da AWS.

Você pode controlar o registro em log do Aurora MySQL usando os parâmetros nesta lista:

- `slow_query_log`: para criar o log de consultas lentas, defina como 1. O padrão é 0.
- `general_log`: para criar o log geral, defina como 1. O padrão é 0.
- `long_query_time`: para evitar que as consultas de execução rápida sejam registradas no log de consultas lentas, especifique um valor para o tempo de execução de consultas mais curto a ser registrado, em segundos. O padrão é 10 segundos; o mínimo é 0. Se log_output = FILE, você poderá especificar um valor de ponto flutuante com resolução por microssegundo. Se log_output = TABLE, você deverá especificar um valor inteiro com a segunda resolução. Apenas as consultas cujo tempo de execução excede o valor `long_query_time` são registradas em log. Por exemplo, definir `long_query_time` como 0.1 impede que qualquer consulta que seja executada por menos de 100 milissegundos seja registrada.
- `log_queries_not_using_indexes`: para registrar todas as consultas que não usam um índice no log de consultas lentas, defina como 1. As consultas que não usam um índice são registradas, mesmo que seu tempo de execução seja inferior ao valor do parâmetro `long_query_time`. O padrão é 0.
- `log_output` *`option`*: você pode especificar uma das seguintes opções para o parâmetro `log_output`.
  - TABLE : grava consultas gerais na tabela `mysql.general_log` e consultas lentas na tabela `mysql.slow_log`.
  - FILE: grave logs de consultas gerais e lentas no sistema de arquivos.
  - NONE: desabilite o registro em log.

  Em relação ao Aurora MySQL versões 2 e 3, o padrão de `log_output` é `FILE`.

Para que dados de consulta lenta apareçam no Amazon CloudWatch Logs, as seguintes condições devem ser atendidas:

- O CloudWatch Logs deve ser configurado para incluir logs de consultas lentas.
- `slow_query_log` deve estar habilitado.
- `log_output` deve ser definido como `FILE`.
- A consulta deve demorar mais do que o tempo configurado para `long_query_time`.

Para obter mais informações sobre os log de consultas gerais e de consultas lentas, acesse os seguintes tópicos na documentação do MySQL:

- O log de consultas lentas
- O log de consultas gerais

Log de auditoria do Aurora MySQL

O registro em log de auditoria do Aurora MySQL é chamado de Auditoria avançada. Para ativar a Auditoria avançada, defina determinados parâmetros de cluster de banco de dados. Para obter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.

Log de instâncias do Aurora MySQL

O Aurora cria um arquivo de log separado para instâncias de banco de dados que têm a pausa automática habilitada. Esse arquivo instance.log registra todos os motivos pelos quais essas não foi possível pausar as instâncias de banco de dados quando esperado. Para ter mais informações sobre o comportamento do arquivo de log da instância e o recurso de pausa automática do Aurora, consulte Monitorar a atividade de pausa e retomada do Aurora Serverless v2.

Alternância e retenção de logs do Aurora MySQL

Quando o registro em log está ativado, o Amazon Aurora alterna ou exclui os arquivos de log em intervalos regulares. Essa medida é uma precaução para reduzir a possibilidade de um arquivo de log grande bloquear o uso do banco de dados ou afetar o desempenho. O Aurora MySQL lida com a alternância e a exclusão da seguinte forma:

- Os tamanhos dos arquivos de log de erros do Aurora MySQL são limitados a 15% do armazenamento local para uma instância de banco de dados. Para manter esse limite, os logs são alternados automaticamente a cada hora. O Aurora MySQL remove logs após 30 dias ou quando 15% do espaço em disco são atingidos. Se o tamanho do arquivo de log combinado exceder o limite após a remoção dos arquivos de log antigos, os arquivos de log mais antigos serão excluídos até o tamanho do arquivo de log deixar de exceder esse limite.
- O Aurora MySQL removerá os logs de auditoria, gerais e de consulta lenta após 24 horas ou quando 15% do armazenamento tiver sido usado.
- Quando o registro em log `FILE` está ativado, os arquivos de log geral e de consulta lenta são examinados a cada hora, e os arquivos de log com mais de 24 horas são excluídos. Em alguns

casos, o tamanho do arquivo de log combinado restante após a exclusão pode exceder o limite de 15% do espaço local da instância de um banco de dados. Nesses casos, os arquivos de log mais antigos serão excluídos até que o tamanho do arquivo de log não exceda o limite.

- Quando o registro em log de `TABLE` está ativado, as tabelas de log não são alternadas nem excluídas. As tabelas de log são truncadas quando o tamanho de todos os logs combinados é grande demais. Você pode assinar a categoria de evento `low storage` para ser notificado quando for necessário alternar ou excluir manualmente tabelas de logs para liberar espaço. Para obter mais informações, consulte Trabalhar com a notificação de eventos do Amazon RDS.

  Você pode alternar a tabela `mysql.general_log` chamando o procedimento `mysql.rds_rotate_general_log`. Você pode rotacionar a tabela `mysql.slow_log` chamando o procedimento `mysql.rds_rotate_slow_log`.

  Quando você alterna as tabelas de logs manualmente, a tabela de logs atual é copiada em uma tabela de logs de backup, e as entradas na tabela de logs atual são removidas. Se a tabela de log de backup já existir, então ela será excluída antes que a tabela de log atual seja copiada ao backup. Você pode consultar a tabela de log de backup, se necessário. A tabela de log de backup para a tabela `mysql.general_log` é denominada `mysql.general_log_backup`. A tabela de log de backup para a tabela `mysql.slow_log` é denominada `mysql.slow_log_backup`.
- Os logs de auditoria do Aurora MySQL são alternados quando o tamanho do arquivo atinge 100 MB e são removidos após 24 horas.
- O Amazon RDS alterna os arquivos de log de erros de autenticação de banco de dados do IAM maiores que 10 MB. O Amazon RDS remove os arquivos de log de erros de autenticação de banco de dados do IAM com mais de cinco dias ou maiores que 100 MB.

Para trabalhar com os logs no console do Amazon RDS, na API do Amazon RDS, na CLI do Amazon RDS ou nos SDKs da AWS, defina o parâmetro `log_output` como FILE. Como o log de erros do Aurora MySQL, esses arquivos de log são alternados por hora. Os arquivos de log que foram gerados durante as 24 horas anteriores são retidos. Observe que o período de retenção é diferente entre o Amazon RDS e o Aurora.

Publicar logs do Aurora MySQL no Amazon CloudWatch Logs

É possível configurar seu cluster de bancos de dados Aurora MySQL para publicar dados de log em um grupo no Amazon CloudWatch Logs. Com o CloudWatch Logs, você pode executar análise em tempo real de dados de log e usar o CloudWatch para criar alarmes e visualizar métricas. Você pode usar o CloudWatch Logs para armazenar seus registros de log em armazenamento resiliente. Para

obter mais informações, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.

Enviar a saída de log do AuroraMySQL para tabelas

Você pode direcionar os logs de consultas gerais e lentas para tabelas na instância de banco de dados, criando um grupo de parâmetros de banco de dados e definindo o parâmetro do servidor `log_output` como `TABLE`. As consultas gerais são registradas na tabela `mysql.general_log` e as consultas lentas são registradas na tabela `mysql.slow_log`. Você pode consultar as tabelas para acessar as informações do log. Habilitar esse registro aumenta a quantidade de dados gravados no banco de dados, o que pode degradar a performance.

O log geral e os logs de consultas lentas estão desabilitados por padrão. Para habilitar o registro em log de tabelas, você também deve definir os parâmetros de servidor `general_log` e `slow_query_log` como `1`.

As tabelas de log continuarão crescendo até que as respectivas atividades de log sejam desativadas com a redefinição do parâmetro apropriado como `0`. Uma grande quantidade de dados geralmente se acumula ao longo do tempo, o que pode consumir uma porcentagem considerável do espaço de armazenamento alocado. O Amazon Aurora não permite truncar tabelas de log, mas é possível mover o conteúdo delas. Rotacionar uma tabela salva seu conteúdo em uma tabela de backup e, em seguida, cria uma nova tabela de log vazia. Você pode rotacionar manualmente as tabelas de log com os seguintes procedimentos de linha de comando, em que o prompt de comando é indicado por `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

Para remover completamente os dados antigos e recuperar o espaço em disco, chame o procedimento apropriado duas vezes sucessivamente.

Configurar o registro em log binário do Aurora MySQL para bancos de dados single-AZ

O log binário é um conjunto de arquivos de log que contêm informações sobre modificações de dados feitas em uma instância do servidor Aurora MySQL. O log binário contém informações como as seguintes:

- Eventos que descrevem alterações no banco de dados, como criação de tabela ou modificações de linha
- Informações sobre a duração de cada instrução que atualizou dados
- Eventos para declarações que poderiam ter dados atualizados, mas não foram

O log binário registra instruções que são enviadas durante a replicação. Também é necessário para algumas operações de recuperação. Para obter mais informações, consulte The Binary Log na documentação do MySQL.

Os logs binários só podem ser acessados pela instância de banco de dados primária, não pelas réplicas.

O MySQL no Amazon Aurora é compatível com os formatos de registro em log binário baseados em linha, baseados em instrução e mistos. Recomendamos misto, a menos que você precise de um formato específico de log binário. Para obter detalhes sobre os diferentes formatos de logs binários do Aurora MySQL, consulte Formatos de registro em log binário na documentação do MySQL.

Se você pretende usar replicação, o formato do registro em log binário é importante porque determina o registro de alterações feitas nos dados salvas na origem e enviadas para os destinos de replicação. Para obter informações sobre as vantagens e as desvantagens de formatos de registro em logs binários para replicação, consulte Vantagens e desvantagens da replicação baseada em instrução e baseada em linha na documentação do MySQL.

> **Important**
>
> Com o MySQL 8.0.34, o MySQL descontinuou o parâmetro `binlog_format`. Em versões posteriores do MySQL, o MySQL pretende remover o parâmetro e oferecer suporte somente à replicação baseada em linhas. Como resultado, recomendamos o uso de registro baseado em linhas para novas configurações de replicação do MySQL. Para obter mais informações, consulte binlog_format na documentação do MySQL.
> As versões 8.0 e 8.4 do MySQL aceitam o parâmetro `binlog_format`. Ao usar esse parâmetro, o MySQL emite um aviso de descontinuação. Em uma futura versão principal, o MySQL removerá o parâmetro `binlog_format`.
> A replicação baseada em instrução pode causar inconsistências entre o cluster de banco de dados de origem e uma réplica de leitura. Para obter mais informações, consulte Determinação de instruções seguras e inseguras no registro de logs binários. na documentação do MySQL.

Habilitar o registro em log binário aumenta o número de operações de E/S do disco de gravação no cluster de banco de dados. Você pode monitorar o uso de IOPS com a métrica `VolumeWriteIOPs` do CloudWatch.

Para definir o formato de registro em log binário do MySQL

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Selecione o grupo de parâmetros do cluster de banco de dados associado ao cluster de banco de dados que você deseja modificar.

   Não é possível modificar um grupo de parâmetros padrão. Se o cluster de banco de dados estiver usando um grupo de parâmetros padrão, crie outro grupo de parâmetros e o associe ao cluster de banco de dados.

   Para ter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.
4. Em Ações, selecione Editar.
5. Defina o parâmetro `binlog_format` como o formato de registro em log binário de sua escolha (`ROW`, `STATEMENT` ou `MIXED`). Você também pode usar o valor `OFF` para desativar o registro em log binário.

   > **Note**
   >
   > Definir `binlog_format` como `OFF` no grupo de parâmetros do cluster de banco de dados desativa a variável de sessão `log_bin`. Isso desabilita o registro em log binário no cluster de banco de dados do Aurora MySQL que, por sua vez, redefine a variável de sessão `binlog_format` como o valor padrão de `ROW` no banco de dados.
6. Escolha Salvar alterações para salvar as atualizações no grupo de parâmetros do cluster de banco de dados.

Depois de realizar essas etapas, você deve reinicializar a instância do gravador no cluster de banco de dados para que suas alterações sejam aplicadas. No Aurora MySQL versão 2.09 e anteriores, quando você reinicializa a instância do gravador, todas as instâncias do leitor no cluster de banco de dados também são reinicializadas. No Aurora MySQL versão 2.10 e posteriores, você deve

reinicializar todas as instâncias do leitor manualmente. Para obter mais informações, consulte Reinicializar um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora.

> **Important**
>
> Alterar um grupo de parâmetros de cluster de banco de dados afeta todos os clusters de banco de dados que usam esse grupo de parâmetros. Se você quiser especificar diferentes formatos do registro em log binário para diferentes clusters de bancos de dados Aurora MySQL em uma região da AWS, os clusters de banco de dados deverão usar diferentes grupos de parâmetros de cluster de banco de dados. Esses grupos de parâmetros identificam diferentes formatos de log. Atribua o grupo de parâmetros de cluster de banco de dados apropriado a cada cluster de banco de dados. Para ter mais informações sobre parâmetros do Aurora MySQL, consulte Parâmetros de configuração do Aurora MySQL.

Acessar logs binários do MySQL

É possível usar o utilitário mysqlbinlog para baixar ou transmitir os logs binários de instâncias do RDS para instâncias de banco de dados do MySQL. O log binário é baixado para o computador local, onde você pode realizar ações como reproduzir o log usando o utilitário mysql. Para ter mais informações sobre como usar o utilitário mysqlbinlog, acesse Usar mysqlbinlog para fazer backup de arquivos de log binários na documentação do MySQL.

Para executar o utilitário mysqlbinlog em uma instância do Amazon RDS, use as seguintes opções:

- `--read-from-remote-server` – obrigatório.
- `--host`: o nome DNS do endpoint da instância.
- `--port`: a porta usada pela instância.
- `--user`: um usuário do MySQL ao qual foi concedida a permissão `REPLICATION SLAVE`.
- `--password`: a senha do usuário do MySQL ou omita um valor de senha de forma que o utilitário solicite uma senha.
- `--raw`: baixe o arquivo em formato binário.
- `--result-file`: o arquivo local para receber a saída bruta.
- `--stop-never`: transmita os arquivos de log binários.

- `--verbose`: ao usar o formato de log binário `ROW`, inclua essa opção para ver os eventos de linha como instruções pseudo-SQL. Para ter mais informações sobre a opção `--verbose`, consulte Exibição de evento da linha mysqlbinlog na documentação do MySQL.
- Especifique os nomes de um ou mais arquivos de log binários. Para obter uma lista dos logs disponíveis, use o comando SQL `SHOW BINARY LOGS`.

Para ter mais informações sobre as opções de mysqlbinlog, consulte mysqlbinlog: utilitário para processar arquivos de log binários na documentação do MySQL.

Os exemplos a seguir mostram como usar o utilitário mysqlbinlog.

Para Linux, macOS ou Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Para Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

Os logs binários devem permanecer disponíveis na instância de banco de dados para que o utilitário mysqlbinlog possa acessá-los. Para garantir a disponibilidade deles, use o procedimento armazenado mysql.rds_set_configuration e especifique um período com tempo suficiente para você

baixar os logs. Se essa configuração não for definida, o Amazon RDS purgará os logs binários o mais rápido possível, causando lacunas nos logs binários que o utilitário mysqlbinlog recupera.

O exemplo a seguir define o período de retenção como 1 dia.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Para exibir a configuração atual, use o procedimento armazenado mysql.rds_show_configuration.

```
call mysql.rds_show_configuration;
```

Arquivos de log do banco de dados Aurora PostgreSQL

Você pode monitorar os seguintes tipos de arquivo de log do Aurora PostgreSQL:

- Log do PostgreSQL
- Log de instância
- Log de erros de autenticação de banco de dados do IAM

> **Note**
>
> Para habilitar os logs de erro de autenticação de banco de dados do IAM, você deve primeiro habilitar a autenticação de banco de dados do IAM para o cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações sobre a autenticação de banco de dados do IAM, consulte Habilitar e desabilitar a autenticação de banco de dados do IAM.

O Aurora PostgreSQL registra as atividades do banco de dados no arquivo de log padrão do PostgreSQL. Para uma instância de banco de dados PostgreSQL on-premises, essas mensagens são armazenadas localmente em `log/postgresql.log`. Para um cluster de banco de dados do Aurora PostgreSQL, o arquivo de log está disponível no cluster do Aurora. Esses logs também podem ser acessados por meio do Console de gerenciamento da AWS, onde você pode visualizá-los ou baixá-los. O nível de registro em log padrão captura falhas de login, erros fatais do servidor, deadlocks e falhas de consulta.

Para ter mais informações sobre a visualização, o download e os logs de banco de dados baseados no monitoramento de arquivos, consulte Monitorar arquivos de log do Amazon Aurora. Para saber mais sobre logs do PostgreSQL, consulte Working with Amazon RDS and Aurora PostgreSQL logs: Part 1 (Trabalhar com o Amazon RDS e logs do Aurora PostgreSQL: parte 1) e Working with RDS and Aurora PostgreSQL logs: Part 2 (Trabalhar com o Amazon RDS e logs do Aurora PostgreSQL: parte 2).

Além dos logs padrão do PostgreSQL abordados neste tópico, o Aurora PostgreSQL também é compatível com a extensão Audit do PostgreSQL (`pgAudit`). A maioria dos setores regulamentados e agências governamentais precisa manter um log de auditoria ou uma trilha de auditoria das alterações feitas nos dados para cumprir os requisitos legais. Para obter informações sobre a instalação e o uso de pgAudit, consulte Usar pgAudit para registrar a atividade do banco de dados.

O Aurora cria um arquivo de log separado para instâncias de banco de dados que têm a pausa automática habilitada. Esse arquivo instance.log registra todos os motivos pelos quais essas não foi possível pausar as instâncias de banco de dados quando esperado. Para ter mais informações sobre o comportamento do arquivo de log da instância e o recurso de pausa automática do Aurora, consulte Monitorar a atividade de pausa e retomada do Aurora Serverless v2.

Tópicos

- Parâmetros para registro em log no Aurora PostgreSQL
- Ativar o registro em log de consultas para o cluster de banco de dados do Aurora PostgreSQL

Parâmetros para registro em log no Aurora PostgreSQL

É possível personalizar o comportamento do registro em log do cluster de banco de dados do Aurora PostgreSQL modificando vários parâmetros. Na tabela a seguir, você encontra os parâmetros que afetam por quanto tempo os logs são armazenados, quando alternar o log e se ele deve ser gerado no formato CSV (valor separado por vírgula). Você também pode encontrar a saída de texto enviada para STDERR, entre outras configurações. Para alterar as configurações dos parâmetros que podem ser modificados, use um grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Grupos de parâmetros para Amazon Aurora.

Parâmetro	Padrão	Descrição
log_destination	stderr	Define o formato de saída para o log. O padrão é `stderr`, mas você também pode especificar valores separados por vírgula (CSV) adicionando `csvlog` à configuração. Para obter mais informações, consulte Definir o destino dos logs (`stderr`, `csvlog`).
log_filename	postgresql.log.%Y-%m-%d-%H%M	Especifica o padrão para o nome do arquivo de log. Além do padrão, esse parâmetro é compatível com `postgresql.log.%Y-%m-%d` e `postgresql.log.%Y-%m-%d-%H` para o padrão de nome de arquivo. Para o

Parâmetro	Padrão	Descrição
		Aurora PostgreSQL versão 17.4 e posterior, não é possível modificar esse parâmetro.
log_line_prefix	%t:%r:%u@%d:[%p]:	Define o prefixo para cada linha de log que é gravada em `stderr`, para anotar a hora (%t), o host remoto (%r), o usuário (%u), o banco de dados (%d) e o ID do processo (%p).
log_rotation_age	60	Minutos após os quais o arquivo de log é alternado automaticamente. É possível alterar esse valor no intervalo de 1 a e 1.440 minutos. Para obter mais informações, consulte Configurar a alternância do arquivo de log.
log_rotation_size	–	O tamanho (kB) no qual o log é alternado automaticamente. É possível alterar esse valor no intervalo de 50.000 a 1.000.000 de quilobytes. Para saber mais, consulte Configurar a alternância do arquivo de log.
rds.log_retention_period	4320	Os logs do PostgreSQL mais antigos que o número especificado de minutos são excluídos. O valor padrão de 4.320 minutos exclui os arquivos de log após três dias. Para ter mais informações, consulte Definir o período de retenção de log.

Para identificar problemas de aplicações, você pode procurar falhas de consulta, falhas de login, deadlocks e erros fatais de servidor no log. Por exemplo, suponha que você converta uma aplicação herdada do Oracle no Aurora PostgreSQL, mas nem todas as consultas foram convertidas corretamente. Essas consultas formatadas incorretamente geram mensagens de erro nos logs, que você pode usar para identificar problemas. Para ter mais informações sobre o registro em log de consultas, consulte Ativar o registro em log de consultas para o cluster de banco de dados do Aurora PostgreSQL.

Nos tópicos a seguir, você encontrará informações sobre como definir vários parâmetros que controlam os detalhes básicos de seus logs do PostgreSQL.

Tópicos

- Definir o período de retenção de log
- Configurar a alternância do arquivo de log
- Definir o destino dos logs (stderr, csvlog)
- Noções básicas sobre o parâmetro log_line_prefix

Definir o período de retenção de log

O parâmetro `rds.log_retention_period` especifica por quanto tempo seu cluster de banco de dados do Aurora PostgreSQL mantém seus arquivos de log. A configuração padrão é de três dias (4.320 minutos), mas você pode definir esse valor de um dia (1.440 minutos) a sete dias (10.080 minutos). O cluster de banco de dados do Aurora PostgreSQL deve ter armazenamento suficiente para armazenar os arquivos de log durante o período.

Recomendamos que você publique habitualmente seus logs no Amazon CloudWatch Logs para que possa visualizar e analisar os dados do sistema muito tempo depois que os logs tiverem sido removidos do cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs. Depois que você configura a publicação do CloudWatch, o Aurora não exclui um log enquanto ele não for publicado no CloudWatch Logs.

O Amazon Aurora compacta logs do PostgreSQL mais antigos quando o armazenamento para a instância de banco de dados atinge um limite. O Aurora compacta os arquivos usando o utilitário de compactação gzip. Para ter mais informações, consulte o site do gzip.

Quando a instância de banco de dados estiver com pouco espaço de armazenamento e todos os logs disponíveis estiverem compactados, você receberá um aviso como o seguinte:

```
Warning: local storage for PostgreSQL log files is critically low for
this Aurora PostgreSQL instance, and could lead to a database outage.
```

Se não houver armazenamento suficiente, o Aurora poderá excluir os logs compactados do PostgreSQL antes do final do período de retenção especificado. Se isso ocorrer, será exibida uma mensagem semelhante à seguinte:

```
The oldest PostgreSQL log files were deleted due to local storage constraints.
```

Configurar a alternância do arquivo de log

Por padrão, o Aurora cria arquivos de log a cada hora. O tempo é controlado pelo parâmetro `log_rotation_age`. Esse parâmetro tem um valor padrão de 60 (minutos), mas você pode definir qualquer valor entre 1 minuto e 24 horas (1.440 minutos). Quando chegar o momento da alternância, será criado um novo arquivo de log distinto. O arquivo é nomeado de acordo com o padrão especificado pelo parâmetro `log_filename`.

Também é possível alternar os arquivos de log de acordo com o tamanho, conforme especificado no parâmetro `log_rotation_size`. Esse parâmetro especifica que o log deva ser alternado quando atingir o tamanho determinado (em kilobytes). O valor padrão `log_rotation_size` é 100 mil kB (kilobytes) para um cluster de banco de dados do Aurora PostgreSQL, mas é possível definir esse valor entre 50 mil e 1 milhão de kilobytes.

Os nomes dos arquivos de log são baseados no padrão de nome do arquivo do parâmetro `log_filename`. As configurações disponíveis para esse parâmetro são as seguintes:

- `postgresql.log.%Y-%m-%d`: formato padrão do nome do arquivo de log. Inclui o ano, o mês e a data no nome do arquivo de log.
- `postgresql.log.%Y-%m-%d-%H`: inclui a hora no formato do nome do arquivo de log.
- `postgresql.log.%Y-%m-%d-%H%M`: inclui hora:minuto no formato do nome do arquivo de log.

Se você definir o parâmetro `log_rotation_age` como menos de 60 minutos, defina o parâmetro `log_filename` como o formato em minutos.

Para ter mais informações, consulte `log_rotation_age` e `log_rotation_size` na documentação do PostgreSQL.

Definir o destino dos logs (**`stderr`**, **`csvlog`**)

Por padrão, o Aurora PostgreSQL gera logs no formato de erro padrão (stderr). Esse formato é a configuração padrão do parâmetro `log_destination`. Cada mensagem é prefixada usando o padrão especificado no parâmetro `log_line_prefix`. Para ter mais informações, consulte Noções básicas sobre o parâmetro log_line_prefix.

O Aurora PostgreSQL também pode gerar os registros no formato `csvlog`. O `csvlog` é útil para analisar os dados de log como dados de valores separados por vírgula (CSV). Por exemplo, digamos

que você use a extensão `log_fdw` para trabalhar com seus logs como tabelas externas. A tabela externa criada nos arquivos de log do `stderr` contém uma única coluna com dados de eventos de log. Ao adicionar `csvlog` ao parâmetro `log_destination`, você obtém o arquivo de log no formato CSV com demarcações para as várias colunas da tabela externa. Agora você pode classificar e analisar os logs com maior facilidade.

Se você especificar `csvlog` para esse parâmetro, lembre-se de que os arquivos `stderr` e `csvlog` são gerados. Monitore o armazenamento consumido pelos logs, levando em consideração o `rds.log_retention_period` e outras configurações que afetam o armazenamento e a rotatividade dos logs. O uso de `stderr` e `csvlog` mais do que dobra o armazenamento consumido pelos logs.

Se você adicionar `csvlog` a `log_destination` e quiser reverter para o `stderr`, precisará redefinir o parâmetro. Para fazer isso, use o console do Amazon RDS e, depois, abra o grupo de parâmetros do cluster de banco de dados para sua instância. Selecione o parâmetro `log_destination`, Edit parameter (Editar parâmetro) e depois Reset (Redefinir).

Para ter mais informações sobre como configurar o registro em log, consulte Trabalhar com logs do Amazon RDS e do Aurora PostgreSQL: parte 1.

Noções básicas sobre o parâmetro log_line_prefix

O formato de log `stderr` prefixa cada mensagem de log com os detalhes especificados pelo parâmetro `log_line_prefix`. O valor padrão é:

```
%t:%r:%u@%d:[%p]:t
```

A partir do Aurora PostgreSQL versão 16, você também pode escolher:

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

Cada entrada de log enviada para stderr inclui as seguintes informações com base no valor selecionado:

- `%t`: hora da entrada do log sem milissegundos.
- `%m`: hora da entrada do log com milissegundos.
- `%r`: endereço do host remoto.
- `%u@%d`: nome de usuário no nome do banco de dados.

- `[%p]`: ID do processo, se disponível.
- `%l`: número da linha de log por sessão.
- `%e`: código de erro SQL.
- `%s`: data e hora de início do processo.
- `%v`: ID da transação virtual.
- `%x`: ID da transação.
- `%c`: ID da sessão.
- `%q`: terminador não relacionado à sessão.
- `%a`: nome da aplicação.

Ativar o registro em log de consultas para o cluster de banco de dados do Aurora PostgreSQL

Você pode coletar informações mais detalhadas sobre suas atividades de banco de dados, inclusive consultas, consultas à espera de bloqueios, pontos de verificação e muitos outros detalhes definindo alguns dos parâmetros listados na tabela a seguir. Este tópico se concentra no registro em log de consultas.

Parâmetro	Padrão	Descrição
log_connections	–	Registra cada conexão bem-sucedida. Para saber como usar esse parâmetro com `log_disconnections` para detectar a interrupção da conexão, consulte Gerenciar a rotatividade de conexão do Aurora PostgreSQL com agrupamento.
log_disconnections	–	Registra o final de cada sessão e sua duração. Para saber como usar esse parâmetro com `log_connections` para detectar a interrupção da conexão, consulte Gerenciar a rotatividade de conexão do Aurora PostgreSQL com agrupamento.
log_checkpoints	–	Não aplicável ao Aurora PostgreSQL

Parâmetro	Padrão	Descrição
log_lock_waits	–	Registra esperas de bloqueio longas. Por padrão, esse parâmetro não está definido.
log_min_duration_sample	–	(ms) Define o tempo de execução mínimo acima do qual uma amostra de declaraçõ es será registrada. O tamanho da amostra é definido usando o parâmetro `log_statement_sample_rate` .
log_min_duration_statement	–	Todas as instruções SQL executadas pelo menos por um período especificado ou mais é registrada. Por padrão, esse parâmetro não está definido. Ativar esse parâmetro pode ajudar a encontrar consultas não otimizadas.
log_statement	–	Define o tipo de instruções registradas. Por padrão, esse parâmetro não está definido, mas você pode alterá-lo para `all`, `ddl` ou `mod` para especificar os tipos de declaração SQL que você deseja registrar. Se você especificar algo diferente de `none` para esse parâmetro, você também deve tomar medidas adicionais para evitar a exposição de senhas nos arquivos de log. Para ter mais informações, consulte Reduzir o risco de exposição de senhas ao usar o registro em log de consultas.
log_statement_sample_rate	–	A porcentagem de declarações que excedem o tempo especificado em `log_min_duration_sample` para serem registradas, expressa como um valor de ponto flutuante entre 0,0 e 1,0.
log_statement_stats	–	Grava estatísticas de desempenho cumulativas no log do servidor.

Usar o registro em log para encontrar consultas de baixa performance

Você pode registrar consultas e declarações SQL para ajudar a encontrar consultas com a performance lenta. Você ativa esse recurso modificando as configurações dos parâmetros `log_statement` e `log_min_duration` conforme descrito nesta seção. Antes de ativar o registro em log de consultas para seu cluster de banco de dados do Aurora PostgreSQL, você deve estar ciente da possível exposição de senhas nos logs e de como reduzir os riscos. Para ter mais informações, consulte Reduzir o risco de exposição de senhas ao usar o registro em log de consultas.

A seguir, você encontrará informações de referência sobre os parâmetros `log_statement` e `log_min_duration`.

log_statement

Esse parâmetro especifica o tipo de declarações SQL que devem ser enviadas ao log. O valor padrão é `none`. Se você alterar esse parâmetro para `all`, `ddl` ou `mod`, realize algumas das ações recomendadas para reduzir o risco de expor senhas nos logs. Para ter mais informações, consulte Reduzir o risco de exposição de senhas ao usar o registro em log de consultas.

todas

Registra todas as declarações. Essa configuração é recomendada para fins de depuração.

ddl

Registra todas as declarações de linguagem de definição de dados (DDL), como CREATE, ALTER, DROP etc.

mod

Registra todas as declarações DDL e declarações de linguagem de manipulação de dados (INSERT, UPDATE e DELETE) que modificam os dados.

nenhuma

Nenhuma declaração SQL é registrada. Recomendamos essa configuração para evitar o risco de expor senhas nos logs.

log_min_duration_statement

Todas as instruções SQL executadas pelo menos por um período especificado ou mais é registrada. Por padrão, esse parâmetro não está definido. Ativar esse parâmetro pode ajudar a encontrar consultas não otimizadas.

–1–2147483647

O número de milissegundos (ms) de tempo de execução durante o qual uma declaração é registrada.

Como configurar o registro em log de consultas

Essas etapas pressupõem que o cluster de banco de dados do Aurora PostgreSQL use um grupo de parâmetros de cluster de banco de dados personalizado.

1. Defina o parâmetro `log_statement` como `all`. O exemplo a seguir mostra a informação gravada no arquivo `postgresql.log`com essa configuração de parâmetro.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
```

```
---------------------- END OF LOG ----------------------
```

2. Defina o parâmetro `log_min_duration_statement`. O exemplo a seguir mostra a informação gravada no arquivo `postgresql.log` quando o parâmetro estiver definido como 1.

   As consultas que excedem a duração especificada no parâmetro `log_min_duration_statement` são registradas. Por exemplo: Você pode visualizar o arquivo de log de seu cluster de banco de dados do Aurora PostgreSQL no console do Amazon RDS.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
---------------------- END OF LOG ----------------------
```

Reduzir o risco de exposição de senhas ao usar o registro em log de consultas

Recomendamos manter `log_statement` definido como `none` para evitar a exposição de senhas. Se você definir `log_statement` como `all`, `ddl` oumod, recomendamos que você siga uma ou mais destas etapas.

- Para o cliente, criptografe informações confidenciais. Para ter mais informações consulte Encryption Options (Opções de criptografia) na documentação do PostgreSQL. Use as opções `ENCRYPTED` (e `UNENCRYPTED`) das declarações `CREATE` e `ALTER`. Para ter mais informações, consulte CREATE USER na documentação do PostgreSQL.
- Para seu cluster de banco de dados do Aurora PostgreSQL, configure e use a extensão de auditoria do PostgreSQL (pgAudit). Essa extensão remove informações confidenciais das declarações CREATE e ALTER enviadas ao log. Para ter mais informações, consulte Usar pgAudit para registrar a atividade do banco de dados.
- Restringir o acesso aos logs CloudWatch.
- Use mecanismos de autenticação mais fortes, como IAM.

Monitorar chamadas de API do Amazon Aurorano AWS CloudTrail

O AWS CloudTrail é um serviço da AWS que ajuda a auditar a conta da AWS. O AWS CloudTrail é ativado na sua conta da AWS quando ela é criada. Para obter mais informações sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Tópicos

- Integração do CloudTrail com o Amazon Aurora
- Entradas do arquivo de log do Amazon Aurora

Integração do CloudTrail com o Amazon Aurora

Todas as ações do Amazon Aurora são registradas em log pelo CloudTrail. O CloudTrail fornece um registro de ações executadas por um usuário, uma função ou um serviço da AWS no Amazon Aurora.

Eventos do CloudTrail

O CloudTrail captura as chamadas de API do Amazon Aurora como eventos. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação e assim por diante. Os eventos incluem as chamadas do console do Amazon RDS e as chamadas de código para as operações de API do Amazon RDS.

A atividade do Amazon Aurora é registrada em um evento do CloudTrail em Event history (Histórico de eventos). Você pode usar o console do CloudTrail para visualizar os últimos 90 dias de eventos e de atividades de API registrados em uma região da AWS. Para obter mais informações, consulte Visualizar eventos com o histórico de eventos do CloudTrail.

Trilhas do CloudTrail

Para obter um registro contínuo de eventos na conta da AWS, incluindo eventos do Amazon Aurora, crie uma trilha. Uma trilha é uma configuração que permite a entrega de eventos a um bucket do Amazon S3 especificado. O CloudTrail normalmente fornece os arquivos de log em até 15 minutos após uma atividade da conta.

> **Note**
>
> Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail no Histórico de eventos.

É possível criar dois tipos de trilhas para uma conta da AWS: uma trilha que se aplica a todas as regiões ou uma trilha que se aplica a uma região. Por padrão, quando uma trilha é criada no console, a mesma é aplicada a todas as regiões da .

Além disso, é possível configurar outros serviços da AWS para analisar e agir melhor com base nos dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configurar notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e Receber arquivos de log do CloudTrail de várias contas

Entradas do arquivo de log do Amazon Aurora

Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada das chamadas de API pública. Dessa forma, eles não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação `CreateDBInstance`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
```

```
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
```

```
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
```

```
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Conforme mostrado no elemento `userIdentity` no exemplo anterior, cada evento ou entrada de log contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário-raiz ou usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações sobre o `userIdentity`, consulte o Elemento userIdentity do CloudTrail. Para obter mais informações sobre `CreateDBInstance` e outras ações do Amazon Aurora, consulte a Referência da API do Amazon RDS.

Monitorar o Amazon Aurora com o recurso Database Activity Streams

Usando o recurso Database Activity Streams, você pode monitorar fluxos quase em tempo real da atividade de um banco de dados.

Tópicos

- Visão geral dos Database Activity Streams
- Pré-requisitos de rede para fluxos de atividades de banco de dados Aurora MySQL
- Iniciar um stream de atividade de banco de dados
- Obter o status de um fluxo de atividade de banco de dados
- Interromper um fluxo de atividade de banco de dados
- Monitorar fluxos de atividades de banco de dados
- Exemplos de políticas do IAM para fluxos de atividades de banco de dados

Visão geral dos Database Activity Streams

Como administrador de um banco de dados do Amazon Aurora, você precisa proteger o banco de dados e atender aos requisitos regulatórios e de compatibilidade. Uma estratégia é integrar fluxos de atividades de banco de dados às suas ferramentas de monitoramento. Dessa forma, você monitora e define alarmes para realizar a auditoria da atividade no cluster do Amazon Aurora.

As ameaças à segurança são externas e internas. Para se proteger contra ameaças internas, você pode controlar o acesso do administrador aos transmissões de dados configurando o recurso Database Activity Streams. Os administradores de bancos de dados não têm acesso à coleta, transmissão, armazenamento e processamento das transmissões.

Sumário

- Como os fluxos de atividade do banco de dados funcionam
- Modo assíncrono e síncronopara fluxos de atividades de banco de dados
- Requisitos e limitações para fluxos de atividade de banco de dados
- Disponibilidade de regiões e versões
- Classes de instância de banco de dados compatíveis para transmissões de atividades de banco de dados

Como os fluxos de atividade do banco de dados funcionam

No Amazon Aurora, você inicia um fluxo de atividades de banco de dados no nível do cluster. Todas as instâncias de banco de dados no cluster têm fluxos de atividades de banco de dados habilitados.

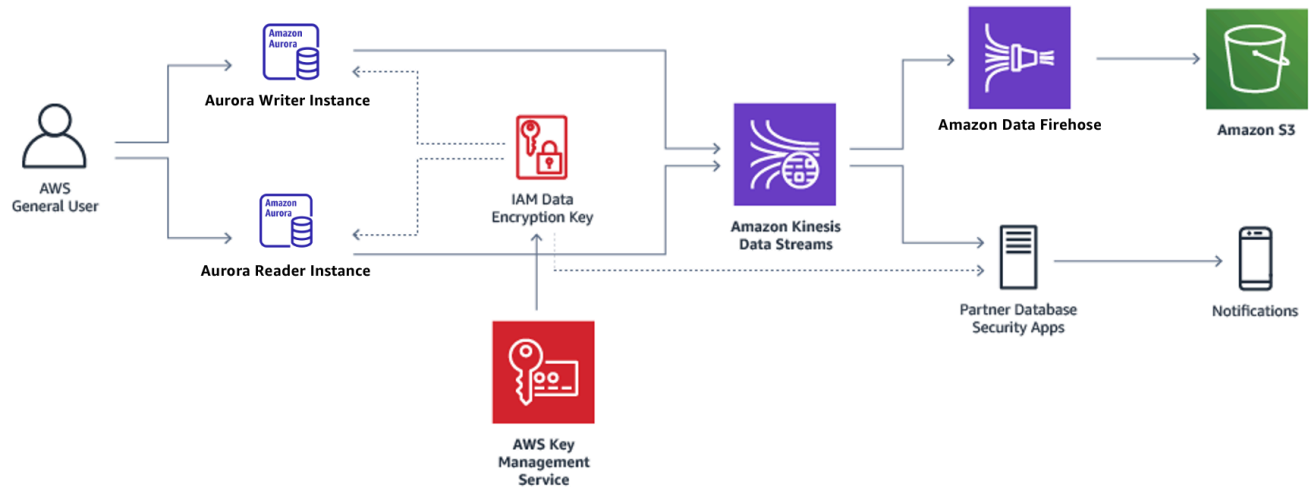
Seu cluster de banco de dados do Aurora envia atividades para um fluxo de dados do Amazon Kinesis quase em tempo real. O fluxo do Kinesis é criado automaticamente. No Kinesis, é possível configurar serviços da AWS, como o Amazon Data Firehose e o AWS Lambda, para consumir o fluxo e armazenar os dados.

> **Important**
>
> O uso do recurso Database Activity Streams é gratuito no Amazon Aurora, mas o Amazon Kinesis cobra pelo fluxo de dados. Para ter mais informações, consulte Definição de preço do Amazon Kinesis Data Streams.

Se você usar um banco de dados global do Aurora, inicie um fluxo de atividades de banco de dados em cada cluster de banco de dados separadamente. Cada cluster fornece dados de auditoria ao seu próprio fluxo do Kinesis dentro de sua própria Região da AWS. Os fluxos de atividade não funcionam de forma diferente durante um failover. Eles continuam a auditar seu banco de dados global como de costume.

É possível configurar aplicações para gerenciamento de conformidade com o objetivo de consumir fluxos de atividades do banco de dados. No Aurora PostgreSQL, as aplicações de conformidade incluem o IBM Security Guardium e o Imperva SecureSphere Database Audit and Protection. Essas aplicações podem usar o fluxo para gerar alertas e auditar atividades em seu cluster de banco de dados Aurora.

O gráfico a seguir mostra um cluster de banco de dados do Aurora configurado com o Amazon Data Firehose.



Modo assíncrono e síncronopara fluxos de atividades de banco de dados

Você pode decidir se prefere que a sessão de banco de dados processe os eventos de atividade de banco de dados em um destes modos:

- Modo assíncrono: quando uma sessão de banco de dados gera um evento de fluxo de atividade, a sessão retorna às atividades normais imediatamente. Em segundo plano, o evento do fluxo de atividade é tornado um registro durável. Se ocorrer um erro na tarefa de segundo plano, é enviado um evento do RDS. Este evento indica o início e o final de qualquer janela de tempo em que os registros do evento do fluxo de atividade podem ter sido perdidos.

  O modo assíncrono favorece a performance do banco de dados sobre a precisão do fluxo de atividade.

  > **Note**
  >
  > O modo assíncrono está disponível para o Aurora PostgreSQL e o Aurora MySQL.

- Modo síncrono: quando uma sessão de banco de dados gera um evento de fluxo de atividade, a sessão bloqueia outras atividades até que o evento torne-se durável. Se não puder tornar o evento por algum motivo, a sessão de banco de dados retorna às atividades normais. No entanto, um evento do RDS é enviado, indicando que os registros do fluxo de atividade podem ser perdidos por algum tempo. Um segundo evento do RDS é enviado depois que o sistema voltar a um estado íntegro.

O modo síncrono favorece a precisão do fluxo de atividade sobre a performance do banco de dados.

> **Note**
>
> O modo síncrono está disponível para o Aurora PostgreSQL. Não é possível usar o modo síncrono com o Aurora MySQL.

Requisitos e limitações para fluxos de atividade de banco de dados

No Aurora, os fluxos de atividade de banco de dados têm os seguintes requisitos e limites:

- O Amazon Kinesis é imprescindível para os fluxos de atividades do banco de dados.
- O AWS Key Management Service (AWS KMS) é imprescindível para fluxos de atividades de banco de dados, pois eles são sempre criptografados.
- A aplicação de criptografia adicional ao seu fluxo de dados do Amazon Kinesis é incompatível com fluxos de atividade do banco de dados, que já estão criptografados com sua chave do AWS KMS.
- Inicie o fluxo de atividades do banco de dados no nível do cluster de banco de dados. Se você adicionar uma instância de banco de dados ao cluster, não precisará iniciar um fluxo de atividades na instância: ele é auditado automaticamente.
- No banco de dados global do Aurora, inicie um fluxo de atividades em cada cluster de banco de dados separadamente. Cada cluster fornece dados de auditoria ao seu próprio fluxo do Kinesis dentro de sua própria Região da AWS.
- No Aurora PostgreSQL, interrompa o fluxo de atividades do banco de dados antes de uma atualização de versão principal. Você poderá iniciar o fluxo de atividades do banco de dados após o término da atualização.

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e regiões com o Aurora e fluxos de atividades de banco de dados, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com fluxos de atividades de banco de dados.

Classes de instância de banco de dados compatíveis para transmissões de atividades de banco de dados

Para o Aurora MySQL, é possível utilizar fluxos de atividades de banco de dados com as seguintes classes de instâncias de banco de dados:

- db.r8g.*large
- db.r7g.*large
- db.r7i.*large
- db.r6g.*large
- db.r6i.*large
- db.r5.*large
- db.x2g.*

Para o Aurora PostgreSQL, é possível utilizar fluxos de atividades de banco de dados com as seguintes classes de instâncias de banco de dados:

- db.r8g.*large
- db.r7i.*large
- db.r7g.*large
- db.r6g.*large
- db.r6i.*large
- db.r6id.*large
- db.r5.*large
- db.r4.*large
- db.x2g.*

Pré-requisitos de rede para fluxos de atividades de banco de dados Aurora MySQL

Na seção a seguir, você aprende a configurar sua nuvem privada virtual (VPC) para uso com transmissões de atividades de banco de dados.

> **Note**
>
> Os pré-requisitos de rede do Aurora MySQL são aplicáveis às seguintes versões do mecanismo:
>
> - Aurora MySQL versão 2, até 2.11.3
> - Aurora MySQL versão 2.12.0
> - Aurora MySQL versão 3, até 3.04.2

Tópicos

- Pré-requisitos para endpoints do AWS KMS
- Pré-requisitos para a disponibilidade pública
- Pré-requisitos para a disponibilidade privada

Pré-requisitos para endpoints do AWS KMS

Instâncias em um cluster Aurora MySQL que usam fluxos de atividade que devem ser capazes de acessar endpoints do AWS KMS. Certifique-se de que este requisito esteja satisfeito antes de ativar fluxos de atividades de banco de dados para o Aurora MySQL cluster. Se o cluster do Aurora estiver disponível publicamente, esse requisito será cumprido automaticamente.

> **Important**
>
> Se o cluster Aurora MySQL de banco de dados não conseguir acessar o endpoint do AWS KMS, o fluxo de atividade será interrompido. Nesse caso, o Aurora notifica você sobre esse problema usando os eventos do RDS.

Pré-requisitos para a disponibilidade pública

Para que um cluster de bancos de dados Aurora seja público, deverá atender aos seguintes requisitos:

- Na página de detalhes do cluster do Console de gerenciamento da AWS, a opção Publicly Accessible (Acessível para o público) será Yes (Sim).

- O cluster de banco de dados está em uma sub-rede pública da Amazon VPC. Para obter mais informações sobre instâncias de bancos de dados publicamente acessíveis, consulte Trabalhar com um cluster de banco de dados em uma VPC. Para obter mais informações sobre sub-redes públicas da Amazon VPC, consulte VPC e sub-redes.

Pré-requisitos para a disponibilidade privada

Se o cluster de banco de dados do Aurora estiver em uma sub-rede pública de VPC e não estiver acessível ao público, ele é privado. Para manter o seu cluster privado e usá-lo com fluxos de atividades de banco de dados, você pode optar pelo seguinte:

- Configure a conversão de endereços de rede (NAT) em sua VPC. Para obter mais informações, consulte Gateways NAT.
- Crie um endpoint do AWS KMS na sua VPC. Recomenda-se essa opção por ser mais fácil de configurar.

Para criar um endpoint do AWS KMS na sua VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, escolha Endpoints.
3. Escolha Create Endpoint (Criar endpoint).

   Aparecerá a página Create Endpoint (Criar endpoint).
4. Faça o seguinte:
   - Em Categoria de serviço, selecione Serviços da AWS.
   - Em Service Name (Nome do serviço), escolha com.amazonaws.***region***.kms, onde *region* é a Região da AWS na qual o cluster está localizado.
   - Em VPC, escolha a VPC em que o seu cluster se localiza.
5. Escolha Create Endpoint (Criar endpoint).

Para obter mais informações sobre como configurar VPC endpoints, consulte VPC Endpoints.

Iniciar um stream de atividade de banco de dados

Para monitorar a atividade de banco de dados para todas as instâncias do cluster de banco de dados do Aurora, inicie um fluxo de atividades no nível do cluster. As instâncias de banco de dados adicionadas ao cluster também são monitoradas automaticamente. Se você usar um banco de dados global do Aurora, inicie um fluxo de atividades de banco de dados em cada cluster de banco de dados separadamente. Cada cluster fornece dados de auditoria ao seu próprio fluxo do Kinesis dentro de sua própria Região da AWS.

Ao iniciar um fluxo de atividade, cada evento da atividade do banco de dados que você configurou na política de auditoria gera um evento do fluxo de atividade. Comandos SQL, como `CONNECT` e `SELECT`, geram eventos de acesso. Comandos SQL, como `CREATE` e `INSERT`, geram eventos de alteração.

Console

Para iniciar um fluxo de atividade de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados em que deseja iniciar um fluxo de atividade.
4. Em Actions (Ações), escolha Start activity (Iniciar atividade).

   A janela Start database activity stream: *name* (Iniciar fluxo de atividade do banco de dados: nome) aparece. Nela, *name* (nome) é o seu cluster de banco de dados.
5. Insira as seguintes configurações:

   - Em AWS KMS key, escolha uma chave na lista de AWS KMS keys.

     > **Note**
     >
     > Se o cluster do Aurora MySQL não conseguir acessar as chaves do KMS, siga as instruções presentes em Pré-requisitos de rede para fluxos de atividades de banco de dados Aurora MySQL para habilitar esse acesso primeiro.

     O Aurora usa a chave do KMS para criptografar a chave que, por sua vez, criptografa a atividade do banco de dados. Escolha uma chave do KMS diferente da chave padrão. Para

obter mais informações sobre as chaves de criptografia e o AWS KMS, consulte O que é o AWS Key Management Service? no Guia do desenvolvedor do AWS Key Management Service.

- Para Database activity stream mode (Modo do fluxo de atividade de banco de dados), escolha Asynchronous (Assíncrono) ou Synchronous (Síncrono).

  > **Note**
  >
  > Essa opção só se aplica ao Aurora PostgreSQL. Para Aurora MySQL, é possível usar apenas o modo assíncrono.

- Escolha Immediately (Imediatamente).

  Quando você escolhe Immediately (Imediatamente), o cluster de banco de dados reinicia imediatamente. Se você escolher During the next maintenance window, (Durante a próxima janela de manutenção), o cluster de banco de dados não reinicia imediatamente. Nesse caso, o stream de atividade do banco de dados não é iniciado até a próxima janela de manutenção.

6. Selecione Start database activity stream (Iniciar fluxo de atividades do banco de dados).

   O status para o cluster de banco de dados mostra que o fluxo de atividade está começando.

   > **Note**
   >
   > Se você receber o erro `You can't start a database activity stream in this configuration`, confira Classes de instância de banco de dados compatíveis para transmissões de atividades de banco de dados para ver se seu cluster de banco de dadossua instância do RDS está usando uma classe de instância compatível.

AWS CLI

Para iniciar fluxos de atividade de banco de dados para um cluster de banco de dados, configure o cluster de banco de dados utilizando o comando start-activity-stream da AWS CLI.

- `--resource-arn` *`arn`*: especifica o nome do recurso da Amazon (ARN) do cluster de banco de dados.

- `--mode` *`sync-or-async`*: especifica o modo síncrono (`sync`) ou assíncrono (`async`). No Aurora PostgreSQL, é possível escolher qualquer um dos valores. No Aurora MySQL, especifique `async`.
- `--kms-key-id` *`key`*: especifica o identificador de chave KMS para criptografar mensagens no fluxo de atividade do banco de dados. O identificador da chave do KMS da AWS é o ARN da chave, o ID da chave ou o ARN do alias ou o nome do alias para AWS KMS key.

O exemplo a seguir inicia um fluxo de atividade de banco de dados para um cluster de banco de dados no modo assíncrono.

Para Linux, macOS ou Unix:

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-cluster-arn \
    --apply-immediately
```

Para Windows:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-cluster-arn ^
    --apply-immediately
```

Amazon RDS API

Para iniciar fluxos de atividade de banco de dados para um cluster de banco de dados, configure o cluster utilizando a operação StartActivityStream.

Chame a ação com os parâmetros abaixo:

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`

> **Note**
>
> Se você receber um erro informando que não é possível iniciar um fluxo de atividades do banco de dados com a versão atual do banco de dados Aurora PostgreSQL, aplique o patch mais recente para o Aurora PostgreSQL antes de iniciar um fluxo de atividades do banco de dados. Para ter informações sobre atualizar o banco de dados Aurora PostgreSQL, consulte Atualização dos clusters de banco de dados do Amazon Aurora.
> A seguir estão as versões mínimas de patch para iniciar fluxos de atividade de banco de dados com o Aurora PostgreSQL.
>
> - 3.4.15 (11.9.15), 11.21.10
> - 12.9.15, 12.15.9, 12.16.10, 12.17.7, 12.18.5, 12.19.4, 12.20.3, 12.22.3
> - 13.9.12, 13.11.9, 13.12.10, 13.13.7, 13.14.5, 13.15.4, 13.16.3, 13.18.3
> - 14.6.12, 14.8.9, 14.9.10, 14.10.7, 14.11.5, 14.12.4, 14.13.3, 14.15.3
> - 15.3.9, 15.4.10, 15.5.7, 15.6.5, 15.7.4, 15.8.3, 15.10.3
> - 16.1.7, 16.2.5, 16.3.4, 16.4.3, 16.6.3

Obter o status de um fluxo de atividade de banco de dados

É possível obter o status de um fluxo de atividade usando o console ou a AWS CLI.

Console

Para obter o status de um fluxo de atividade de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha o link do cluster de banco de dados.
3. Escolha a guia Configuração e verifique o status do Stream de atividades do banco de dados.

AWS CLI

É possível obter a configuração do fluxo de atividade para um cluster de banco de dados como a resposta a uma solicitação de CLI describe-db-clusters.

O exemplo a seguir descreve *my-cluster*.

```
aws rds --region my-region describe-db-clusters --db-cluster-identifier my-cluster
```

O exemplo a seguir mostra uma resposta JSON. Os campos a seguir são mostrados:

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`
- `ActivityStreamMode`
-

Estes campos são os mesmos para o Aurora PostgreSQL e o Aurora MySQL, exceto `ActivityStreamMode` que está sempre `async` para o Aurora MySQL; enquanto para o Aurora PostgreSQL, ele pode ser `sync` ou `async`.

```
{
    "DBClusters": [
        {
      "DBClusterIdentifier": "my-cluster",
            ...
            "ActivityStreamKinesisStreamName": "aws-rds-das-cluster-
A6TSYXITZCZXJHIRVFUBZ5LTWY",
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "12345678-abcd-efgh-ijkl-bd041f170262",
            "ActivityStreamMode": "async",
            "DbClusterResourceId": "cluster-ABCD123456"
            ...
        }
    ]
}
```

API do RDS

É possível obter a configuração do fluxo de atividades para um cluster de banco de dados como a resposta a uma operação DescribeDBClusters.

Interromper um fluxo de atividade de banco de dados

É possível interromper um fluxo de atividade usando o console ou a AWS CLI.

Se você excluir seu cluster de banco de dados, o fluxo de atividade será interrompido e o fluxo subjacente do Amazon Kinesis será excluído automaticamente.

Console

Como desativar um fluxo de atividade

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha um cluster de banco de dados de onde deseja interromper o fluxo de atividade de banco de dados.
4. Em Actions (Ações), escolha Stop activity (Interromper atividade). A janela Database Activity Stream (Fluxo de atividade de banco de dados) é exibida.
   a. Escolha Immediately (Imediatamente).

      Quando você escolhe Immediately (Imediatamente), o cluster de banco de dados reinicia imediatamente. Se você escolher During the next maintenance window, (Durante a próxima janela de manutenção), o cluster de banco de dados não reinicia imediatamente. Nesse caso, o fluxo de atividade do banco de dados não será interrompido até a próxima janela de manutenção.
   b. Escolha Continue.

AWS CLI

Para interromper fluxos de atividades de banco de dados para seu cluster de banco de dados, configure o cluster de banco de dados usando o comando da AWS CLI stop-activity-stream. Identifique a região da AWS para o cluster de banco de dados usando o parâmetro `--region`. O parâmetro `--apply-immediately` é opcional.

Para Linux, macOS ou Unix:

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_CLUSTER_ARN \
    --apply-immediately
```

Para Windows:

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_CLUSTER_ARN ^
    --apply-immediately
```

API do RDS

Para interromper os fluxos de atividades de banco de dados para o cluster de banco de dados, configure o cluster usando a operação StopActivityStream. Identifique a região da AWS para o cluster de banco de dados usando o parâmetro `Region`. O parâmetro `ApplyImmediately` é opcional.

Monitorar fluxos de atividades de banco de dados

Os fluxos de atividades de banco de dados monitoram e relatam atividades. O stream de atividade é coletado e transmitido para o Amazon Kinesis. No Kinesis, é possível monitorar o stream de atividades ou outros serviços e aplicações podem consumir o stream de atividades para análise posterior. É possível encontrar o nome do fluxo subjacente do Kinesis utilizando o comando `describe-db-clusters` da AWS CLI ou a operação `DescribeDBClusters` da API do RDS.

O Aurora gerencia o fluxo do Kinesis para você da seguinte forma:

- O Aurora cria o fluxo do Kinesis automaticamente com um período de retenção de 24 horas.
- O Aurora escala o fluxo do Kinesis, se necessário.
- Se você interromper o fluxo de atividades do banco de dados ou excluir o cluster de banco de dados, o Aurora excluirá o fluxo do Kinesis.

As categorias de atividade a seguir são monitoradas e colocadas no log de auditoria do fluxo de atividade:

- Comandos SQL: todos os comandos SQL são auditados e também instruções preparadas, funções integradas e funções em PL/SQL. Chamadas para procedimentos armazenados são auditadas. Quaisquer instruções SQL emitidas dentro de procedimentos armazenados ou funções também são auditadas.
- Outras informações do banco de dados: atividades monitoradas incluem a instrução SQL completa, a contagem de linhas afetadas dos comandos DML, os objetos acessados e o nome do banco de dados exclusivo. No Aurora PostgreSQL, os fluxos de atividades de banco de dados também monitoram as variáveis de ligação e os parâmetros de procedimento armazenados.

> **Important**
>
> O texto SQL completo de cada instrução é visível no log de auditoria do stream de atividades, incluindo quaisquer dados confidenciais. No entanto, as senhas do usuário do banco de dados serão editadas se o Aurora puder determiná-las pelo contexto, como na instrução SQL a seguir.
>
> ```
> ALTER ROLE role-name WITH password
> ```

- Informações de conexão: a atividade monitorada inclui informações de rede e sessão, o ID do processo de servidor e códigos de saída.

Se um fluxo de atividade tiver uma falha ao monitorar uma instância de banco de dados, você será notificado por meio de eventos do RDS.

Nas seções a seguir, é possível acessar, auditar e processar fluxos de atividades do banco de dados.

Tópicos

- Acessar um fluxo de atividades no Amazon Kinesis
- Conteúdo do log de auditoria e exemplos de log de fluxos de atividade do banco de dados
- Matriz JSON databaseActivityEventList para fluxos de atividades de banco de dados
- Processar um fluxo de atividade usando o SDK da AWS

Acessar um fluxo de atividades no Amazon Kinesis

Ao habilitar um fluxo de atividade para um cluster de banco de dados, um fluxo do Kinesis será criado para você. No Kinesis, você pode monitorar a atividade de banco de dados em tempo real. Para analisar detalhadamente a atividade de banco de dados, é possível conectar o stream do Kinesis para aplicações consumidoras. Também é possível conectar o fluxo a aplicações de gerenciamento de compatibilidade como o IBM Security Guardium ou Imperva SecureSphere Database Audit and Protection.

Você pode acessar seu fluxo do Kinesis a partir do console do RDS ou do console do Kinesis.

Como acessar um fluxo de atividade pelo Kinesis usando o console do RDS

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados em que iniciou um fluxo de atividade.
4. Escolher configuração.
5. Em Database activity stream (Fluxo de atividades do banco de dados), clique no link em Kinesis stream (Fluxo do Kinesis).
6. No console do Kinesis, selecione Monitoring (Monitoramento) para começar a observar a atividade de banco de dados.

Como acessar um fluxo de atividade pelo Kinesis usando o console do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. Escolha o fluxo de atividade na lista de fluxos do Kinesis.

   O nome de um fluxo de atividade contém o prefixo `aws-rds-das-cluster-` seguido pelo ID de recurso do cluster de banco de dados. Veja um exemplo a seguir.

   ```
   aws-rds-das-cluster-NHVOV4PCLWHGF52NP
   ```

   Para utilizar o console do Amazon RDS para encontrar o ID do recurso do cluster de banco de dados, escolha o cluster de banco de dados na lista de bancos de dados e escolha a guia Configuration (Configuração).

   Para utilizar a AWS CLI a fim de encontrar o nome completo do fluxo do Kinesis para um fluxo de atividades, use uma solicitação de CLI describe-db-clusters e anote os valores de `ActivityStreamKinesisStreamName` na resposta.
3. Escolha Monitoring (Monitoramento) para começar a observar a atividade de banco de dados.

Para obter mais informações sobre como usar o Amazon Kinesis, consulte O que é o Amazon Kinesis Data Streams?.

Conteúdo do log de auditoria e exemplos de log de fluxos de atividade do banco de dados

Os eventos monitorados são representados no fluxo de atividade do banco de dados como strings JSON. A estrutura consiste em um objeto JSON que contém um `DatabaseActivityMonitoringRecord`, que, por sua vez, contém uma matriz `databaseActivityEventList` de eventos de atividade.

> **Note**
>
> Para fluxos de atividades do banco de dados, a matriz JSON `paramList` não inclui valores nulos de aplicações Hibernate.

Tópicos

- Exemplos de log de auditoria para fluxos de atividade
- Objeto JSON DatabaseActivityMonitoringRecords
- Objeto JSON databaseActivityEvents

Exemplos de log de auditoria para fluxos de atividade

Veja a seguir exemplos de log de auditoria JSON descriptografados de registros de evento de atividade.

Example Registro de eventos de atividade de uma instrução SQL CONNECT do Aurora PostgreSQL instrução SQL CONNECT

O registro de eventos de atividade a seguir mostra um login com o uso de uma instrução SQL `CONNECT` (`command`) por um cliente psql (`clientApplication`).

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
```

```
        {
          "startTime": "2019-10-30 00:39:49.940668+00",
          "logTime": "2019-10-30 00:39:49.990579+00",
          "statementId": 1,
          "substatementId": 1,
          "objectType": null,
          "command": "CONNECT",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "49804",
          "sessionId": "5ce5f7f0.474b",
          "rowCount": null,
          "commandText": null,
          "paramList": [],
          "pid": 18251,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "MISC",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Registro de evento de atividade de uma instrução SQL CONNECT do Aurora MySQL

O registro de eventos de atividade a seguir mostra um login com o uso de uma instrução SQL `CONNECT` (`command`) por um cliente mysql (`clientApplication`).

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
```

```
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:13.267214+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"rdsadmin",
      "databaseName":"",
      "remoteHost":"localhost",
      "remotePort":"11053",
      "command":"CONNECT",
      "commandText":"",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"",
      "statementId":0,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725121",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:13.267207+00",
      "endTime":"2020-05-22 18:07:13.267213+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```


Example Registro de evento de atividades de uma instrução CREATE TABLE do Aurora PostgreSQL

O exemplo a seguir mostra um evento `CREATE TABLE` para o Aurora PostgreSQL.


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
```

```
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:36:54.403455+00",
          "logTime": "2019-05-24 00:36:54.494235+00",
          "statementId": 2,
          "substatementId": 1,
          "objectType": null,
          "command": "CREATE TABLE",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": null,
          "commandText": "create table my_table (id serial primary key, name
 varchar(32));",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "DDL",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```


Example Registro de evento de atividade de uma instrução CREATE TABLE do Aurora MySQL

O exemplo a seguir mostra uma instrução `CREATE TABLE` para o Aurora MySQL. A operação é representada como dois registros de eventos separados. Um evento tem `"class":"MAIN"`. O

outro evento tem `"class":"AUX"`. As mensagens podem chegar em qualquer ordem. O campo `logTime` do evento `MAIN` é sempre anterior aos campos `logTime` de quaisquer eventos `AUX` correspondentes.

O exemplo a seguir mostra o evento com um valor `class` de `MAIN`.


```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.250221+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"CREATE TABLE test1 (id INT)",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.250222+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
```

```
}
```

O exemplo a seguir mostra o evento correspondente a um valor `class` de AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.247182+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"CREATE",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.247182+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Example Registro de evento de atividades de uma instrução SELECT do Aurora PostgreSQL

O exemplo a seguir mostra um evento `SELECT` .


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:39:49.920564+00",
          "logTime": "2019-05-24 00:39:49.940668+00",
          "statementId": 6,
          "substatementId": 1,
          "objectType": "TABLE",
          "command": "SELECT",
          "objectName": "public.my_table",
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": 10,
          "commandText": "select * from my_table;",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "READ",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
```

```
   "key":"decryption-key"
}
```


```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
```

```
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x0105000000000005150000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

Example Registro de evento de atividade de uma instrução SELECT do Aurora MySQL

O exemplo a seguir mostra um evento `SELECT`.

O exemplo a seguir mostra o evento com um valor `class` de `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986467+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
```

```
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"SELECT * FROM test1 WHERE id < 28",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"726571",
      "rowCount":2,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986467+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

O exemplo a seguir mostra o evento correspondente a um valor `class` de AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986399+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"READ",
```

```
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"726571",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986399+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Objeto JSON DatabaseActivityMonitoringRecords

Os registros de eventos de atividade de banco de dados estão em um objeto JSON que contém as informações a seguir.

Campo JSON	Tipo de dados	Descrição
`type`	string	O tipo de registro JSON. O valor é `DatabaseActivityMonitoringRecords` .
`version`	string	A versão dos registros de monitoramento de atividade de banco de dados. A versão dos registros de atividade de banco de dados gerados depende da versão do mecanismo do cluster de banco de dados:

Campo JSON	Tipo de dados	Descrição
		• Os registros de atividade de banco de dados versão 1.1 são gerados para clusters de banco de dados Aurora PostgreSQL que executam as versões do mecanismo 10.10, as versões secundárias posteriores e as versões 11.5 e posteriores. • Os registros de atividade de banco de dados versão 1.0 são gerados para clusters de banco de dados Aurora PostgreSQL que executam as versões do mecanismo 10.7 e 11.4. Todos os campos a seguir estão na versão 1.0 e na versão 1.1, exceto quando indicado especificamente.
databaseActivityEvents	string	Um objeto JSON que contém os eventos de atividade.
key	string	Uma chave de criptografia que você usa para descriptografar o Matriz JSON databaseActivityEventList

Objeto JSON databaseActivityEvents

O objeto JSON `databaseActivityEvents` contém as informações a seguir.

Campos de nível superior no registro JSON

Cada evento no log de auditoria é incluído em um registro no formato JSON. Esse registro contém os campos a seguir.

type

Esse campo sempre tem o valor `DatabaseActivityMonitoringRecords`.

versão

Esse campo representa a versão do contrato ou do protocolo de dados de streaming da atividade do banco de dados. Define quais campos estão disponíveis.

A versão 1.0 representa o suporte de fluxos de atividades de dados originais para o Aurora PostgreSQL versões 10.7 e 11.4. A versão 1.1 representa o suporte de fluxos de atividades de dados para o Aurora PostgreSQL versões 10.10 e posterior e o Aurora PostgreSQL 11.5 e posterior. A versão 1.1 inclui os campos adicionais `errorMessage` e `startTime`. A versão 1.2 representa o suporte de fluxos de atividades de dados para o Aurora MySQL 2.08 e posterior. A versão 1.2 inclui os campos adicionais `endTime` e `transactionId`.

databaseActivityEvents

Uma string criptografada que representa um ou mais eventos de atividade. Ela é representada como uma matriz de bytes base64. Quando você descriptografa a string, o resultado é um registro no formato JSON com campos, conforme mostrado nos exemplos nesta seção.

chave

A chave de dados criptografada usada para criptografar a string `databaseActivityEvents`. Esta é a mesma AWS KMS key que você forneceu ao iniciar o fluxo de atividade do banco de dados.

O exemplo a seguir mostra o formato desse registro.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

Siga as seguintes etapas para descriptografar o conteúdo do campo `databaseActivityEvents`:

1. Descriptografe o valor no campo JSON da `key` usando a chave do KMS que forneceu ao iniciar o fluxo de atividade do banco de dados. Fazer isso retorna a chave de criptografia de dados em texto não criptografado.

2. Decodifique em base64 o valor no campo JSON `databaseActivityEvents` para obter o texto cifrado, em formato binário, da carga útil de auditoria.
3. Descriptografe o texto cifrado binário com a chave de criptografia de dados que você decodificou na primeira etapa.
4. Descompacte a carga útil descriptografada.
   - A carga criptografada está no campo `databaseActivityEvents`.
   - O campo `databaseActivityEventList` contém uma matriz de registros de auditoria. Os campos `type` na matriz podem ser `record` ou `heartbeat`.

O registro do evento de atividade do log de auditoria é um objeto JSON que contém as informações a seguir.

Campo JSON	Tipo de dados	Descrição
`type`	string	O tipo de registro JSON. O valor é `DatabaseActivityMonitoringRecord` .
`clusterId`	string	O identificador de recurso de cluster de banco de dados. Corresponde ao atributo de cluster de banco de dados `DbClusterResourceId` .
`instanceId`	string	O identificador de recurso da instância de banco de dados. Ele corresponde ao atributo de instância de banco de dados `DbiResourceId` .
Matriz JSON databaseActivityEventList	string	Uma matriz de registros de auditoria de atividade ou mensagens de pulsação.

Matriz JSON databaseActivityEventList para fluxos de atividades de banco de dados

A carga do log de auditoria é uma matriz JSON `databaseActivityEventList` criptografada. A tabela a seguir lista em ordem alfabética os campos para cada evento de atividade na matriz `DatabaseActivityEventList` descriptografada de um log de auditoria. Os campos vão diferir, se

você usar o Aurora PostgreSQL ou o Aurora MySQL. Consulte a tabela que se aplica ao mecanismo de banco de dados.

> **Important**
>
> A estrutura do evento está sujeita a alterações. O Aurora pode adicionar novos campos a eventos de atividade no futuro. Em aplicações que analisam os dados JSON, confirme se o código pode ignorar ou executar ações apropriadas para nomes de campo desconhecidos.

Campos databaseActivityEventList para o Aurora PostgreSQL

Veja a seguir campos `databaseActivityEventList` do Aurora PostgreSQL.

Campo	Tipo de dados	Descrição
`class`	string	A classe do evento de atividade. Os valores válidos para o Aurora PostgreSQL são os seguintes: • `ALL` • `CONNECT`: um evento de conexão ou desconexão. • `DDL`: uma instrução DDS que não está incluída na lista de instruções da classe `ROLE`. • `FUNCTION`: uma chamada de função ou um bloco `DO`. • `MISC`: um comando diverso, como `DISCARD`, `FETCH`, `CHECKPOINT` ou `VACUUM`. • `NONE` • `READ`: uma instrução `SELECT` ou `COPY` quando a origem é uma relação ou uma consulta. • `ROLE`: uma instrução relacionada a funções e privilégios incluindo `GRANT`, `REVOKE` e `CREATE/ALTER/DROP ROLE`. • `WRITE`: uma instrução `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` ou `COPY` quando o destino é uma relação.

Campo	Tipo de dados	Descrição
`clientApplication`	string	O aplicativo que o cliente usou para conectar-se conforme reportado pelo cliente. O cliente não precisa fornecer essas informações, então o valor pode ser nulo.
`command`	string	O nome do comando SQL sem nenhum detalhe do comando.
`commandText`	string	A real instrução SQL transmitida pelo usuário. No Aurora PostgreSQL, o valor é idêntico à instrução SQL original. Este campo é usado para todos os tipos de registros, exceto para registros de conexão ou desconexão e, nesse caso, o valor é nulo. **Important** O texto SQL completo de cada instrução é visível no log de auditoria do stream de atividades, incluindo quaisquer dados confidenciais. No entanto, as senhas do usuário do banco de dados serão editadas se o Aurora puder determiná-las pelo contexto, como na instrução SQL a seguir. `ALTER ROLE role-name WITH password`
`databaseName`	string	O banco de dados ao qual o usuário se conectou.
`dbProtocol`	string	O protocolo de banco de dados, por exemplo `Postgres 3.0`.
`dbUserName`	string	O usuário do banco de dados com o qual o cliente foi autenticado.

Campo	Tipo de dados	Descrição
`errorMessage` (somente registros de atividade de banco de dados da versão 1.1)	string	Se houver algum erro, esse campo será preenchido com a mensagem de erro que seria gerada pelo servidor de banco de dados. O valor `errorMessage` é nulo para instruções normais que não resultaram em um erro. Um erro é definido como qualquer atividade que produziria um evento de log de erro PostgreSQL visível para o cliente em um nível de gravidade igual a `ERROR` ou superior. Para obter mais informações, consulte PostgreSQL Message Severity Levels. Por exemplo, erros de sintaxe e cancelamentos de consulta geram uma mensagem de erro. Erros internos do servidor PostgreSQL, como erros de processo de checkpointer em segundo plano, não geram uma mensagem de erro. No entanto, os registros para esses eventos ainda são emitidos independentemente da configuração do nível de gravidade do log. Isso impede que os invasores desativem o registro em log para tentar evitar a detecção. Veja também o campo `exitCode`.
`exitCode`	int	Um valor usado para um registro de saída de sessão. Em uma saída limpa, isso contém o código de saída. Um código de saída não pode ser sempre obtido em alguns cenários de falha. Exemplos são se o PostgreSQL fizer um `exit()` ou se um operador realizar um comando como um `kill -9`. Se houver algum erro, o campo `exitCode` exibirá o código de erro SQL, `SQLSTATE`, conforme listado em PostgreSQL Error Codes. Veja também o campo `errorMessage` .

Campo	Tipo de dados	Descrição
`logTime`	string	Um time stamp como registrado no caminho do código de auditoria. Isso representa a hora de término da execução da instrução SQL. Veja também o campo `startTime` .
`netProtocol`	string	O protocolo de comunicação da rede
`objectName`	string	O nome do objeto do banco de dados se a instrução SQL estiver operando em um. Este campo será usado somente quando a instrução SQL operar em um objeto de banco de dados. Se a instrução SQL não estiver operando em um objeto, esse valor será nulo.
`objectType`	string	O tipo do objeto de banco de dados, como tabela, índice, visualização etc. Este campo será usado somente quando a instrução SQL operar em um objeto de banco de dados. Se a instrução SQL não estiver operando em um objeto, esse valor será nulo. Entre os valores válidos estão os seguintes: • `COMPOSITE TYPE` • `FOREIGN TABLE` • `FUNCTION` • `INDEX` • `MATERIALIZED VIEW` • `SEQUENCE` • `TABLE` • `TOAST TABLE` • `VIEW` • `UNKNOWN`
`paramList`	string	Uma matriz de parâmetros separados por vírgulas transmiti dos à instrução SQL. Se a instrução SQL não tiver parâmetro s, esse valor será uma matriz vazia.

Campo	Tipo de dados	Descrição
`pid`	int	O ID do processo de backend que está alocado para atender à conexão do cliente.
`remoteHost`	string	O endereço IP do cliente ou o nome do host. No Aurora PostgreSQL, qual deles é usado depende da configuração do parâmetro `log_hostname` de banco de dados. O valor `remoteHost` também inclui `[local]` e `localhost` que indicam a atividade do usuário `rdsadmin`.
`remotePort`	string	O número da porta do cliente.
`rowCount`	int	O número de linhas da tabela afetadas ou recuperadas pela instrução SQL. Este campo é usado somente para instruções SQL que são instruções da linguage de manipulação de dados (DML). Se a instrução SQL não for uma instrução DML, esse valor será nulo.
`serverHost`	string	O endereço IP do host do servidor do banco de dados. O valor `serverHost` também inclui `[local]` e `localhost` que indicam a atividade do usuário `rdsadmin`.
`serverType`	string	O tipo de servidor do banco de dados, por exemplo `PostgreSQL` .
`serverVersion`	string	A versão do servidor de banco de dados, por exemplo, `2.3.1` para o Aurora PostgreSQL.
`serviceName`	string	O nome do serviço, por exemplo `Amazon Aurora PostgreSQL-Compatible edition` .
`sessionId`	int	Um identificador de sessão pseudo-exclusivo.
`sessionId`	int	Um identificador de sessão pseudo-exclusivo.

Campo	Tipo de dados	Descrição
`startTime` (somente registros de atividade de banco de dados da versão 1.1)	string	A hora em que a execução começou para a instrução SQL. Para calcular o tempo aproximado de execução da instrução SQL, use `logTime - startTime` . Veja também o campo `logTime`.
`statementId`	int	Um identificador para a instrução SQL do cliente. O contador está no nível da sessão e incrementa com cada instrução SQL inserida pelo cliente.
`substatementId`	int	Um identificador para uma substituição SQL. Esse valor conta as subinstruções contidas para cada instrução SQL identificada pelo campo `statementId` .
`type`	string	O tipo de evento. Os valores válidos são `record` ou `heartbeat` .

Campos databaseActivityEventList para o Aurora MySQL

Veja a seguir campos `databaseActivityEventList` do Aurora MySQL.

Campo	Tipo de dados	Descrição
`class`	string	A classe do evento de atividade. Os valores válidos para o Aurora MySQL são os seguintes: • `MAIN`: o evento principal que representa uma instrução SQL. • `AUX`: um evento complementar que contém detalhes adicionais. Por exemplo, uma instrução que renomeie um

Campo	Tipo de dados	Descrição
		objeto pode ter um evento com classe AUX que reflete o novo nome. Para localizar eventos MAIN e AUX correspondentes à mesma instrução, verifique se há eventos diferentes que tenham os mesmos valores para o campo pid e o campo statementId .
clientApplication	string	O aplicativo que o cliente usou para conectar-se conforme reportado pelo cliente. O cliente não precisa fornecer essas informações, então o valor pode ser nulo.

Campo	Tipo de dados	Descrição
command	string	A categoria geral da instrução SQL. Os valores para este campo dependem do valor de class. Os valores quando class é MAIN incluem o seguinte: • CONNECT: quando uma sessão de cliente está conectada. • QUERY: uma instrução SQL. Acompanhado por um ou mais eventos com um valor class de AUX. • DISCONNECT : quando uma sessão de cliente é desconectada. • FAILED_CONNECT : quando um cliente tenta se conectar, mas não consegue. • CHANGEUSER : uma alteração de estado que faz parte do protocolo de rede MySQL, não de uma instrução que você emite. Os valores quando class é AUX incluem o seguinte: • READ: uma instrução SELECT ou COPY quando a origem é uma relação ou uma consulta. • WRITE: uma instrução INSERT, UPDATE, DELETE, TRUNCATE ou COPY quando o destino é uma relação. • DROP: excluir um objeto. • CREATE: criar um objeto. • RENAME: renomear um objeto. • ALTER: alterar as propriedades de um objeto.

Campo	Tipo de dados	Descrição
`commandText`	string	Para eventos com um valor `class` de `MAIN`, esse campo representa a instrução SQL real passada pelo usuário. Este campo é usado para todos os tipos de registros, exceto para registros de conexão ou desconexão e, nesse caso, o valor é nulo. Para eventos com um valor `class` de `AUX`, esse campo contém informações complementares sobre os objetos envolvidos no evento. No Aurora MySQL, caracteres como aspas são precedidos por uma barra invertida, representando um caractere de escape. **Important** O texto SQL completo de cada instrução é visível no log de auditoria, incluindo quaisquer dados confidenciais. No entanto, as senhas do usuário do banco de dados serão editadas se o Aurora puder determiná-las pelo contexto, como na instrução SQL a seguir. `mysql> SET PASSWORD = 'my-password ';` **Note** Especifique uma senha diferente do prompt mostrado aqui como prática recomendada de segurança.
`databaseName`	string	O banco de dados ao qual o usuário se conectou.

Campo	Tipo de dados	Descrição
`dbProtocol`	string	o protocolo do banco de dados. Atualmente, esse valor é sempre `MySQL` para o Aurora MySQL.
`dbUserName`	string	O usuário do banco de dados com o qual o cliente foi autenticado.
`endTime` (somente registros de atividade de banco de dados da versão 1.2)	string	A hora em que a execução terminou para a instrução SQL. Ela é representada no formato de Tempo Universal Coordenado (UTC). Para calcular o tempo de execução da instrução SQL, use `endTime - startTime` . Veja também o campo `startTime` .
`errorMessage` (somente registros de atividade de banco de dados da versão 1.1)	string	Se houver algum erro, esse campo será preenchido com a mensagem de erro que seria gerada pelo servidor de banco de dados. O valor `errorMessage` é nulo para instruções normais que não resultaram em um erro. Um erro é definido como qualquer atividade que produziri a um evento de log de erro MySQL visível para o cliente em um nível de gravidade igual a `ERROR` ou superior. Para obter mais informações, consulte The Error Log no Guia de referência do MySQL. Por exemplo, erros de sintaxe e cancelamentos de consulta geram uma mensagem de erro. Erros internos do servidor MySQL, como erros de processo de checkpointer em segundo plano, não geram uma mensagem de erro. No entanto, os registros para esses eventos ainda são emitidos independentemente da configura ção do nível de gravidade do log. Isso impede que os invasores desativem o registro em log para tentar evitar a detecção. Veja também o campo `exitCode`.

Campo	Tipo de dados	Descrição
`exitCode`	int	Um valor usado para um registro de saída de sessão. Em uma saída limpa, isso contém o código de saída. Um código de saída não pode ser sempre obtido em alguns cenários de falha. Nesses casos, esse valor pode ser zero ou pode estar em branco.
`logTime`	string	Um time stamp como registrado no caminho do código de auditoria. Ela é representada no formato de Tempo Universal Coordenado (UTC). Para obter a maneira mais precisa de calcular a duração da instrução, consulte os campos `startTime` e `endTime`.
`netProtocol`	string	O protocolo de comunicação da rede Atualmente, esse valor é sempre `TCP` para o Aurora MySQL.
`objectName`	string	O nome do objeto do banco de dados se a instrução SQL estiver operando em um. Este campo será usado somente quando a instrução SQL operar em um objeto de banco de dados. Se a instrução SQL não estiver operando em um objeto, esse valor estará em branco. Para construir o nome totalmente qualificado do objeto, combine `databaseName` e `objectName` . Se a consulta envolver vários objetos, esse campo pode ser uma lista de nomes separada por vírgulas.
`objectType`	string	O tipo do objeto de banco de dados, como tabela, índice etc. Este campo será usado somente quando a instrução SQL operar em um objeto de banco de dados. Se a instrução SQL não estiver operando em um objeto, esse valor será nulo. Os valores válidos para o Aurora MySQL incluem o seguinte: • `INDEX` • `TABLE` • `UNKNOWN`

Campo	Tipo de dados	Descrição
`paramList`	string	Este campo não é usado para o Aurora MySQL e é sempre nulo.
`pid`	int	O ID do processo de backend que está alocado para atender à conexão do cliente. Quando o servidor de banco de dados é reiniciado, o `pid` é alterado e o contador do campo `statementId` recomeça a contagem.
`remoteHost`	string	O endereço IP ou o nome do host do cliente que emitiu a instrução SQL. No Aurora MySQL, qual deles é usado depende da configuração do parâmetro `skip_name_resolve` de banco de dados. O valor `localhost` indica atividade do usuário especial `rdsadmin`.
`remotePort`	string	O número da porta do cliente.
`rowCount`	int	O número de linhas retornado pela instrução SQL. Por exemplo, se uma instrução SELECT retornar 10 linhas, rowCount será 10. Para instruções INSERT ou UPDATE, rowCount é 0.
`serverHost`	string	O identificador da instância do servidor de banco de dados.
`serverType`	string	O tipo de servidor do banco de dados, por exemplo `MySQL`.
`serverVersion`	string	A versão do servidor do banco de dados. Atualmente, esse valor é sempre `MySQL 5.7.12` para o Aurora MySQL.
`serviceName`	string	O nome do serviço da Atualmente, esse valor é sempre `Amazon Aurora MySQL` para o Aurora MySQL.
`sessionId`	int	Um identificador de sessão pseudo-exclusivo.

Campo	Tipo de dados	Descrição
`startTime` (somente registros de atividade de banco de dados da versão 1.1)	string	A hora em que a execução começou para a instrução SQL. Ela é representada no formato de Tempo Universal Coordenado (UTC). Para calcular o tempo de execução da instrução SQL, use `endTime - startTime` . Veja também o campo `endTime`.
`statementId`	int	Um identificador para a instrução SQL do cliente. O contador incrementa com cada instrução SQL inserida pelo cliente. O contador é redefinido quando a instância de banco de dados é reiniciada.
`substatementId`	int	Um identificador para uma substituição SQL. Este valor é 1 para eventos com classe `MAIN` e 2 para eventos com classe `AUX`. Use o campo `statementId` para identificar todos os eventos gerados pela mesma instrução.
`transactionId` (somente registros de atividade de banco de dados da versão 1.2)	int	Um identificador para uma transação.
`type`	string	O tipo de evento. Os valores válidos são `record` ou `heartbeat` .

Processar um fluxo de atividade usando o SDK da AWS

É possível processar de maneira programática um fluxo de atividade usando o SDK da AWS. Veja a seguir exemplos de Java e Python em total funcionamento de como você pode processar o fluxo de dados do Kinesis.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
```

```
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {

    private static final String STREAM_NAME = "aws-rds-das-[cluster-external-
resource-id]";
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String DBC_RESOURCE_ID = "[cluster-external-resource-id]";
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
```

```
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
        String netProtocol;
        String objectName;
        String objectType;
        List<String> paramList;
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId;
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {
```

```
        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }

        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }
```

```
        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);

                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:dbc-id", DBC_RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
```

```
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
```

```
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
CREDENTIALS_PROVIDER, workerId);

kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
```

```
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # cluster-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-cluster-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
```

```
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)


def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:dbc-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
```

```
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

Exemplos de políticas do IAM para fluxos de atividades de banco de dados

Qualquer usuário com privilégios de função do AWS Identity and Access Management (IAM) para os fluxos de atividades de banco de dados pode criar, iniciar, interromper e modificar as configurações do fluxo de atividade para um cluster de banco de dados. Essas ações estão incluídas no log de auditoria do fluxo. Para realizar as práticas recomendadas de conformidade, recomendamos não fornecer esses privilégios a DBAs.

Você define o acesso aos fluxos de atividades de banco de dados usando políticas do IAM. Para obter mais informações sobre autenticação do Aurora, consulte Gerenciamento de identidade e acesso no Amazon Aurora. Para obter mais informações sobre como criar políticas do IAM, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM.

Example Política para permitir a configuração de fluxos de atividades de banco de dados

Para conceder aos usuários acesso refinado para modificar fluxos de atividade, use as chaves de contexto de operação específicas de serviço `rds:StartActivityStream` e `rds:StopActivityStream` em uma política do IAM. O exemplo de política do IAM a seguir permite que um usuário ou uma função configure fluxos de atividade.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureActivityStreams",
            "Effect": "Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Example Política para permitir o início de fluxos de atividades de banco de dados

O exemplo de política do IAM a seguir permite que um usuário ou uma função inicie fluxos de atividade.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example Política para permitir a interrupção de fluxos de atividades de banco de dados

O exemplo de política do IAM a seguir permite que um usuário ou uma função interrompa fluxos de atividade.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
```

```
}
```

Example Política para negar o início de fluxos de atividades de banco de dados

O exemplo de política do IAM a seguir impede que um usuário ou uma função inicie fluxos de atividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Política para interromper o início de fluxos de atividades de banco de dados

O exemplo de política do IAM a seguir impede que um usuário ou uma função interrompa fluxos de atividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Monitorar ameaças com o Amazon GuardDuty RDS Protection para Amazon Aurora

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Usando modelos de machine learning (ML) e recursos de detecção de anomalias e ameaças, o GuardDuty monitora continuamente diferentes fontes de log e atividades de runtime para identificar e priorizar possíveis riscos de segurança e atividades maliciosas no seu ambiente.

O GuardDuty RDS Protection analisa e traça o perfil de eventos de login em busca de possíveis ameaças de acesso aos seus bancos de dados do Amazon Aurora. Quando você ativa o RDS Protection, o GuardDuty consome eventos de login do RDS de seus bancos de dados do Aurora. O RDS Protection monitora esses eventos e traça o perfil de possíveis ameaças internas ou agentes externos.

Para ter mais informações sobre como habilitar a Proteção do GuardDuty RDS, consulte Proteção do GuardDuty RDS no Guia do usuário do Amazon GuardDuty.

Quando o RDS Protection detecta uma ameaça em potencial, como um padrão incomum em tentativas de login bem-sucedidas ou malsucedidas, o GuardDuty gera uma nova descoberta com detalhes sobre o banco de dados possivelmente comprometido. Você pode visualizar os detalhes da descoberta na seção de resumo da descoberta no console do Amazon GuardDuty. Os detalhes da descoberta variam de acordo com o tipo de descoberta. Os detalhes principais, o tipo de recurso e o perfil do recurso determinam o tipo de informação disponível para qualquer descoberta. Para ter mais informações sobre os detalhes comumente disponíveis para descobertas e os tipos de descoberta, consulte Encontrar detalhes e Tipos de descoberta da Proteção do GuardDuty RDS, respectivamente, no Guia do usuário do Amazon GuardDuty.

Você pode ativar ou desativar o recurso RDS Protection para qualquer Conta da AWS em qualquer Região da AWS onde esse recurso esteja disponível. Quando o RDS Protection não está ativado, o GuardDuty não detecta bancos de dados do Aurora possivelmente comprometidos nem fornece detalhes sobre o comprometimento.

Uma conta do GuardDuty existente pode habilitar a Proteção do RDS com um período de teste de 30 dias. Para uma nova conta do GuardDuty, a Proteção do RDS já está habilitada e incluída no período de teste gratuito de 30 dias. Para ter mais informações, consulte Custo da Proteção do GuardDuty RDS no Guia do usuário do Amazon GuardDuty.

Para saber em quais Regiões da AWS não é possível usar o GuardDuty com a Proteção do RDS, consulte Region-specific feature availability no Guia do usuário do Amazon GuardDuty.

Para ter informações sobre as versões do banco de dados do Aurora que o GuardDuty RDS Protection aceita, consulte Supported Amazon Aurora, Amazon RDS, and Aurora Limitless databases no Guia do usuário do Amazon GuardDuty.

Como trabalhar com o Amazon Aurora MySQL

O Amazon Aurora MySQL é um mecanismo de banco de dados relacional totalmente gerenciado e compatível com o MySQL, que combina a velocidade e a confiabilidade de bancos de dados comerciais de ponta com a simplicidade e a economia de bancos de dados de código aberto. O Aurora MySQL é uma substituição de entrada do MySQL que torna mais simples e econômico configurar, operar e escalar suas implantações do MySQL novas e existentes, permitindo que você se concentre em seus negócios e aplicações. O Amazon RDS fornece administração do Aurora ao processar as tarefas de rotina de bancos de dados, como provisionamento, aplicação de patches, backup, recuperação, detecção de falhas e reparo. O Amazon RDS também fornece as ferramentas de migração push para converter suas aplicações existentes do Amazon RDS para MySQL e do Amazon RDS para PostgreSQL para o Aurora MySQL.

Tópicos

- Visão geral do Amazon Aurora MySQL
- Segurança com o Amazon Aurora MySQL
- Atualizar aplicações para conexão com clusters de banco de dados do Aurora MySQL usando novos certificados TLS
- Usar a autenticação Kerberos para Aurora MySQL
- Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL
- Como gerenciar o Amazon Aurora MySQL
- Ajustar o Aurora MySQL
- Consulta paralela do Amazon Aurora MySQL
- Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL
- Replicação com o Amazon Aurora MySQL
- Usar o encaminhamento de gravação local em um cluster de banco de dados do Amazon Aurora MySQL
- Integração do Amazon Aurora MySQL com outros produtos da AWS
- Modo de laboratório do Amazon Aurora MySQL
- Melhores práticas do Amazon Aurora MySQL
- Solucionar problemas de desempenho do banco de dados do Amazon Aurora MySQL
- Referência do Amazon Aurora MySQL

- Atualizações do mecanismo de banco de dados Amazon Aurora MySQL

Visão geral do Amazon Aurora MySQL

As seções a seguir fornecem uma visão geral do Amazon Aurora MySQL.

Tópicos

- Melhorias de performance do Amazon Aurora MySQL
- Amazon Aurora MySQL e dados espaciais
- Aurora MySQL versão 3 compatível com o MySQL 8.0
- Aurora MySQL versão 2 compatível com o MySQL 5.7

Melhorias de performance do Amazon Aurora MySQL

O Amazon Aurora inclui melhorias de performance para dar suporte às diversas necessidades dos bancos de dados comerciais de alto nível.

Inserção rápida

A inserção rápida acelera as inserções paralelas classificadas pela chave primária e aplicam-se especificamente a instruções `LOAD DATA` e `INSERT INTO ... SELECT ...`. A inserção rápida armazena em cache a posição de um cursor em um percurso de índice ao executar a instrução. Isso evita percorrer de novo pelo índice desnecessariamente.

A inserção rápida só será habilitada para as tabelas regulares do InnoDB no Aurora MySQL versão 3.03.2 e posterior. Essa otimização não funciona para tabelas temporárias do InnoDB. Ela está desabilitada no Aurora MySQL versão 2 para todas as versões 2.11 e 2.12. A otimização da inserção rápida funcionará somente se a otimização do Adaptive Hash Index estiver desabilitada.

Você pode monitorar as seguintes métricas para determinar a eficácia da inserção rápida quanto ao cluster de banco de dados:

- `aurora_fast_insert_cache_hits`: um contador que é incrementado quando o cursor em cache é recuperado e verificado com sucesso.
- `aurora_fast_insert_cache_misses`: um contador que é incrementado quando o cursor em cache não é mais válido e o Aurora realiza um percurso de índice normal.

Você pode recuperar o valor atual das métricas de inserção rápida usando o seguinte comando:

```
mysql> show global status like 'Aurora_fast_insert%';
```

Você terá um resultado semelhante ao seguinte:

```
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| Aurora_fast_insert_cache_hits   | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+---------------------------------+-----------+
```

Amazon Aurora MySQL e dados espaciais

A lista a seguir resume os principais recursos espaciais do Aurora MySQL e explica como eles correspondem aos recursos espaciais no MySQL:

- O Aurora MySQL versão 2 oferece suporte aos mesmos tipos de dados espaciais e funções de relação espacial que o MySQL 5.7. Para obter mais informações sobre esses tipos de dados e funções, consulte o tópico sobre Tipos de dados espaciais e Funções de relação espacial, na documentação do MySQL 5.7.
- O Aurora MySQL versão 3 oferece suporte aos mesmos tipos de dados espaciais e funções de relação espacial que o MySQL 8.0. Para obter mais informações sobre esses tipos de dados e funções, consulte o tópico sobre Tipos de dados espaciais e Funções de relação espacial, na documentação do MySQL 8.0.
- O Aurora MySQL oferece suporte à indexação espacial em tabelas do InnoDB. A indexação espacial melhora a performance das consultas em conjuntos de dados grandes para consultas sobre dados espaciais. No MySQL, a indexação espacial para tabelas do InnoDB está disponível para MySQL 5.7 e 8.0.

  O Aurora MySQL utiliza uma estratégia de indexação espacial diferente do MySQL para alto desempenho com consultas espaciais. A implementação do índice espacial do Aurora usa uma curva de preenchimento de espaço em uma árvore B, que se destina a fornecer uma performance maior para as verificações de alcance espacial do que uma árvore R.

> **Note**
>
> No Aurora MySQL, uma transação em uma tabela com um índice espacial definido em uma coluna com um identificador de referência espacial (SRID) não pode ser inserida em uma área selecionada para atualização por outra transação.

Há suporte às seguintes instruções de linguagem de definição de dados (DDL) para criar índices em colunas que usam tipos de dados espaciais.

CREATE TABLE

É possível usar as palavras-chave `SPATIAL INDEX` em uma instrução `CREATE TABLE` para adicionar um índice espacial a uma coluna em uma nova tabela. Veja um exemplo a seguir.

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

É possível usar as palavras-chave `SPATIAL INDEX` em uma instrução `ALTER TABLE` para adicionar um índice espacial a uma coluna em uma tabela existente. Veja um exemplo a seguir.

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

É possível usar a palavra-chave `SPATIAL` em uma instrução `CREATE INDEX` para adicionar um índice espacial a uma coluna em uma tabela existente. Veja um exemplo a seguir.

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Aurora MySQL versão 3 compatível com o MySQL 8.0

É possível utilizar o Aurora MySQL versão 3 para obter os recursos mais recentes compatíveis com MySQL, aprimoramentos de performance e correções de bugs. A seguir, você pode saber mais sobre o Aurora MySQL versão 3 com compatibilidade com o MySQL 8.0. Você pode aprender a atualizar clusters e aplicações para o Aurora MySQL versão 3.

Alguns recursos do Aurora, como o Aurora Serverless v2, requer o Aurora MySQL versão 3.

Tópicos

- Recursos do MySQL 8.0 Community Edition
- Pré-requisito do Aurora MySQL versão 3 para o Aurora MySQL Serverless v2
- Notas de release do Aurora MySQL versão 3
- Novas otimizações de consultas paralelas
- Otimizações para reduzir o tempo de reinicialização do banco de dados
- Novo comportamento de tabela temporária no Aurora MySQL versão 3
- Comparar o Aurora MySQL versão 2 e o Aurora MySQL versão 3
- Comparar o Aurora MySQL versão 3 e o MySQL 8.0 Community Edition
- Fazer upgrade para o Aurora MySQL versão 3

Recursos do MySQL 8.0 Community Edition

A versão inicial do Aurora MySQL versão 3 é compatível com o MySQL 8.0.23 Community Edition. O MySQL 8.0 apresenta vários novos recursos, incluindo os seguintes:

- Compatibilidade com a linguagem de definição de dados (DDL) atômica. Para ter mais informações, consulte Compatibilidade com a Linguagem de definição de dados (DDL) atômica.
- Funções JSON. Para obter mais informações, consulte o tópico sobre Funções JSON, no Guia de referência do MySQL.
- Funções de janela. Para obter mais informações, consulte o tópico sobre Funções de janela, no Guia de referência do MySQL.
- Expressões de tabela comuns (CTEs), utilizando a cláusula `WITH`. Para obter mais informações, consulte WITH (Expressões de tabela comuns) no Guia de referência do MySQL.
- Otimização de cláusulas `ADD COLUMN` e `RENAME COLUMN` para a instrução `ALTER TABLE`. Essas otimizações são chamadas de “DDL instantânea”. O Aurora MySQL versão 3 tem compatibilidade com o recurso de DDL instantânea do MySQL edição da comunidade. O antigo recurso de DDL rápida do Aurora não é utilizado. Para obter informações sobre uso da DDL instantânea, consulte DDL instantânea (Aurora MySQL versão 3).
- Índices descendentes, funcionais e invisíveis. Para obter mais informações, consulte os tópicos sobre Índices invisíveis, Índices decrescentes e a Instrução CREATE INDEX, no Guia de referência do MySQL.

- Privilégios baseados em função controlados por instruções SQL. Para obter mais informações sobre as alterações feitas no modelo de privilégios, consulte Modelo de privilégios baseados em funções.
- Cláusulas `NOWAIT` e `SKIP LOCKED` com a instrução `SELECT ... FOR SHARE`. Essas cláusulas evitam esperar que outras transações liberem bloqueios de linhas. Para obter mais informações, consulte o tópico sobre Leituras de bloqueio, no Guia de referência do MySQL.
- Melhorias na replicação de log binário (binlog). Para obter detalhes referentes ao Aurora MySQL, consulte Replicação de log binário. Em particular, é possível realizar uma replicação filtrada. Para obter informações de uso sobre a replicação filtrada, consulte o tópico sobre Como servidores avaliam regras de filtragem de replicação, no Guia de referência do MySQL.
- Dicas. Algumas das dicas compatíveis com o MySQL 8.0 já foram transferidas ao Aurora MySQL versão 2. Para obter informações sobre como utilizar dicas com o Aurora MySQL, consulte Dicas do Aurora MySQL. Para acessar a lista completa de dicas no MySQL 8.0 edição da comunidade, consulte Dicas do otimizador, no Guia de referência do MySQL.

Para acessar a lista completa de recursos adicionados ao MySQL 8.0 edição da comunidade, consulte a postagem do blog Lista completa dos novos recursos no MySQL 8.0.

O Aurora MySQL versão 3 também inclui alterações em palavras-chave para linguagem inclusiva, transferidas do MySQL 8.0.26 edição da comunidade. Para conhecer os detalhes dessas alterações, consulte Alterações de linguagem inclusiva do Aurora MySQL versão 3.

Pré-requisito do Aurora MySQL versão 3 para o Aurora MySQL Serverless v2

O Aurora MySQL versão 3 é um pré-requisito para todas as instâncias de banco de dados em um cluster do Aurora MySQL Serverless v2. O Aurora MySQL Serverless v2 inclui suporte para instâncias de leitor em um cluster de banco de dados e outros recursos do Aurora que não estão disponíveis para o Aurora MySQL Serverless v1. Ele também tem uma escalabilidade mais rápida e granular do que o Aurora MySQL Serverless v1.

Notas de release do Aurora MySQL versão 3

Para acessar as notas de lançamento de todas as versões do Aurora MySQL versão 3, consulte Atualizações no mecanismo de banco de dados do Amazon Aurora MySQL versão 3 em Notas de lançamento do Aurora MySQL.

Novas otimizações de consultas paralelas

A otimização de consultas paralelas do Aurora agora é aplicável a mais operações SQL:

- A consulta paralela agora é aplicável a tabelas que contêm os tipos de dados `TEXT`,`BLOB`,`JSON`,`GEOMETRY`, `VARCHAR` e `CHAR` com mais de 768 bytes.
- A consulta paralela é capaz de otimizar consultas envolvendo tabelas particionadas.
- A consulta paralela é capaz de otimizar consultas envolvendo chamadas de função agregadas na lista de seleção e na cláusula `HAVING`.

Para obter mais informações sobre esses aprimoramentos, consulte Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3. Para obter informações gerais sobre a consulta paralela do Aurora, consulte Consulta paralela do Amazon Aurora MySQL.

Otimizações para reduzir o tempo de reinicialização do banco de dados

O cluster de banco de dados do Aurora MySQL deve estar altamente disponível durante interrupções planejadas e não planejadas.

Os administradores de banco de dados precisam realizar manutenção ocasional do banco de dados. Essa manutenção inclui aplicação de patches no banco de dados, atualizações, modificações nos parâmetros do banco de dados que exigem uma reinicialização manual, execução de um failover para reduzir o tempo necessário para, por exemplo, mudanças de classe e assim por diante. Essas ações planejadas necessitam de tempo de inatividade.

No entanto, o tempo de inatividade também pode ser causado por ações não planejadas, como um failover inesperado devido a uma falha de hardware subjacente ou à limitação de recursos do banco de dados. Todas essas ações planejadas e não planejadas resultam na reinicialização do banco de dados.

No Aurora MySQL versão 3.05 e superior, introduzimos otimizações que reduzem o tempo de reinicialização do banco de dados. Essas otimizações fornecem até 65% menos tempo de inatividade do que sem otimizações e menos interrupções nas workloads do banco de dados após uma reinicialização.

Durante a inicialização do banco de dados, muitos componentes da memória interna são inicializados. O maior deles é o pool de buffer do InnoDB, que no Aurora MySQL é 75% do tamanho da memória da instância por padrão. Nossos testes descobriram que o tempo de inicialização é proporcional ao tamanho do pool de buffer do InnoDB e, portanto, é escalável com o tamanho da

classe da instância de banco de dados. Durante essa fase de inicialização, o banco de dados não pode aceitar conexões, o que causa maior tempo de inatividade durante as reinicializações. A primeira fase da reinicialização rápida do Aurora MySQL otimiza a inicialização do pool de buffer, o que reduz o tempo de inicialização do banco de dados e, consequentemente, o tempo geral de reinicialização.

Para obter mais detalhes, consulte o blog Reduzir o tempo de inatividade com as otimizações do tempo de reinicialização do banco de dados Amazon Aurora MySQL.

Novo comportamento de tabela temporária no Aurora MySQL versão 3

O Aurora MySQL versão 3 processa as tabelas temporárias internas de forma diferente das versões anteriores do Aurora MySQL. Esse novo comportamento é herdado do MySQL 8.0 Community Edition. Existem dois tipos de tabelas temporárias que podem ser criadas com o Aurora MySQL versão 3:

- Tabelas temporárias internas (ou implícitas): criadas pelo mecanismo do Aurora MySQL para lidar com operações, como agregação de classificação, tabelas derivadas ou expressões de tabela comuns (CTEs).
- Tabelas temporárias criadas pelo usuário (ou explícitas): criadas pelo mecanismo do Aurora MySQL quando você usa a instrução `CREATE TEMPORARY TABLE`.

Há considerações adicionais sobre tabelas temporárias internas e criadas pelo usuário em instâncias de banco de dados do leitor do Aurora. Abordamos essas alterações nas seções a seguir.

Tópicos

- Mecanismo de armazenamento para tabelas temporárias internas (implícitas)
- Limitar o tamanho de tabelas temporárias internas na memória
- Mitigar problemas de volume em tabelas temporárias internas em réplicas do Aurora
- Otimizar o parâmetro temptable_max_mmap nas instâncias de banco de dados do Aurora MySQL
- Tabelas temporárias criadas pelo usuário (explícitas) em instâncias de banco de dados de leitor
- Mitigação e erros de criação de tabelas temporárias

Mecanismo de armazenamento para tabelas temporárias internas (implícitas)

Ao gerar conjuntos de resultados intermediários, o Aurora MySQL inicialmente tenta gravar em tabelas temporárias na memória. Esse procedimento pode não ser bem-sucedido devido a tipos de

dados incompatíveis ou limites configurados. Se esse for o caso, a tabela temporária será convertida em uma tabela temporária no disco, em vez de mantida na memória. Mais informações sobre isso podem ser encontradas em Uso de tabela temporária interna no MySQL na documentação do MySQL.

No Aurora MySQL versão 3, a maneira como as tabelas temporárias internas funcionam é diferente das versões anteriores do Aurora MySQL. Em vez de escolher entre os mecanismos de armazenamento InnoDB e MyISAM para essas tabelas temporárias, agora você escolhe entre os mecanismos de armazenamento `TempTable` e `MEMORY`.

Com o mecanismo de armazenamento `TempTable`, é possível fazer uma escolha adicional de como lidar com determinados dados. Os dados afetados transbordam o pool de memória que contém todas as tabelas temporárias internas da instância de banco de dados.

Essas opções podem influenciar a performance de consultas que geram altos volumes de dados temporários, por exemplo, ao realizar agregações como `GROUP BY` em tabelas grandes.

> **Tip**
>
> Se a sua workload incluir consultas que geram tabelas temporárias internas, confirme a performance da sua aplicação com essa alteração executando benchmarks e monitorando métricas de performance.
> Em alguns casos, a quantidade de dados temporários se encaixa no grupo de memória `TempTable` ou apenas transborda o grupo de memória em uma pequena quantidade. Nesses casos, convém utilizar a configuração `TempTable` para tabelas temporárias internas e arquivos mapeados para a memória a fim de conter quaisquer dados de estouro. Essa é a configuração padrão.

O mecanismo de armazenamento `TempTable` é o padrão. `TempTable` usa um grupo de memória comum para todas as tabelas temporárias que usam esse mecanismo, em vez de um limite máximo de memória por tabela. O tamanho desse grupo de memória é especificado pelo parâmetro temptable_max_ram. O padrão é 1 GiB em instâncias de banco de dados com 16 GiB ou mais de memória e 16 MB em instâncias de banco de dados com menos de 16 GiB de memória. O tamanho do grupo de memória influencia o consumo de memória em nível de sessão.

Em alguns casos, quando você usa o mecanismo de armazenamento `TempTable`, os dados temporários podem exceder o tamanho do grupo de memória. Nesse caso, o Aurora MySQL armazena os dados de transbordamento usando um mecanismo secundário.

É possível definir o parâmetro temptable_max_mmap para escolher se os dados transbordam para arquivos temporários mapeados pela memória ou para tabelas temporárias internas do InnoDB no disco. Os diferentes formatos de dados e critérios de transbordamento desses mecanismos de transbordamento podem afetar a performance das consultas. Isso ocorre devido à sua influência sobre a quantidade de dados gravados no disco e a demanda na taxa de transferência de armazenamento em disco.

O Aurora MySQL versão 3 armazena os dados de transbordamento da seguinte forma:

- Na instância de banco de dados de gravação, os dados que transbordam para tabelas temporárias internas do InnoDB ou para arquivos temporários mapeados em memória residem no armazenamento local na instância.
- Em instâncias de banco de dados de leitor, os dados de transbordamento sempre residem em arquivos temporários mapeados para a memória no armazenamento local.

  Instâncias somente leitura não podem armazenar dados no volume do cluster do Aurora.

Os parâmetros de configuração relacionados a tabelas temporárias internas são aplicáveis de maneira diferente às instâncias de gravador e de leitor no seu cluster:

- Em instâncias do leitor, o Aurora MySQL sempre utiliza o mecanismo de armazenamento `TempTable`.
- O tamanho de `temptable_max_mmap` é de 1 GiB por padrão, tanto para instâncias do gravador quanto do leitor, independentemente do tamanho da memória da instância de banco de dados. Você pode ajustar esse valor em instâncias do gravador e do leitor.
- A configuração de `temptable_max_mmap` para `0` desativa o uso de arquivos temporários mapeados na memória em instâncias do gravador.
- Você não pode definir `temptable_max_mmap` como `0` em instâncias do leitor.

> **Note**
>
> Não recomendamos utilizar o parâmetro temptable_use_mmap. Ele foi descontinuado, e há previsões de que o suporte será removido em uma versão futura do MySQL.

Limitar o tamanho de tabelas temporárias internas na memória

Conforme abordado em Mecanismo de armazenamento para tabelas temporárias internas (implícitas), é possível controlar recursos de tabelas temporárias globalmente usando as configurações temptable_max_ram e temptable_max_mmap.

Você também pode limitar o tamanho de qualquer tabela temporária interna individual na memória usando o parâmetro de banco de dados tmp_table_size. Esse limite tem como objetivo evitar que consultas individuais consumam uma quantidade excessiva de recursos globais de tabelas temporárias, o que pode afetar a performance de consultas simultâneas que exigem esses recursos.

O parâmetro `tmp_table_size` define o tamanho máximo das tabelas temporárias criadas pelo mecanismo de armazenamento `MEMORY` no Aurora MySQL versão 3.

No Aurora MySQL versão 3.04 e posterior, `tmp_table_size` também define o tamanho máximo das tabelas temporárias criadas pelo mecanismo de armazenamento `TempTable` quando o parâmetro de banco de dados `aurora_tmptable_enable_per_table_limit` é definido como `ON`. Esse comportamento, que está desabilitado (`OFF`) por padrão, é o mesmo que no Aurora MySQL versão 3.03 e versões anteriores.

- Quando `aurora_tmptable_enable_per_table_limit` está `OFF`, `tmp_table_size` não é considerado para tabelas temporárias internas na memória criadas pelo mecanismo de armazenamento `TempTable`.

  No entanto, o limite dos recursos `TempTable` globais ainda se aplica. O Aurora MySQL tem o seguinte comportamento quando o limite de recursos `TempTable` globais é atingido:

  - Instâncias de banco de dados de gravador: o Aurora MySQL converte automaticamente a tabela temporária na memória em uma tabela temporária em disco do InnoDB.
  - Instâncias de banco de dados de leitor: a consulta termina com um erro.

    ```
    ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
    ```

- Quando `aurora_tmptable_enable_per_table_limit` está `ON` e o limite de `tmp_table_size` é atingido, o Aurora MySQL tem o seguinte comportamento:
  - Instâncias de banco de dados de gravador: o Aurora MySQL converte automaticamente a tabela temporária na memória em uma tabela temporária em disco do InnoDB.
  - Instâncias de banco de dados de leitor: a consulta termina com um erro.

```
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
```

Tanto o limite de recursos `TempTable` globais quanto o limite por tabela se aplicam nesse caso.

> **Note**
>
> O parâmetro `aurora_tmptable_enable_per_table_limit` não tem efeito quando internal_tmp_mem_storage_engine está definido como `MEMORY`. Nesse caso, o tamanho máximo de uma tabela temporária na memória é definido pelo valor tmp_table_size ou max_heap_table_size, o que for menor.

Os exemplos a seguir mostram o comportamento do parâmetro `aurora_tmptable_enable_per_table_limit` para instâncias de banco de dados de gravador e leitor.

Example da instância de banco de dados de gravador com **`aurora_tmptable_enable_per_table_limit`** definido como **`OFF`**.

A tabela temporária na memória não é convertida em uma tabela temporária em disco do InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  0 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> show status like '%created_tmp_disk%';
```

```
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (13.99 sec)

mysql> show status like '%created_tmp_disk%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)
```

Example da instância de banco de dados de gravador com **aurora_tmptable_enable_per_table_limit** definido como **ON**.

A tabela temporária na memória não é convertida em uma tabela temporária em disco do InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+-----------------+----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+-----------------+----------------------------------------
+------------------+
```

```
|                  0 | 3.04.0           |                                          1 |
      16777216 |
+--------------------+------------------+--------------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
+---------+
| max(n)  |
+---------+
| 6000000 |
+---------+
1 row in set (4.10 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example da instância de banco de dados de leitor com **`aurora_tmptable_enable_per_table_limit`** definido como **`OFF`**.

A consulta termina sem um erro porque `tmp_table_size` não se aplica e o limite de recursos `TempTable` globais não foi atingido.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (14.05 sec)
```

Example da instância de banco de dados de leitor com **`aurora_tmptable_enable_per_table_limit`** definido como **`OFF`**.

Essa consulta atinge o limite global de recursos TempTable com `aurora_tmptable_enable_per_table_limit` definido como DESATIVADO. A consulta termina com um erro nas instâncias de leitor.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
```

```
+-------------------+-----------------+-----------------------------------------
+--------------------+---------------------+
|                   1 | 3.04.0          |                                       0 |
       1073741824 |          1073741824 |
+-------------------+-----------------+-----------------------------------------
+--------------------+---------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.01 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 120000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_1586_2' is full
```

Example da instância de banco de dados de leitor com **`aurora_tmptable_enable_per_table_limit`** definido como **`ON`**.

A consulta termina com um erro quando o limite `tmp_table_size` é atingido.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+-------------------+-----------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+-------------------+-----------------+-----------------------------------------
+------------------+
|                   1 | 3.04.0          |                                       1 |
      16777216 |
+-------------------+-----------------+-----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_8_2' is full
```

Mitigar problemas de volume em tabelas temporárias internas em réplicas do Aurora

Para evitar problemas de limitação de tamanho de tabelas temporárias, defina os parâmetros `temptable_max_ram` e `temptable_max_mmap` para um valor combinado que pode atender aos requisitos de sua workload.

Tenha cuidado ao definir o valor do parâmetro `temptable_max_ram`. Definir o valor muito alto reduz a memória disponível na instância do banco de dados, o que pode causar uma condição de falta de memória. Monitore a memória média livre na instância de banco de dados. Depois, determine um valor apropriado para `temptable_max_ram` para que você ainda tenha uma quantidade razoável de memória livre na instância. Para obter mais informações, consulte Problemas de memória liberável no Amazon Aurora.

Também é importante monitorar o tamanho do armazenamento local e o consumo de espaço em tabelas temporárias. É possível monitorar o armazenamento temporário disponível para uma instância de banco de dados específica com a métrica `FreeLocalStorage` do Amazon CloudWatch, descrita em Métricas do Amazon CloudWatch para o Amazon Aurora.

> **Note**
>
> Esse procedimento não funciona quando o parâmetro `aurora_tmptable_enable_per_table_limit` está definido como `0N`. Para obter mais informações, consulte Limitar o tamanho de tabelas temporárias internas na memória.

Example 1

Você sabe que suas tabelas temporárias aumentam até um tamanho cumulativo de 20 GiB. Você deseja definir tabelas temporárias na memória como 2 GiB e aumentar para um máximo de 20 GiB em disco.

Defina `temptable_max_ram` como **`2,147,483,648`** e `temptable_max_mmap` como **`21,474,836,480`**. Esses valores estão em bytes.

Essas configurações de parâmetros garantem que suas tabelas temporárias possam aumentar até um total cumulativo de 22 GiB.

Example 2

O tamanho da instância atual é 16xlarge ou maior. Você não sabe o tamanho total das tabelas temporárias que pode precisar. Você deseja ter a capacidade de usar até 4 GiB na memória e até o tamanho máximo de armazenamento disponível no disco.

Defina `temptable_max_ram` como **`4,294,967,296`** e `temptable_max_mmap` como **`1,099,511,627,776`**. Esses valores estão em bytes.

Aqui você está definindo `temptable_max_mmap` como 1 TiB, que é inferior ao armazenamento local máximo de 1,2 TiB em uma instância de banco de dados 16xlarge do Aurora.

Em um tamanho de instância menor, ajuste o valor de `temptable_max_mmap` para que ele não preencha o armazenamento local disponível. Por exemplo, uma instância 2xlarge tem apenas 160 GiB de armazenamento local disponível. Por isso, recomendamos definir o valor como menos de 160 GiB. Para obter mais informações sobre o armazenamento local disponível para tamanhos de instância de banco de dados, consulte Limites de armazenamento temporário para o Aurora MySQL.

Otimizar o parâmetro temptable_max_mmap nas instâncias de banco de dados do Aurora MySQL

O parâmetro `temptable_max_mmap` no Aurora MySQL controla a quantidade máxima de espaço em disco local que pode ser usada por arquivos mapeados na memória antes de transbordar para as tabelas temporárias do InnoDB em disco (em instâncias de banco de dados de gravador) ou causar um erro (em instâncias de banco de dados de leitor). Definir esse parâmetro de instância de banco de dados corretamente pode ajudar a otimizar o desempenho das instâncias de banco de dados.

Pré-requisitos

1. O Performance Schema deve estar habilitado. Você pode verificar isso executando o seguinte comando SQL:

   ```
   SELECT @@performance_schema;
   ```

   Um valor de saída de 1 indica que ele está habilitado.

2. Confirme se a instrumentação de memória de tabela temporária está habilitada. Você pode verificar isso executando o seguinte comando SQL:

   ```
   SELECT name, enabled FROM performance_schema.setup_instruments WHERE name LIKE
    '%memory%temptable%';
   ```

A coluna `enabled` mostra `YES` para as entradas relevantes de instrumentação de memória de tabela temporária.

Monitorar o uso de tabelas temporárias

Ao definir o valor inicial para `temptable_max_mmap`, recomendamos que você comece com 80% do tamanho do armazenamento local para a classe de instância de banco de dados que está usando. Isso garante que as tabelas temporárias tenham espaço em disco suficiente para operar com eficiência, ao mesmo tempo em que deixa espaço para outros usos do disco na instância.

Para encontrar o tamanho do armazenamento local para sua classe de instância de banco de dados, consulte Limites de armazenamento temporário para o Aurora MySQL.

Por exemplo, se você estiver usando a classe de instância de banco de dados db.r5.large, o tamanho do armazenamento local é de 32 GiB. Nesse caso, você definiria inicialmente o parâmetro `temptable_max_mmap` para 80% de 32 GiB, que é 25,6 GiB.

Depois de definir o valor inicial de `temptable_max_mmap`, execute a workload máxima nas instâncias do Aurora MySQL. Monitore o uso atual e máximo do disco de tabela temporária usando a seguinte consulta SQL:

```
SELECT event_name, current_count, current_alloc, current_avg_alloc, high_count,
 high_alloc, high_avg_alloc
FROM sys.memory_global_by_current_bytes WHERE event_name LIKE 'memory/temptable/%';
```

Essa consulta recupera as seguintes informações:

- `event_name`: o nome do evento de uso de memória ou disco de tabelas temporárias.
- `current_count`: o número atual de blocos de memória ou disco alocados de tabelas temporárias.
- `current_alloc`: a quantidade atual de memória ou disco alocada para tabelas temporárias.
- `current_avg_alloc`: o tamanho médio atual dos blocos de memória ou disco de tabelas temporárias.
- `high_count`: o maior número de blocos de memória ou disco alocados de tabelas temporárias.
- `high_alloc`: a maior quantidade de memória ou disco alocada para tabelas temporárias.
- `high_avg_alloc`: o maior tamanho médio dos blocos de memória ou disco de tabelas temporárias.

Se as consultas falharem com um erro Table is full usando essa configuração, isso indica que a workload requer mais espaço em disco para operações de tabelas temporárias. Nesse caso, considere aumentar o tamanho da instância de banco de dados para uma com mais espaço de armazenamento local.

Definir o valor ideal de **`temptable_max_mmap`**

Use o procedimento a seguir para monitorar e definir o tamanho correto para o parâmetro `temptable_max_mmap`.

1. Analise a saída da consulta anterior e identifique o pico de uso de disco para tabelas temporárias, conforme indicado pela coluna `high_alloc`.
2. Com base no pico de uso de disco de tabelas temporárias, ajuste o parâmetro `temptable_max_mmap` no grupo de parâmetros do banco de dados para as instâncias de banco de dados do Aurora MySQL.

   Defina o valor para ser um pouco maior do que o pico de uso de disco de tabelas temporárias para acomodar o crescimento futuro.
3. Aplique as alterações do grupo de parâmetros às instâncias de banco de dados.
4. Monitore o uso de disco de tabelas temporárias novamente durante o pico de workload para garantir que o novo valor de `temptable_max_mmap` seja apropriado.
5. Repita as etapas anteriores conforme necessário para ajustar o parâmetro `temptable_max_mmap`.

Tabelas temporárias criadas pelo usuário (explícitas) em instâncias de banco de dados de leitor

Você pode criar explicitamente tabelas temporárias usando uma palavra-chave `TEMPORARY` em sua instrução `CREATE TABLE`. As tabelas temporárias explícitas são compatíveis com a instância de banco de dados do gravador em um cluster de banco de dados do Aurora. Você também pode usar tabelas temporárias explícitas em instâncias de banco de dados do leitor, mas as tabelas não podem impor o uso do mecanismo de armazenamento InnoDB.

Para evitar erros ao criar tabelas temporárias explícitas em instâncias de banco de dados do leitor do Aurora MySQL, execute todas as instruções `CREATE TEMPORARY TABLE` em instâncias de banco de dados do leitor de uma destas maneiras:

- Não especifique a cláusula `ENGINE=InnoDB`.
- Não defina o modo SQL como `NO_ENGINE_SUBSTITUTION`.

Mitigação e erros de criação de tabelas temporárias

O erro retornado é diferente dependendo de você utilizar uma instrução `CREATE TEMPORARY TABLE` simples ou a variação `CREATE TEMPORARY TABLE AS SELECT`. Os exemplos a seguir mostram os diferentes tipos de erros.

Esse comportamento de tabela temporária apenas se aplica a instâncias somente leitura. Esse primeiro exemplo confirma que este é o tipo de instância à qual a sessão está conectada.

```
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
```

Para instruções `CREATE TEMPORARY TABLE` simples, a instrução falha quando o modo SQL `NO_ENGINE_SUBSTITUTION` está habilitado. Quando `NO_ENGINE_SUBSTITUTION` está desativado (padrão), a substituição do mecanismo apropriado é feita e a criação temporária da tabela é bem-sucedida.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql>  CREATE TEMPORARY TABLE tt2 (id int) ENGINE=InnoDB;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> CREATE TEMPORARY TABLE tt4 (id int) ENGINE=InnoDB;

mysql> SHOW CREATE TABLE tt4\G
*************************** 1. row ***************************
       Table: tt4
Create Table: CREATE TEMPORARY TABLE `tt4` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Para instruções `CREATE TEMPORARY TABLE AS SELECT`, a instrução falha quando o modo SQL `NO_ENGINE_SUBSTITUTION` está ativado. Quando `NO_ENGINE_SUBSTITUTION` está desativado (padrão), a substituição do mecanismo apropriado é feita e a criação temporária da tabela é bem-sucedida.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql> CREATE TEMPORARY TABLE tt1 ENGINE=InnoDB AS SELECT * FROM t1;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> show create table tt3;
+-------+-----------------------------------------------------------------+
| Table | Create Table                                                    |
+-------+-----------------------------------------------------------------+
| tt3   | CREATE TEMPORARY TABLE `tt3` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

Para obter mais informações sobre os aspectos de armazenamento e as implicações de performance de tabelas temporárias no Aurora MySQL versão 3, consulte a postagem do blog Usar o mecanismo de armazenamento TempTable no Amazon RDS para MySQL e Amazon Aurora MySQL.

Comparar o Aurora MySQL versão 2 e o Aurora MySQL versão 3

Use as seguintes informações para saber mais sobre as alterações a serem observadas ao fazer upgrade do cluster do Aurora MySQL versão 2 para a versão 3.

Tópicos

- Compatibilidade com a Linguagem de definição de dados (DDL) atômica
- Diferenças de recursos entre as versões 2 e 3 do Aurora MySQL
- Suporte a classes de instâncias
- Alterações de parâmetros do Aurora MySQL versão 3
- Variáveis de status
- Alterações de linguagem inclusiva do Aurora MySQL versão 3
- Valores de AUTO_INCREMENT
- Replicação de log binário

Compatibilidade com a Linguagem de definição de dados (DDL) atômica

Uma das maiores mudanças do MySQL 5.7 para o 8.0 é a introdução do Atomic Data Dictionary. Antes do MySQL 8.0, o dicionário de dados do MySQL usava uma abordagem baseada em arquivos para armazenar determinados metadados, como definições de tabela (.frm), gatilhos (.trg) e funções, à parte dos metadados do mecanismo de armazenamento (como os do InnoDB). Isso causava alguns problemas, incluindo o risco de as tabelas ficarem "órfãs" se algo inesperado acontecesse durante uma operação de DDL, fazendo com que os metadados baseados em arquivos e do mecanismo de armazenamento ficassem fora de sincronia.

Para corrigir isso, o MySQL 8.0 introduziu o Atomic Data Dictionary, que armazena todos os metadados em um conjunto de tabelas internas do InnoDB no esquema `mysql`. Essa nova arquitetura oferece uma forma transacional compatível com ACID de gerenciar metadados de banco de dados, resolvendo o problema de "DDL atômica" com a antiga abordagem baseada em arquivos. Consulte mais informações sobre o Atomic Data Dictionary em Removal of file-based metadata storage e em Atomic data definition statement support no MySQL Reference Manual.

Devido a essa mudança de arquitetura, você deve considerar o seguinte ao atualizar da versão 2 do Aurora MySQL para a versão 3:

- Os metadados baseados em arquivo da versão 2 devem ser migrados para as novas tabelas do dicionário de dados durante o processo de atualização para a versão 3. Dependendo do número de objetos de banco de dados que forem migrados, isso poderá levar algum tempo.
- As mudanças também incluíram algumas novas incompatibilidades que podem precisar ser resolvidas antes da atualização do MySQL 5.7 para 8.0. Por exemplo, a versão 8.0 introduziu algumas palavras-chave reservadas que podem entrar em conflito com os nomes de objeto de banco de dados existentes.

Para ajudar você a identificar essas incompatibilidades antes de atualizar o mecanismo, o Aurora MySQL executa uma série de verificações de compatibilidade de atualização (pré-verificações) antes de realizar a atualização do dicionário de dados para determinar se há algum objeto incompatível. Consulte mais informações sobre as pré-verificações em Verificações prévias de atualização da versão principal do Aurora MySQL.

Diferenças de recursos entre as versões 2 e 3 do Aurora MySQL

Os recursos a seguir no Amazon Aurora MySQL têm suporte no Aurora MySQL para o MySQL 5.7, mas não têm suporte atualmente no Aurora MySQL para o MySQL 8.0:

- Não é possível utilizar o Aurora MySQL versão 3 para clusters do Aurora Serverless v1. O Aurora MySQL versão 3 funciona com o Aurora Serverless v2.
- O modo de laboratório não é aplicável ao Aurora MySQL versão 3. Não há recursos de modo de laboratório no Aurora MySQL versão 3. A DDL instantâneo substitui o recurso de DDL on-line rápida que estava disponível no modo de laboratório. Para ver um exemplo, consulte DDL instantânea (Aurora MySQL versão 3).
- O cache de consulta foi removido do MySQL 8.0 edição da comunidade e também do Aurora MySQL versão 3.
- O Aurora MySQL versão 3 tem compatibilidade com o recurso de junção de hash do MySQL da comunidade. A implementação específica do Aurora de junções de hash no Aurora MySQL versão 2 não é utilizada. Para obter informações sobre como utilizar junções de hash com a consulta paralela do Aurora, consulte Habilitar a junção de hash para clusters de consulta paralela e Dicas do Aurora MySQL. Para obter informações gerais de uso sobre junções de hash, consulte Otimização de junções de hash, no Guia de referência do MySQL.
- O procedimento armazenado `mysql.lambda_async` que foi marcado como defasado no Aurora MySQL versão 2 foi removido na versão 3. Para a versão 3, use a função assíncrona `lambda_async` no lugar.
- O conjunto de caracteres padrão no Aurora MySQL versão 3 é `utf8mb4`. No Aurora MySQL versão 2, o conjunto de caracteres padrão era `latin1`. Para obter informações sobre esse conjunto de caracteres, consulte O conjunto de caracteres utf8mb4 (4-Byte UTF-8 Unicode Encoding), no Guia de referência do MySQL.

Determinados recursos do Aurora MySQL estão disponíveis para determinadas combinações de região da AWS e versão do mecanismo de banco de dados. Para obter detalhes, consulte Recursos compatíveis com o Amazon Aurora por Região da AWS e com o mecanismo de banco de dados do Aurora.

Suporte a classes de instâncias

O Aurora MySQL versão 3 oferece suporte a um conjunto diferente de classes de instâncias em comparação com o Aurora MySQL versão 2:

- Para instâncias maiores, é possível utilizar as classes de instâncias modernas, como `db.r5`, `db.r6g` e `db.x2g`.
- Para instâncias menores, é possível utilizar as classes de instâncias modernas, como `db.t3` e `db.t4g`.

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Uso de classes de instância T para desenvolvimento e testes.

As classes de instância a seguir do Aurora MySQL versão 2 não estão disponíveis para o Aurora MySQL versão 3:

- `db.r4`
- `db.r3`
- `db.t3.small`
- `db.t2`

Confira se há declarações da CLI em seus scripts de administração que criem instâncias de banco de dados do Aurora MySQL. Nomes de classes de instâncias de código fixo que não estejam disponíveis para o Aurora MySQL versão 3. Se necessário, modifique os nomes das classes de instância para nomes que são compatíveis no Aurora MySQL versão 3.

> **Tip**
>
> Para verificar as classes de instância que podem ser utilizadas para uma combinação específica de versão do Aurora MySQL e região da AWS, use o comando `describe-orderable-db-instance-options` AWS CLI.

Para obter detalhes completos sobre classes de instâncias do Aurora, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Alterações de parâmetros do Aurora MySQL versão 3

O Aurora MySQL versão 3 inclui novos parâmetros de configuração em nível de cluster e de instância. O Aurora MySQL versão 3 também remove alguns parâmetros anteriormente presentes no Aurora MySQL versão 2. Alguns nomes de parâmetros foram modificados como resultado da iniciativa de linguagem inclusiva. Para compatibilidade com versões anteriores, ainda é possível

recuperar valores de parâmetros utilizando os nomes antigos ou os novos. Porém, você deve utilizar os novos nomes para especificar valores de parâmetros em um grupo de parâmetros personalizado.

No Aurora MySQL versão 3, o valor do parâmetro `lower_case_table_names` é definido permanentemente no momento da criação do cluster. Se você utilizar um valor não padrão para essa opção, configure o grupo de parâmetros personalizado do Aurora MySQL versão 3 antes do upgrade. Em seguida, especifique o grupo de parâmetros durante a operação de criação de cluster ou restauração do snapshot.

> **Note**
>
> Com um banco de dados Aurora global baseado no Aurora MySQL, você não poderá executar uma atualização no local do Aurora MySQL versão 2 para a versão 3 se o parâmetro `lower_case_table_names` estiver ativado. Use o método de restauração de snapshot.

Na versão 3 do Aurora MySQL, os parâmetros `init_connect` e `read_only` não se aplicam aos usuários que têm o privilégio `CONNECTION_ADMIN`. Isso inclui o usuário principal do Aurora. Para ter mais informações, consulte Modelo de privilégios baseados em funções.

Para obter a lista completa dos parâmetros de cluster do Aurora MySQL, consulte Parâmetros no nível do cluster. A tabela engloba todos os parâmetros do Aurora MySQL versões 2 e 3. A tabela inclui observações mostrando quais parâmetros são novos no Aurora MySQL versão 3 ou foram removidos do Aurora MySQL versão 3.

Para obter a lista completa de parâmetros de instância do Aurora MySQL, consulte Parâmetros no nível da instância. A tabela engloba todos os parâmetros do Aurora MySQL versões 2 e 3. A tabela inclui observações mostrando quais parâmetros são novos no Aurora MySQL versão 3 e quais parâmetros foram removidos do Aurora MySQL versão 3. Ela também inclui observações indicando quais parâmetros eram modificáveis em versões anteriores, mas não no Aurora MySQL versão 3.

Para obter informações sobre nomes de parâmetros modificados, consulte Alterações de linguagem inclusiva do Aurora MySQL versão 3.

Variáveis de status

Para obter informações sobre variáveis de status não aplicáveis ao Aurora MySQL, consulte Variáveis de status do MySQL não se aplicam ao Aurora MySQL.

Alterações de linguagem inclusiva do Aurora MySQL versão 3

O Aurora MySQL versão 3 tem compatibilidade com a versão 8.0.23 do MySQL edição da comunidade. O Aurora MySQL versão 3 também inclui alterações do MySQL 8.0.26 referentes a palavras-chave e esquemas de sistema para linguagem inclusiva. Por exemplo, o uso do comando `SHOW REPLICA STATUS` agora é preferencial ao do comando `SHOW SLAVE STATUS`.

As seguintes métricas do Amazon CloudWatch têm novos nomes no Aurora MySQL versão 3.

No Aurora MySQL versão 3, somente os nomes de métricas novos estão disponíveis. Certifique-se de atualizar alarmes ou qualquer outra automação que dependa de nomes de métricas ao fazer upgrade para o Aurora MySQL versão 3.

Nome antigo	Novo nome
`ForwardingMasterDMLLatency`	`ForwardingWriterDMLLatency`
`ForwardingMasterOpenSessions`	`ForwardingWriterOpenSessions`
`AuroraDMLRejectedMasterFull`	`AuroraDMLRejectedWriterFull`
`ForwardingMasterDMLThroughput`	`ForwardingWriterDMLThroughput`

As seguintes variáveis de status têm novos nomes no Aurora MySQL versão 3.

Para compatibilidade, é possível utilizar qualquer um dos nomes na versão inicial do Aurora MySQL versão 3. Os nomes de variáveis de status antigos serão removidos em um release futuro.

Nome a ser removido	Nome novo ou preferencial
`Aurora_fwd_master_dml_stmt_duration`	`Aurora_fwd_writer_dml_stmt_duration`
`Aurora_fwd_master_dml_stmt_count`	`Aurora_fwd_writer_dml_stmt_count`

Nome a ser removido	Nome novo ou preferencial
`Aurora_fwd_master_select_stmt_duration`	`Aurora_fwd_writer_select_stmt_duration`
`Aurora_fwd_master_select_stmt_count`	`Aurora_fwd_writer_select_stmt_count`
`Aurora_fwd_master_errors_session_timeout`	`Aurora_fwd_writer_errors_session_timeout`
`Aurora_fwd_master_open_sessions`	`Aurora_fwd_writer_open_sessions`
`Aurora_fwd_master_errors_session_limit`	`Aurora_fwd_writer_errors_session_limit`
`Aurora_fwd_master_errors_rpc_timeout`	`Aurora_fwd_writer_errors_rpc_timeout`

Os seguintes parâmetros de configuração têm nomes novos no Aurora MySQL versão 3.

Para fins de compatibilidade, é possível verificar os valores dos parâmetros no cliente `mysql` utilizando qualquer nome no release inicial do Aurora MySQL versão 3. Você só pode usar os novos nomes ao modificar os valores em um grupo de parâmetros personalizado. Os nomes de parâmetros antigos serão removidos em um release futuro.

Nome a ser removido	Nome novo ou preferencial
`aurora_fwd_master_idle_timeout`	`aurora_fwd_writer_idle_timeout`
`aurora_fwd_master_max_connections_pct`	`aurora_fwd_writer_max_connections_pct`

Nome a ser removido	Nome novo ou preferencial
master_verify_checksum	source_verify_checksum
sync_master_info	sync_source_info
init_slave	init_replica
rpl_stop_slave_timeout	rpl_stop_replica_timeout
log_slow_slave_statements	log_slow_replica_statements
slave_max_allowed_packet	replica_max_allowed_packet
slave_compressed_protocol	replica_compressed_protocol
slave_exec_mode	replica_exec_mode
slave_type_conversions	replica_type_conversions
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_parallel_type	replica_parallel_type
slave_preserve_commit_order	replica_preserve_commit_order
log_slave_updates	log_replica_updates
slave_allow_batching	replica_allow_batching

Nome a ser removido	Nome novo ou preferencial
slave_load_tmpdir	replica_load_tmpdir
slave_net_timeout	replica_net_timeout
sql_slave_skip_counter	sql_replica_skip_counter
slave_skip_errors	replica_skip_errors
slave_checkpoint_period	replica_checkpoint_period
slave_checkpoint_group	replica_checkpoint_group
slave_transaction_retries	replica_transaction_retries
slave_parallel_workers	replica_parallel_workers
slave_pending_jobs_size_max	replica_pending_jobs_size_max
pseudo_slave_mode	pseudo_replica_mode

Os seguintes procedimentos armazenados têm novos nomes no Aurora MySQL versão 3.

Para compatibilidade, é possível utilizar qualquer um dos nomes na versão inicial do Aurora MySQL versão 3. Os nomes de procedimentos antigos serão removidos em um release futuro.

Nome a ser removido	Nome novo ou preferencial
mysql.rds_set_master_auto_position	mysql.rds_set_source_auto_position

Nome a ser removido	Nome novo ou preferencial
`mysql.rds_set_external_master`	`mysql.rds_set_external_source`
`mysql.rds_set_external_master_with_auto_position`	`mysql.rds_set_external_source_with_auto_position`
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`

Valores de AUTO_INCREMENT

No Aurora MySQL versão 3, o Aurora preserva o valor `AUTO_INCREMENT` para cada tabela ao reiniciar cada instância de banco de dados. No Aurora MySQL versão 2, o valor `AUTO_INCREMENT` não era preservado após uma reinicialização.

O valor `AUTO_INCREMENT` não é preservado quando você configura um novo cluster por meio da restauração com um snapshot, aplicação de uma recuperação em um ponto anterior no tempo e clonagem de um cluster. Nesses casos, o valor `AUTO_INCREMENT` é inicializado para o valor com base no maior valor de coluna na tabela na ocasião em que o snapshot foi criado. Esse comportamento é diferente daquele no RDS para MySQL 8.0, em que o valor `AUTO_INCREMENT` é preservado durante essas operações.

Replicação de log binário

No MySQL 8.0 edição da comunidade, a replicação de logs binários está habilitada por padrão. No Aurora MySQL versão 3, a replicação de logs binários está desabilitada por padrão.

> **Tip**
>
> Se os seus requisitos de alta disponibilidade forem atendidos pelos recursos de replicação integrados do Aurora, será possível deixar a replicação de logs binários desabilitada. Dessa forma, é possível evitar a sobrecarga de performance da replicação de logs binários. Você

também pode evitar os processos de monitoramento e solução de problemas associados que são necessários para gerenciar a replicação de logs binários.

O Aurora oferece suporte à replicação de logs binários de uma fonte compatível com MySQL 5.7 para o Aurora MySQL versão 3. O sistema de origem pode ser um cluster de banco de dados do Aurora MySQL, uma instância de banco de dados do RDS para MySQL ou uma instância do MySQL on-premises.

Assim como o MySQL edição da comunidade, o Aurora MySQL oferece suporte para replicação a partir de uma origem que executa uma versão específica para um destino que executa a mesma versão principal ou uma versão principal superior. Por exemplo, não há suporte para a replicação de um sistema compatível com MySQL 5.6 para o Aurora MySQL versão 3. Não há suporte para a replicação do Aurora MySQL versão 3 para um sistema compatível com o MySQL 5.7 ou MySQL 5.6. Para obter detalhes sobre como utilizar a replicação de logs binários, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

O Aurora MySQL versão 3 inclui melhorias na replicação de logs binários no MySQL 8.0 edição da comunidade, como a replicação filtrada. Para obter detalhes sobre as melhorias no MySQL 8.0 edição da comunidade, consulte Como servidores avaliam regras de filtragem de replicação, no Guia de referência do MySQL.

Compactação de transações para replicação de logs binários

Para obter informações de uso sobre a compactação de logs binários, consulte Compactação de transações de logs binários, no Guia de referência do MySQL.

As seguintes limitações aplicam-se à compactação de logs binários no Aurora MySQL versão 3:

- Transações cujos dados de logs binários sejam maiores que o tamanho máximo permitido do pacote não são compactadas. Isso se aplica independentemente de a configuração de compactação de logs binários do Aurora MySQL estar ativada ou não. Essas transações são replicadas sem serem compactadas.
- Se você utilizar um conector para captura de dados de alterações (CDC) que ainda não ofereça suporte ao MySQL 8.0, não será possível utilizar esse recurso. Recomendamos testar completamente todos os conectores de terceiros com a compactação de logs binários. Além disso, recomendamos fazer isso antes de ativar a compactação de binlog em sistemas que utilizam a replicação de binlog para CDC.

Comparar o Aurora MySQL versão 3 e o MySQL 8.0 Community Edition

Use as seguintes informações para saber mais sobre as alterações a serem observadas ao converter de um sistema compatível com MySQL 8.0 diferente para o Aurora MySQL versão 3.

Em geral, o Aurora MySQL versão 3 é compatível com o conjunto de recursos do MySQL 8.0.23 edição da comunidade. Alguns novos recursos do MySQL 8.0 edição da comunidade não se aplicam ao Aurora MySQL. Alguns desses recursos não são compatíveis com alguns aspectos do Aurora, como a arquitetura de armazenamento. Outros recursos não são necessários, pois o serviço de gerenciamento do Amazon RDS fornece funcionalidade equivalente. Os seguintes recursos no MySQL 8.0 edição da comunidade não têm suporte ou funcionam de maneira diferente no Aurora MySQL versão 3.

Para acessar as notas de lançamento de todas as versões do Aurora MySQL versão 3, consulte Atualizações no mecanismo de banco de dados do Amazon Aurora MySQL versão 3 em Notas de lançamento do Aurora MySQL.

Tópicos

- Recursos do MySQL 8.0 que não estão disponíveis no Aurora MySQL versão 3
- Modelo de privilégios baseados em funções
- Localização do ID do servidor do banco de dados
- Autenticação

Recursos do MySQL 8.0 que não estão disponíveis no Aurora MySQL versão 3

Os seguintes recursos no MySQL 8.0 edição da comunidade não estão disponíveis ou funcionam de maneira diferente no Aurora MySQL versão 3.

- Os grupos de recursos e as instruções SQL associadas não têm suporte no Aurora MySQL.
- O Aurora MySQL não é compatível com espaços de tabela undo definidos pelo usuário e instruções SQL associadas, como `CREATE UNDO TABLESPACE`, `ALTER UNDO TABLESPACE ... SET INACTIVE` e `DROP UNDO TABLESPACE`.
- O Aurora MySQL não é compatível com o truncamento de espaço de tabela undo para versões do Aurora MySQL anteriores a 3.06. No Aurora MySQL versão 3.06 e posterior, o truncamento automático do espaço de tabela undo é compatível.
- O plug-in de validação de senha não é aceito.

- Não é possível modificar as configurações de nenhum dos plug-ins do MySQL, incluindo o plug-in de validação de senha.
- Não há suporte para o plugin X.
- Não há suporte à replicação em várias origens.

Modelo de privilégios baseados em funções

Com o Aurora MySQL versão 3, não é possível modificar as tabelas no banco de dados `mysql` diretamente. Em particular, não é possível configurar usuários inserindo-os na tabela `mysql.user`. Em vez disso, use instruções SQL para conceder privilégios baseados em funções. Você também não pode criar outros tipos de objeto, como procedimentos armazenados no banco de dados do `mysql`. Você ainda pode consultar as tabelas `mysql`. Se você utilizar a replicação de logs binários, as alterações feitas diretamente nas tabelas `mysql` no cluster de origem não serão replicadas no cluster de destino.

Em alguns casos, sua aplicação pode utilizar atalhos para criar usuários ou outros objetos inserindo-os nas tabelas `mysql`. Em caso afirmativo, altere o código da aplicação para utilizar as instruções correspondentes, como `CREATE USER`. Se a aplicação criar procedimentos armazenados ou outros objetos no banco de dados do `mysql`, use um banco de dados diferente.

Para exportar metadados para usuários de banco de dados durante a migração de um banco de dados externo do MySQL, é possível usar um comando do MySQL Shell em vez de `mysqldump`. Consulte mais informações em Instance Dump Utility, Schema Dump Utility, and Table Dump Utility.

Para simplificar o gerenciamento de permissões para vários usuários ou aplicações, é possível utilizar a instrução `CREATE ROLE` para criar uma função que tenha um conjunto de permissões. Em seguida, você pode utilizar as instruções `GRANT` e `SET ROLE` e a função `current_role` para atribuir funções a usuários ou aplicações, alternar a função atual e verificar quais funções estão em vigor. Para obter mais informações sobre o sistema de permissões baseadas em funções no MySQL 8.0, consulte o tópico sobre como Usar funções, no Guia de referência do MySQL.

> **Important**
>
> É altamente recomendável não usar o usuário mestre diretamente nas aplicações. Em vez disso, siga as práticas recomendadas de usar um usuário do banco de dados criado com os privilégios mínimos obrigatórios para a aplicação.

Tópicos

- rds_superuser_role
- Usuário de verificação de privilégios para replicação de logs binários.
- Perfis para acessar outros serviços da AWS

rds_superuser_role

O Aurora MySQL versão 3 inclui uma função especial que tem todos os privilégios a seguir. Essa função se chama `rds_superuser_role`. O usuário administrativo principal de cada cluster já tem essa função concedida por padrão. A função `rds_superuser_role` inclui os seguintes privilégios para todos os objetos de banco de dados:

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CONNECTION_ADMIN`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `DROP ROLE`
- `EVENT`
- `EXECUTE`
- `FLUSH_OPTIMIZER_COSTS` (Aurora MySQL versão 3.09 e posterior)
- `FLUSH_STATUS` (Aurora MySQL versão 3.09 e posterior)
- `FLUSH_TABLES` (Aurora MySQL versão 3.09 e posterior)
- `FLUSH_USER_RESOURCES` (Aurora MySQL versão 3.09 e posterior)
- `INDEX`

- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW_ROUTINE (Aurora MySQL versão 3.04 e posterior)
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

A definição da função também inclui `WITH GRANT OPTION`, permitindo que um usuário administrativo a conceda para outros usuários. Em particular, o administrador deve conceder quaisquer privilégios necessários para executar a replicação de logs binários com o cluster do Aurora MySQL como destino.

> **Tip**
>
> Para visualizar os detalhes completos das permissões, insira as seguintes instruções.
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> SHOW GRANTS FOR name_of_administrative_user_for_your_cluster@'%';
> ```

Usuário de verificação de privilégios para replicação de logs binários.

O Aurora MySQL versão 3 inclui um usuário de verificação de privilégios para replicação de logs binários (binlog), `rdsrepladmin_priv_checks_user`. Além dos privilégios de `rds_superuser_role`, esse usuário tem o privilégio `replication_applier`.

Quando você ativa a replicação do log binário chamando o procedimento armazenado `mysql.rds_start_replication`, `rdsrepladmin_priv_checks_user` é criado.

O usuário `rdsrepladmin_priv_checks_user@localhost` é reservado. Não o modifique.

Perfis para acessar outros serviços da AWS

O Aurora MySQL versão 3 inclui perfis que podem ser usados para acessar outros serviços da AWS. É possível definir muitos desses perfis como uma alternativa à concessão de privilégios. Por exemplo, especifique `GRANT AWS_LAMBDA_ACCESS TO` *`user`* no lugar de `GRANT INVOKE LAMBDA ON *.* TO` *`user`*. Para conhecer os procedimentos para acessar outros serviços da AWS, consulte Integração do Amazon Aurora MySQL com outros produtos da AWS. O Aurora MySQL versão 3 inclui as seguintes funções relacionadas ao acesso a outros serviços da AWS:

- `AWS_LAMBDA_ACCESS`: alternativa ao privilégio `INVOKE LAMBDA`. Para ter mais informações, consulte Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL.
- `AWS_LOAD_S3_ACCESS`: alternativa ao privilégio `LOAD FROM S3`. Para ter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.
- `AWS_SELECT_S3_ACCESS`: alternativa ao privilégio `SELECT INTO S3`. Para ter mais informações, consulte Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3.
- `AWS_COMPREHEND_ACCESS`: alternativa ao privilégio `INVOKE COMPREHEND`. Para ter mais informações, consulte Conceder aos usuários de banco de dados acesso ao machine learning do Aurora.
- `AWS_SAGEMAKER_ACCESS`: alternativa ao privilégio `INVOKE SAGEMAKER`. Para ter mais informações, consulte Conceder aos usuários de banco de dados acesso ao machine learning do Aurora.
- `AWS_BEDROCK_ACCESS`: não há privilégio `INVOKE` análogo para o Amazon Bedrock. Para ter mais informações, consulte Conceder aos usuários de banco de dados acesso ao machine learning do Aurora.

Quando você concede acesso utilizando funções no Aurora MySQL versão 3, também ativa a função utilizando a instrução `SET ROLE role_name` ou `SET ROLE ALL`. O exemplo a seguir mostra como. Substitua o nome da função apropriado para `AWS_SELECT_S3_ACCESS`.

```
# Grant role to user.

mysql> GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the AWS_SELECT_S3_ACCESS role
 has not been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-----------------------------------------------------+
| CURRENT_ROLE()                                      |
+-----------------------------------------------------+
| `AWS_SELECT_S3_ACCESS`@`%`,`rds_superuser_role`@`%` |
+-----------------------------------------------------+
```

Localização do ID do servidor do banco de dados

O ID do servidor do banco de dados (`server_id`) é necessário para a replicação de log binário (binlog). A forma de encontrar o ID do servidor é diferente no Aurora MySQL e no MySQL Community.

No MySQL Community, o ID do servidor é um número, que você obtém usando a seguinte sintaxe enquanto está conectado ao servidor:

```
mysql> select @@server_id;

+-------------+
| @@server_id |
+-------------+
| 2           |
+-------------+
1 row in set (0.00 sec)
```

No Aurora MySQL, o ID do servidor é o ID da instância de banco de dados, que você obtém usando a seguinte sintaxe enquanto está conectado à instância de banco de dados:

```
mysql> select @@aurora_server_id;

+------------------------+
| @@aurora_server_id     |
+------------------------+
| mydbcluster-instance-2 |
+------------------------+
1 row in set (0.00 sec)
```

Para obter mais informações sobre replicação de log binário, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Autenticação

No MySQL 8.0 edição da comunidade, o plugin de autenticação padrão é `caching_sha2_password`. O Aurora MySQL versão 3 ainda utiliza o plugin `mysql_native_password`. Você não pode alterar a configuração `default_authentication_plugin`. No entanto, você pode criar usuários e alterar os usuários atuais, e as senhas individuais deles usarão o novo plug-in de autenticação. Veja um exemplo a seguir.

```
mysql> CREATE USER 'testnewsha'@'%' IDENTIFIED WITH caching_sha2_password BY
 'aNewShaPassword';
Query OK, 0 rows affected (0.74 sec)
```

Fazer upgrade para o Aurora MySQL versão 3

Para ter informações sobre como atualizar o banco de dados do Aurora MySQL versão 2 para a versão 3, consulte Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL.

Aurora MySQL versão 2 compatível com o MySQL 5.7

Este tópico descreve as diferenças entre o Aurora MySQL versão 2 e o MySQL 5.7 Community Edition.

> **Important**
>
> O Aurora MySQL versão 2 atingiu o fim do suporte padrão em 31 de outubro de 2024. Para ter mais informações, consulte Preparação para o fim do suporte padrão do Amazon Aurora edição compatível com o MySQL versão 2.

Recursos não compatíveis com o Aurora MySQL versão 2

Os recursos a seguir são compatíveis com o MySQL 5.7, mas não são compatíveis com o Aurora MySQL versão 2 atualmente:

- Instrução SQL `CREATE TABLESPACE`
- Plugin de replicação de grupo
- Maior tamanho de página
- Carregamento de grupo de buffers InnoDB na inicialização
- Plugin de analisador de texto completo do InnoDB
- Replicação em várias origens
- Redimensionamento on-line do grupo de buffers
- Plugin de validação de senha: é possível instalar o plugin, mas ele não é compatível. Não é possível personalizar o plugin.
- Plugins de regravação de consulta
- Filtragem de replicação
- Protocolo X

Para obter mais informações sobre esses recursos, consulte a documentação do MySQL 5.7.

Comportamento de espaço de tabela temporário no Aurora MySQL versão 2

No MySQL 5.7, o espaço de tabela temporário se estende automaticamente e aumenta de tamanho conforme necessário para acomodar tabelas temporárias em disco. Quando se excluem tabelas temporárias, o espaço liberado pode ser reutilizado para novas tabelas temporárias, mas o espaço de tabela temporário permanece no tamanho estendido e não diminui. O espaço de tabela temporário é excluído e recriado quando o mecanismo é reiniciado.

No Aurora MySQL versão 2, o seguinte comportamento se aplica:

- Para novos clusters de banco de dados do Aurora MySQL criados com a versão 2.10 e posterior, o espaço de tabela temporário é removido e recriado quando você reinicia o banco de dados. Isso permite que o recurso de redimensionamento dinâmico recupere o espaço de armazenamento.
- Para clusters de banco de dados do Aurora MySQL existentes que foram atualizados para:
  - Versão 2.10 ou posterior: o espaço de tabela temporário é removido e recriado quando você reinicia o banco de dados. Isso permite que o recurso de redimensionamento dinâmico recupere o espaço de armazenamento.
  - Versão 2.09: o espaço de tabela temporário não é removido quando você reinicia o banco de dados.

Você pode verificar o tamanho do espaço de tabela temporário no cluster de banco de dados do Aurora MySQL versão 2 usando a seguinte consulta:

```
SELECT
    FILE_NAME,
    TABLESPACE_NAME,
    ROUND((TOTAL_EXTENTS * EXTENT_SIZE) / 1024 / 1024 / 1024, 4) AS SIZE
FROM
    INFORMATION_SCHEMA.FILES
WHERE
    TABLESPACE_NAME = 'innodb_temporary';
```

Para obter mais informações, consulte The Temporary Tablespace na documentação do MySQL.

Mecanismo de armazenamento para tabelas temporárias em disco

O Aurora MySQL versão 2 usa mecanismos de armazenamento diferentes para tabelas temporárias internas em disco, dependendo do perfil da instância.

- Na instância gravadora, as tabelas temporárias em disco usam o mecanismo de armazenamento InnoDB por padrão. Elas são armazenados no espaço de tabela temporário no volume do cluster do Aurora.

  Você pode alterar esse comportamento na instância gravadora modificando o valor do parâmetro de banco de dados `internal_tmp_disk_storage_engine`. Para ter mais informações, consulte Parâmetros no nível da instância.
- Nas instâncias leitoras, as tabelas temporárias em disco usam o mecanismo de armazenamento MyISAM, que usa armazenamento local. Isso ocorre porque instâncias somente leitura não podem armazenar dados no volume do cluster do Aurora.

Segurança com o Amazon Aurora MySQL

A segurança do Amazon Aurora MySQL é gerenciada em três níveis:

- Para controlar quem pode realizar ações de gerenciamento do Amazon RDS em clusters de banco de dados e instâncias de banco de dados do Aurora MySQL, utilize o AWS Identity and Access Management (IAM). Ao se conectar à AWS usando credenciais do IAM, sua conta da AWSdeve ter políticas do IAM que concedam as permissões necessárias para executar operações de gerenciamento do Amazon RDS. Para obter mais informações, consulte . Gerenciamento de identidade e acesso no Amazon Aurora

  Se você estiver utilizando o IAM para acessar o console do Amazon RDS, primeiro faça login no Console de gerenciamento da AWS com suas credenciais de usuário do IAM. Depois, acesse o console do Amazon RDS em https://console.aws.amazon.com/rds/.

- Crie clusters de banco de dados do Aurora MySQL em uma nuvem pública virtual (VPC) com base no serviço Amazon VPC. Para controlar quais dispositivos e instâncias do Amazon EC2 podem abrir conexões com o endpoint e a porta da instância de banco de dados dos clusters de banco de dados do Aurora MySQL em uma VPC, use um grupo de segurança da VPC. É possível estabelecer essas conexões de endpoint e porta usando Transport Layer Security (TLS). Além disso, as regras de firewall em sua empresa podem controlar se dispositivos sendo executados nela podem abrir conexões em uma instância de banco de dados. Para ter mais informações sobre VPCs, consulte Amazon VPC e Amazon Aurora.

  A locação da VPC compatível depende da classe de instância de banco de dados usada pelos seus clusters de banco de dados do Aurora MySQL. Com a locação de VPC `default`, a VPC é executada em hardware compartilhado. Com a locação de VPC `dedicated`, a VPC é executada em uma instância de hardware dedicada. As classes de instâncias de banco de dados com performance intermitente apenas oferecem suporte para locação de VPC padrão. Essas classes de instâncias incluem db.t2, db.t3 e db.t4g. Todas as outras classes de instância de banco de dados do Aurora MySQL oferecem suporte para locação de VPC padrão e dedicada.

  > **Note**
  >
  > Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Uso de classes de instância T para desenvolvimento e testes.

Para ter mais informações sobre as classes da instância, consulte Classes de instâncias de banco de dados do Amazon Aurora. Para ter mais informações sobre a locação de VPC `default` e `dedicated`, consulte Instâncias dedicadas no Guia do usuário do Amazon Elastic Compute Cloud.

- Para autenticar o login e as permissões de um cluster de banco de dados do Amazon Aurora MySQL, siga uma das seguintes abordagens ou uma combinação delas.
  - Você pode seguir a mesma abordagem de uma instância autônoma do MySQL.

    Comandos, como `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` e `SET PASSWORD` funcionam exatamente como em bancos de dados on-premises, assim como modificando diretamente tabelas de esquema de banco de dados. Para obter mais informações, consulte Access Control and Account Management na documentação do MySQL.
  - Você também pode usar a autenticação de banco de dados do IAM.

    Com a autenticação de banco de dados do IAM, é possível autenticar seu cluster de banco de dados usando um usuário do IAM ou um perfil do IAM e um token de autenticação. Um token de autenticação é um valor exclusivo, gerado usando o processo de assinatura Signature Version 4. Ao usar a autenticação de banco de dados do IAM, você pode usar as mesmas credenciais para controlar o acesso aos seus recursos e bancos de dados da AWS. Para obter mais informações, consulte Autenticação do banco de dados do IAM.

> **Note**
>
> Para obter mais informações, consulte Segurança no Amazon Aurora.

Nas seções a seguir, consulte informações sobre permissões de usuário para conexões do Aurora MySQL e do TLS com clusters de banco de dados do Aurora MySQL.

Tópicos

- Privilégios de usuário mestre com Amazon Aurora MySQL.
- Conexões do TLS com clusters de banco de dados do Aurora MySQL

Privilégios de usuário mestre com Amazon Aurora MySQL.

Quando você cria uma instância de banco de dados do Amazon Aurora MySQL, o usuário principal apresenta os seguintes privilégios padrão listados em Privilégios da conta de usuário mestre.

Para fornecer serviços de gerenciamento para cada cluster de banco de dados, os usuários `admin` e `rdsadmin` são criados quando o cluster de banco de dados é criado. Tentar descartar, renomear ou alterar a senha, ou alterar os privilégios, para a conta `rdsadmin` resulta em um erro.

Nos clusters de banco de dados do Aurora MySQL versão 2, os usuários `admin` e `rdsadmin` são criados quando o cluster de banco de dados é criado. Nos clusters de banco de dados do Aurora MySQL versão 3, os usuários `admin`, `rdsadmin` e `rds_superuser_role` são criados.

> **Important**
>
> É altamente recomendável não usar o usuário mestre diretamente nas aplicações. Em vez disso, siga as práticas recomendadas de usar um usuário do banco de dados criado com os privilégios mínimos obrigatórios para a aplicação.

Visando o gerenciamento do cluster de banco de dados Aurora MySQL, os comandos `kill` e `kill_query` padrão foram restritos. Em vez disso, use os comandos do Amazon RDS `rds_kill` e `rds_kill_query` para encerrar sessões ou consultas de usuários em instâncias de banco de dados Aurora MySQL.

> **Note**
>
> A criptografia de uma instância de banco de dados e os snapshots não são compatíveis com a região China (Ningxia).

Conexões do TLS com clusters de banco de dados do Aurora MySQL

Os clusters de banco de dados do Amazon Aurora MySQL são compatíveis com conexões Transport Layer Security (TLS) de aplicações que usam o mesmo processo e a mesma chave pública que as instâncias de banco de dados do RDS para MySQL.

O Amazon RDS cria um certificado TLS e o instala na instância de banco de dados quando o Amazon RDS provisiona a instância. Esses certificados são assinados por uma autoridade de

certificado. O certificado TLS inclui o endpoint da instância de banco de dados como o nome comum (CN) do certificado TLS para se proteger contra ataques de falsificação. Por isso, você pode usar o endpoint do cluster de banco de dados para se conectar a um cluster de banco de dados usando TLS somente se o seu cliente for compatível com Subject Alternative Names (SAN). Caso contrário, será necessário usar o endpoint da instância de gravação.

Para obter informações sobre como baixar certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

Recomendamos o driver JDBC da AWS como um cliente que comporta SAN com TLS. Consulte mais informações sobre o driver JDBC da AWS e siga as instruções para usá-lo em Amazon Web Services (AWS) JDBC Driver GitHub repository.

Tópicos

- Exigir uma conexão TLS com um cluster de banco de dados do Aurora MySQL
- Versões do TLS para o Aurora MySQL
- Configurar conjuntos de criptografia para conexões com clusters de banco de dados do Aurora MySQL
- Criptografar conexões com um cluster de banco de dados do Aurora MySQL

Exigir uma conexão TLS com um cluster de banco de dados do Aurora MySQL

É possível exigir que todas as conexões de usuário com o cluster de banco de dados do Aurora MySQL usem TLS com o parâmetro de cluster de banco de dados `require_secure_transport`. Por padrão, o parâmetro `require_secure_transport` é definido como `OFF`. Você pode definir o parâmetro `require_secure_transport` como `ON` para exigir TLS para conexões com o cluster de banco de dados.

Você pode definir o valor do parâmetro `require_secure_transport` atualizando o grupo de parâmetros do seu cluster de banco de dados. Você não precisa reinicializar seu cluster de banco de dados para que a alteração entre em vigor. Para obter mais informações sobre parameter groups, consulte Grupos de parâmetros para Amazon Aurora.

> **Note**
>
> O parâmetro `require_secure_transport` está disponível para o Aurora MySQL versões 2 e 3. É possível definir esse parâmetro em um grupo de parâmetros de cluster de banco

de dados personalizado. O parâmetro não está disponível em grupos de parâmetros de instância de banco de dados.

Quando o parâmetro `require_secure_transport` é definido como `ON` para um cluster de banco de dados, um cliente de banco de dados pode se conectar a ele se puder estabelecer uma conexão criptografada. Caso contrário, uma mensagem de erro semelhante à seguinte é retornada para o cliente:

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

Versões do TLS para o Aurora MySQL

O Aurora MySQL é compatível com as versões 1.0, 1.1, 1.2 e 1.3 do Transport Layer Security (TLS). A partir do Aurora MySQL versão 3.04.0 e posterior, é possível usar o protocolo TLS 1.3 para proteger suas conexões. A tabela a seguir mostra o suporte a TLS para as versões do Aurora MySQL.

Aurora MySQL versão	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Padrão
Aurora MySQL versão 2	Suspenso	Suspenso	Compatível	Não compatível	Todas as versões do TLS compatíveis
Aurora MySQL versão 3 (inferior à 3.04.0)	Suspenso	Suspenso	Compatível	Não compatível	Todas as versões do TLS compatíveis
Aurora MySQL versão 3	Não suportado	Não suportado	Compatível	Compatível	Todas as versões do TLS compatíveis

Aurora MySQL versão	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Padrão
(3.04.0 e posterior)					

> **Important**
>
> Se você estiver usando grupos de parâmetros personalizados para clusters do Aurora MySQL com a versão 2 e versões anteriores à 3.04.0, recomendamos usar o TLS 1.2 porque o TLS 1.0 e 1.1 são menos seguros. A edição comunitária do MySQL 8.0.26 e do Aurora MySQL 3.03 e suas versões secundárias descontinuaram o suporte ao TLS versões 1.1 e 1.0.
>
> A edição comunitária do MySQL 8.0.28 e as versões compatíveis do Aurora MySQL 3.04.0 e posterior não são compatíveis com TLS 1.1 e TLS 1.0. Se você estiver usando a versão 3.04.0 ou posterior do Aurora MySQL, não defina o protocolo TLS como 1.0 e 1.1 em seu grupo de parâmetros personalizados.
>
> Para as versões 3.04.0 e posterior do Aurora MySQL, a configuração padrão é TLS 1.3 e TLS 1.2.

Você pode usar o parâmetro do cluster de banco de dados `tls_version` para indicar as versões de protocolo permitidas. Parâmetros de cliente semelhantes existem para a maioria das ferramentas de cliente ou drivers de banco de dados. Alguns clientes mais antigos podem não oferecer suporte às versões mais recentes do TLS. Por padrão, o cluster de banco de dados tenta usar a versão mais recente do protocolo TLS permitida pela configuração do servidor e do cliente.

Defina o parâmetro `tls_version` do cluster de banco de dados como um dos seguintes valores:

- `TLSv1.3`
- `TLSv1.2`
- `TLSv1.1`
- `TLSv1`

Você também pode definir o parâmetro `tls_version` como uma string de uma lista separada por vírgula. Se você quiser usar os protocolos TLS 1.2 e TLS 1.3, o parâmetro `tls_version` deverá incluir todos os protocolos, do mais baixo ao mais alto. Nesse caso, `tls_version` é definido como:

```
tls_version=TLSv1.2,TLSv1.3
```

Para obter informações sobre como modificar parâmetros em um grupo de parâmetros do cluster de banco de dados, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora. Se você usar a AWS CLI para modificar o parâmetro de cluster de banco de dados `tls_version`, o `ApplyMethod` deverá ser definido como `pending-reboot`. Quando o método de aplicação é `pending-reboot`, as alterações nos parâmetros são aplicadas após você interromper e reiniciar os clusters de banco de dados associados ao grupo de parâmetros.

Configurar conjuntos de criptografia para conexões com clusters de banco de dados do Aurora MySQL

Usando conjuntos de cifras configuráveis, você pode ter mais controle sobre a segurança de suas conexões de banco de dados. É possível especificar uma lista de pacotes de criptografias que você deseja permitir para proteger conexões TLS do cliente com o banco de dados. Com conjuntos de cifras configuráveis, você pode controlar a criptografia de conexão aceita pelo servidor de banco de dados. Esse procedimento ajuda a impedir o uso de criptografia insegura ou obsoleta.

Os pacotes de criptografias configuráveis são compatíveis com o Aurora MySQL versão 3 e o Aurora MySQL versão 2. Para especificar a lista de criptografias permitidas do TLS 1.2, do TLS 1.1 e do TLS 1.0 para criptografar conexões, modifique o parâmetro de cluster `ssl_cipher`. Defina o parâmetro `ssl_cipher` em um grupo de parâmetros do cluster usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Defina o parâmetro `ssl_cipher` para uma string de valores de criptografia separados por vírgulas para a sua versão do TLS. Para a aplicação cliente, você pode especificar a criptografia a ser usada para conexões criptografadas usando a opção `--ssl-cipher` ao se conectar ao banco de dados. Para obter mais informações sobre como se conectar ao seu banco de dados, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora MySQL.

A partir do Aurora MySQL versão 3.04.0 e posterior, é possível especificar pacotes de criptografia do TLS 1.3. Para especificar os pacotes de criptografia do TLS 1.3 permitidos, modifique o parâmetro `tls_ciphersuites` em seu grupo de parâmetros. O TLS 1.3 reduziu o número de pacotes de criptografia disponíveis devido a mudanças na convenção de nomenclatura que remove

o mecanismo de troca de chaves e o certificado usados. Defina o `tls_ciphersuites` como uma string de valores de criptografia separados por vírgulas para TLS 1.3.

A tabela a seguir mostra as criptografias compatíveis junto com o protocolo de criptografia TLS e as versões válidas do mecanismo do Aurora MySQL para cada criptografia.

Cifra	Protocolo de criptografia	Versões compatíveis do Aurora MySQL
`ECDHE-RSA-AES128-SHA`	TLS 1.0	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-RSA-AES128-SHA256`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-RSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-RSA-AES256-SHA`	TLS 1.0	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-RSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-RSA-CHACHA20-POLY1305`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-ECDSA-AES128-SHA`	TLS 1.0	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-ECDSA-AES256-SHA`	TLS 1.0	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-ECDSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
`ECDHE-ECDSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores

Cifra	Protocolo de criptografia	Versões compatíveis do Aurora MySQL
ECDHE-ECDSA-CHACHA20-POLY1305	TLS 1.2	3.04.0 e posteriores, 2.11.0 e posteriores
TLS_AES_128_GCM_SHA256	TLS 1.3	3.04.0 e posterior
TLS_AES_256_GCM_SHA384	TLS 1.3	3.04.0 e posterior
TLS_CHACHA20_POLY1305_SHA256	TLS 1.3	3.04.0 e posterior

Para obter informações sobre como modificar parâmetros em um grupo de parâmetros do cluster de banco de dados, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora. Se você usar a CLI para modificar o parâmetro de cluster de banco de dados `ssl_cipher`, o `ApplyMethod` deverá ser definido como `pending-reboot`. Quando o método de aplicação é `pending-reboot`, as alterações nos parâmetros são aplicadas após você interromper e reiniciar os clusters de banco de dados associados ao grupo de parâmetros.

Você também pode usar o comando da CLI describe-engine-default-cluster-parameters para determinar quais conjuntos de cifras são atualmente compatíveis com uma família de grupos de parâmetros específica. O exemplo a seguir mostra como obter os valores permitidos para o parâmetro de cluster `ssl_cipher` para o Aurora MySQL versão 2.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
mysql5.7

        ...some output truncated...
 {
        "ParameterName": "ssl_cipher",
        "ParameterValue": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "Description": "The list of permissible ciphers for connection encryption.",
        "Source": "system",
```

```
        "ApplyType": "static",
        "DataType": "list",
        "AllowedValues": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "IsModifiable": true,
        "SupportedEngineModes": [
            "provisioned"
        ]
    },
       ...some output truncated...
```

Para obter mais informações sobre criptografia, consulte a variável ssl_cipher na documentação do MySQL. Para obter mais informações sobre formatos de conjunto de cifras, consulte a documentação sobre o formato de lista de cifras do openssl e o formato de strings de cifras do openssl no site do OpenSSL.

Criptografar conexões com um cluster de banco de dados do Aurora MySQL

Para criptografar conexões usando o cliente `mysql` padrão, inicie o cliente mysql usando o parâmetro `--ssl-ca` para referenciar a chave pública, por exemplo:

Para MySQL 5.7 e 8.0:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-mode=VERIFY_IDENTITY
```

Para MySQL 5.6:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-verify-server-cert
```

Substitua *`full_path_to_CA_certificate`* pelo caminho completo do certificado da autoridade de certificação (CA). Para obter informações sobre como baixar certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

É possível exigir conexões TLS para determinadas contas de usuários. Por exemplo, é possível usar uma das seguintes declarações (dependendo da versão do MySQL) para exigir conexões SSL na conta do usuário `encrypted_user`.

Para MySQL 5.7 e 8.0:

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

Para MySQL 5.6:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

Ao usar um RDS Proxy, você se conecta ao endpoint do proxy e não ao endpoint do cluster usual. É possível tornar o SSL/TLS obrigatório ou opcional para conexões com o proxy, da mesma forma que para conexões feitas diretamente ao cluster de banco de dados do Aurora. Para receber informações sobre como usar o RDS Proxy, consulte Amazon RDS Proxypara Aurora.

> **Note**
>
> Para receber mais informações sobre as conexões TLS com MySQL, consulte a documentação do MySQL.

Atualizar aplicações para conexão com clusters de banco de dados do Aurora MySQL usando novos certificados TLS

Em 13 de janeiro de 2023, o Amazon RDS publicou novos certificados de autoridade de certificação (CA) para conexão com clusters de banco de dados do Aurora via Transport Layer Security (TLS). A seguir, você pode encontrar informações sobre como atualizar seus aplicativos para usar os novos certificados.

Este tópico pode ajudar você a determinar se alguma aplicação cliente usa TLS para se conectar a clusters de banco de dados. Em caso positivo, examine mais a fundo se esse aplicativo exige verificação de certificado para se conectar.

> **Note**
>
> Alguns aplicativos são configurados para se conectar a clusters de banco de dados MySQL do Aurora apenas quando podem verificar com sucesso o certificado no servidor.
> Para esses aplicativos, você deve atualizar os repositórios confiáveis de aplicativos cliente para incluir os novos certificados de CA.

Depois de atualizar seus certificados de CA nos armazenamentos confiáveis do aplicativo cliente, você pode fazer o rodízio dos certificados nos seus clusters de banco de dados. É altamente recomendável testar esses procedimentos em um ambiente de desenvolvimento ou teste antes de implementá-los em seus ambientes de produção.

Para obter mais informações sobre a mudança de certificados, consulte Alternar o certificado SSL/TLS. Para ter mais informações sobre como fazer download de certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados. Para obter informações sobre como usar o TLS com clusters de banco de dados do Aurora MySQL, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

Tópicos

- Determinar se alguma aplicação está se conectando ao cluster de banco de dados do Aurora MySQL usando TLS
- Determinar se um cliente requer verificação de certificado para se conectar
- Atualizar o armazenamento confiável de aplicações
- Exemplo de código Java para estabelecer conexões TLS

Determinar se alguma aplicação está se conectando ao cluster de banco de dados do Aurora MySQL usando TLS

Se você estiver usando o Aurora MySQL versão 2 (compatível com MySQL 5.7) e o Esquema de Performance estiver habilitado, execute a seguinte consulta para verificar se as conexões estão usando TLS. Para obter informações sobre como habilitar o Esquema de performance, consulte Performance Schema quick start na documentação do MySQL.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

Neste exemplo de saída, é possível ver que a sua própria sessão (`admin`) e uma aplicação conectada como `webapp1` estão usando TLS.

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

Determinar se um cliente requer verificação de certificado para se conectar

É possível verificar se os clientes JDBC e MySQL exigem verificação de certificado para se conectarem.

JDBC

O exemplo a seguir com o MySQL Connector/J 8.0 mostra uma maneira de verificar as propriedades da conexão JDBC de um aplicativo para determinar se conexões bem-sucedidas exigem um certificado válido. Para obter mais informações sobre todas as opções de conexão JDBC para MySQL, consulte Propriedades da configuração na documentação do MySQL.

Ao usar o MySQL Connector/J 8.0, uma conexão TLS exigirá verificação com base no certificado de CA de servidor se as propriedades da sua conexão tiverem `sslMode` definido como `VERIFY_CA` ou `VERIFY_IDENTITY`, como no exemplo a seguir.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> Se você usar o MySQL Java Connector v5.1.38 ou posterior, ou o MySQL Java Connector v8.0.9 ou posterior para se conectar a seus bancos de dados, mesmo que você não tenha configurado explicitamente suas aplicações para usar TLS ao se conectar a seus bancos de dados, esses drivers cliente usam TLS como padrão. Além disso, ao usar TLS, eles executam a verificação parcial do certificado e haverá falha na conexão se o certificado do servidor de banco de dados tiver expirado.

MySQL

Os exemplos a seguir com o cliente MySQL mostram duas maneiras de verificar a conexão MySQL de um script para determinar se conexões bem-sucedidas exigem um certificado válido. Para obter mais informações sobre todas as opções de conexão com o cliente MySQL, consulte Configuração no lado do cliente para conexões criptografadas na documentação do MySQL.

Ao usar o cliente MySQL 5.7 ou MySQL 8.0, uma conexão TLS vai exigir verificação com base no certificado de CA de servidor se, para a opção `--ssl-mode`, você especificar `VERIFY_CA` ou `VERIFY_IDENTITY`, como no exemplo a seguir.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

Ao usar o Cliente MySQL 5.6, uma conexão SSL exigirá verificação com base no certificado de CA de servidor se você especificar a opção `--ssl-verify-server-cert`, como no exemplo a seguir.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

Atualizar o armazenamento confiável de aplicações

Para obter informações sobre como atualizar o armazenamento confiável para aplicações MySQL, consulte Installing SSL certificates na documentação do MySQL.

> **Note**
>
> Ao atualizar o armazenamento confiável, é possível reter certificados mais antigos, além de adicionar os novos certificados.

Atualizar o armazenamento confiável de aplicações para JDBC

Você pode atualizar o armazenamento confiável para aplicações que usam JDBC para conexões TLS.

Para obter informações sobre como baixar o certificado raiz, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

Para obter exemplos de scripts que importam certificados, consulte Script de exemplo para importar certificados para o seu armazenamento confiável.

Se você estiver usando o driver JDBC mysql em um aplicativo, defina as seguintes propriedades nesse aplicativo.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

> **Note**
>
> Especifique uma senha diferente do prompt mostrado aqui como prática recomendada de segurança.

Ao iniciar o aplicativo, defina as seguintes propriedades.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

Exemplo de código Java para estabelecer conexões TLS

O exemplo de código a seguir mostra como configurar a conexão SSL que valida o certificado de servidor usando JDBC.

```
public class MySQLSSLTest {

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void test(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");


        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.setProperty("sslMode", "VERIFY_IDENTITY");
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);

        Connection connection = DriverManager.getConnection("jdbc:mysql://jagdeeps-ssl-
test.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
}
```

Important

Depois de determinar que suas conexões com o banco de dados usam TLS e atualizar o armazenamento confiável de aplicações, você poderá atualizar o banco de dados para usar os certificados rds-ca-rsa2048-g1. Para obter instruções, consulte a etapa 3 em Atualizar o certificado CA modificando a instância de banco de dados.

Usar a autenticação Kerberos para Aurora MySQL

É possível usar a autenticação Kerberos para autenticar usuários quando se conectam ao seu cluster de banco de dados do Aurora MySQL. Para fazer isso, configure seu cluster de banco de dados para usar o AWS Directory Service for Microsoft Active Directory para autenticação Kerberos. O AWS Directory Service for Microsoft Active Directory também é chamado de AWS Managed Microsoft AD. É um recurso disponível com o Directory Service. Para saber mais, consulte What is Directory Service? (O que é o ?) no Guia de administração do AWS Directory Service.

Para iniciar, crie um diretório AWS Managed Microsoft AD para armazenar credenciais de usuário. Depois, forneça o domínio do Active Directory e outras informações ao seu cluster de banco de dados do Aurora MySQL. Quando os usuários são autenticados com o cluster de banco de dados do Aurora MySQL, as solicitações de autenticação são encaminhadas ao diretório AWS Managed Microsoft AD.

Manter todas as suas credenciais no mesmo diretório pode economizar tempo e esforço. Com essa abordagem, há um lugar centralizado para armazenar e gerenciar credenciais para vários clusters de banco de dados. O uso de um diretório também pode melhorar o perfil de segurança geral.

Além disso, é possível acessar credenciais de seu próprio Microsoft Active Directory on-premises. Para fazer isso, crie uma relação de domínio confiável para que o diretório AWS Managed Microsoft AD confie no Microsoft Active Directory on-premises. Dessa forma, seus usuários podem acessar os clusters de banco de dados do Aurora MySQL com a mesma experiência de autenticação única (SSO) do Windows de quando eles acessam workloads em sua rede on-premises.

Um banco de dados pode usar autenticação Kerberos, do AWS Identity and Access Management (IAM) ou ambas. No entanto, como a autenticação Kerberos e o IAM oferecem métodos de autenticação diferentes, um usuário específico pode fazer login somente em um banco de dados usando um ou outro método de autenticação, mas não em ambos. Para ter mais informações sobre a autenticação do IAM, consulte Autenticação do banco de dados do IAM.

Sumário

- Visão geral da autenticação Kerberos para clusters do Aurora MySQL
- Limitações da autenticação Kerberos para o Aurora MySQL
- Configurar a autenticação Kerberos para clusters de banco de dados do Aurora MySQL
  - Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD
  - Etapa 2: (Opcional) Criar uma confiança para um Active Directory on-premises

- Etapa 3: Criar um perfil do IAM para ser usada pelo Amazon Aurora
- Etapa 4: Criar e configurar usuários
- Etapa 5: Criar ou modificar um cluster de banco de dados do Aurora MySQL
- Etapa 6: Criar usuários do Aurora MySQL que usam a autenticação Kerberos
  - Modificar um login existente do Aurora MySQL
- Etapa 7: Configurar um cliente MySQL
- Etapa 8: (Opcional) Configurar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas
- Conectar-se ao Aurora MySQL com a autenticação Kerberos
  - Utilizar o login do Kerberos do Aurora MySQL para se conectar ao cluster de banco de dados
  - Autenticação Kerberos com bancos de dados globais Aurora
  - Migração do RDS para MySQL para o Aurora MySQL
  - Evitar o armazenamento em cache de tíquetes
  - Registro em log da autenticação Kerberos
- Gerenciar um cluster de banco de dados em um domínio
  - Compreensão da associação de domínio

Visão geral da autenticação Kerberos para clusters do Aurora MySQL

Para configurar a autenticação Kerberos para um cluster de banco de dados do Aurora MySQL, conclua as etapas gerais a seguir. Essas etapas serão descritas em mais detalhes mais adiante.

1. Use AWS Managed Microsoft AD para criar um diretório do AWS Managed Microsoft AD. É possível usar o Console de gerenciamento da AWS, o AWS CLI ou o Directory Service para criar o diretório. Para obter instruções, consulte Criar seu diretório AWS Managed Microsoft AD no Guia de administração do AWS Directory Service.
2. Crie uma função do AWS Identity and Access Management (IAM) que use a politica gerenciada `AmazonRDSDirectoryServiceAccess` do IAM. O perfil permite ao Amazon Aurora fazer chamadas para seu diretório.

   Para o perfil permitir o acesso, o endpoint do AWS Security Token Service (AWS STS) deve estar ativado na Região da AWS de sua conta da AWS. Os endpoints do AWS STS são ativados por padrão em todas as Regiões da AWS, e você pode usá-los sem precisar tomar medidas

adicionais. Para ter mais informações, consulte Ativar e desativar o AWS STS em uma Região da AWS no Guia do usuário do IAM.

3. Crie e configure usuários no diretório AWS Managed Microsoft AD usando as ferramentas do Microsoft Active Directory. Para ter mais informações sobre como criar usuários em seu Active Directory, consulte Gerenciar usuários e grupos no Microsoft AD gerenciado pela AWS no Guia de administração doAWS Directory Service.
4. Crie ou modifique um cluster de banco de dados do Aurora MySQL. Se você usar a CLI ou a API do RDS na solicitação de criação, especifique um identificador de domínio com o parâmetro `Domain`. Use o identificador `d-*` que foi gerado quando você criou o diretório e o nome do perfil do IAM que você criou.

   Se você modificar um cluster de banco de dados do Aurora MySQL existente para usar a autenticação Kerberos, defina os parâmetros de domínio e perfil do IAM para o cluster de banco de dados. Localize o cluster de banco de dados na mesma VPC que o diretório de domínio.
5. Use as credenciais de usuário primário do Amazon RDS para conectar-se ao cluster de banco de dados do Aurora MySQL. Crie o usuário do banco de dados no Aurora MySQL usando as instruções em Etapa 6: Criar usuários do Aurora MySQL que usam a autenticação Kerberos.

   Os usuários que você cria dessa maneira podem fazer login no cluster de banco de dados do Aurora MySQL usando a autenticação Kerberos. Para ter mais informações, consulte Conectar-se ao Aurora MySQL com a autenticação Kerberos.

Para obter a autenticação Kerberos com um Microsoft Active Directory on-premises ou auto-hospedado, crie uma confiança de floresta. Confiança de floresta é uma relação de confiança entre dois grupos de domínios. A confiança pode ser unidirecional ou bidirecional. Para ter mais informações sobre como configurar confianças de floresta usando o Directory Service, consulte Quando criar um relacionamento de confiança no Guia de administração do AWS Directory Service.

Limitações da autenticação Kerberos para o Aurora MySQL

As seguintes limitações se aplicam à autenticação Kerberos para Aurora MySQL:

- A autenticação Kerberos é compatível com o Aurora MySQL versão 3.03 e posterior.

  Para ter informações sobre a Região da AWS, consulte Autenticação do Kerberos com o Aurora MySQL.

- Para usar a autenticação Kerberos com o Aurora MySQL, seu cliente ou conector MySQL deve usar a versão 8.0.26 ou posterior em plataformas Unix, 8.0.27 ou posterior no Windows. Caso contrário, o plug-in `authentication_kerberos_client` do lado do cliente não estará disponível e você não poderá se autenticar.
- Só o AWS Managed Microsoft AD é compatível com o Aurora MySQL. Contudo, você pode associar os clusters de banco de dados do Aurora MySQL a domínios gerenciados do Microsoft AD compartilhados de propriedade de contas diferentes na mesma Região da AWS.

  Também é possível usar seu próprio Active Directory on-premises. Para ter mais informações, consulte Etapa 2: (Opcional) Criar uma confiança para um Active Directory on-premises.
- Ao usar o Kerberos para autenticar um usuário conectado ao cluster do Aurora MySQL usando clientes MySQL ou drivers no sistema operacional Windows, a capitalização dos caracteres do nome de usuário do banco de dados deve corresponder à capitalização do usuário no Active Directory. Por exemplo, se o usuário no Active Directory for exibido como `Admin`, o nome de usuário do banco de dados deverá ser `Admin`.

  No entanto, agora você pode usar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas com o plug-in `authentication_kerberos`. Para ter mais informações, consulte Etapa 8: (Opcional) Configurar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas.
- Você deve reinicializar as instâncias de banco de dados do leitor depois de ativar o recurso para instalar o plug-in `authentication_kerberos`.
- A replicação para instâncias de banco de dados que não são compatíveis com o plug-in `authentication_kerberos` pode causar falha na replicação.
- Para que os bancos de dados globais do Aurora usem a autenticação Kerberos, você deve configurá-la para cada cluster de banco de dados no banco de dados global.
- O nome do domínio deve ter menos de 62 caracteres.
- Não modifique a porta do cluster de banco de dados depois de ativar a autenticação Kerberos. Se você modificar a porta, a autenticação Kerberos deixará de funcionar.

Configurar a autenticação Kerberos para clusters de banco de dados do Aurora MySQL

Use o AWS Managed Microsoft AD para configurar a autenticação Kerberos para um cluster de banco de dados do Aurora MySQL. Para configurar a autenticação Kerberos, execute as etapas a seguir.

Tópicos

- Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD
- Etapa 2: (Opcional) Criar uma confiança para um Active Directory on-premises
- Etapa 3: Criar um perfil do IAM para ser usada pelo Amazon Aurora
- Etapa 4: Criar e configurar usuários
- Etapa 5: Criar ou modificar um cluster de banco de dados do Aurora MySQL
- Etapa 6: Criar usuários do Aurora MySQL que usam a autenticação Kerberos
- Etapa 7: Configurar um cliente MySQL
- Etapa 8: (Opcional) Configurar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas

Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD

O Directory Service cria um Active Directory totalmente gerenciado na Nuvem AWS. Ao criar um diretório do AWS Managed Microsoft AD, o Directory Service cria dois controladores de domínio e servidores do Domain Name System (DNS) em seu nome. Os servidores do diretório são criados em sub-redes diferentes em uma VPC. Essa redundância ajuda a garantir que o diretório permaneça acessível mesmo se ocorrer uma falha.

Ao criar um diretório do AWS Managed Microsoft AD, o Directory Service executa as seguintes tarefas em seu nome:

- Configura um Active Directory dentro da VPC.
- Cria uma conta de administrador do diretório com o nome de usuário `Admin` e a senha especificada. Use essa conta para gerenciar seu diretório.

> **Note**
>
> Salve essa senha. O Directory Service não a armazena. Você pode redefini-la, mas não recuperá-la.

- Cria um grupo de segurança para os controladores do diretório.

Quando você inicia o AWS Managed Microsoft AD, o AWS cria uma Unidade organizacional (UO) que contém todos os objetos do diretório. Essa UO tem o nome de NetBIOS que você digitou ao criar o diretório. Ele está localizado na raiz do domínio, que pertence à AWS e é gerenciado por ela.

A conta `Admin`, que foi criada com o diretório AWS Managed Microsoft AD, tem permissões para as atividades administrativas mais comuns da UO, como:

- Criar, atualizar ou excluir usuários
- Adicionar recursos ao domínio, como servidores de arquivos ou de impressão, e atribuir permissões para esses recursos aos usuários na UO
- Criar OUs adicionais e contêineres
- Delegar autoridade
- Restaurar objetos excluídos da Lixeira do Active Directory
- Execute os módulos AD e DNS do Windows PowerShell no Active Directory Web Service

A conta `Admin` também possui direitos para executar as seguintes atividades de domínio:

- Gerenciar configurações de DNS (adicionar, remover ou atualizar registros, zonas e encaminhadores)
- Visualizar logs de eventos de DNS
- Visualizar logs de eventos de segurança

Como criar um diretório com AWS Managed Microsoft AD

1. Faça login no Console de gerenciamento da AWS e abra o console do Directory Service em https://console.aws.amazon.com/directoryservicev2/.
2. No painel de navegação, escolha Directories (Diretórios) e escolha Set up directory (Configurar diretório).

3. Escolha AWS Managed Microsoft AD. O AWS Managed Microsoft AD é a única opção que você pode usar atualmente com o Amazon RDS.
4. Insira as seguintes informações:

   Nome do DNS do diretório

   O nome completo do diretório, como **`corp.example.com`**.

   Nome de NetBIOS do diretório

   O nome curto do diretório, como **`CORP`**.

   Descrição do diretório

   (Opcional) Uma descrição do diretório.

   Senha do Admin

   A senha do administrador do diretório. O processo de criação do diretório cria uma conta de administrador com o nome de usuário Admin e essa senha.

   A senha do administrador do diretório e não pode incluir a palavra "admin". A senha diferencia letras maiúsculas de minúsculas e deve ter entre 8 e 64 caracteres. Ela também precisa conter pelo menos um caractere de três das quatro categorias a seguir:

   - Letras minúsculas (a–z)
   - Letras maiúsculas (A–Z)
   - Números (0–9)
   - Caracteres não alfanuméricos (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   Confirmar senha

   A senha do administrador reinserida.
5. Escolha Próximo.
6. Insira as seguintes informações na seção Networking (Rede) e escolha Next (Próximo):

   VPC

   A VPC do diretório. Crie o cluster de banco de dados do Aurora MySQL nesta mesma VPC.

Sub-redes

Sub-redes para os servidores do diretório. As duas sub-redes deve estar em diferentes zonas de disponibilidade.

7. Revise as informações do diretório e faça as alterações necessárias. Quando as informações estiverem corretas, selecione Create directory (Criar diretório).
Página de detalhes do diretório durante a criação

A criação do diretório leva alguns minutos. Depois que o diretório tiver sido criado com sucesso, o valor de Status muda para Active (Ativo).

Para ver informações sobre o diretório, selecione o nome do diretório na listagem de diretórios. Anote o valor do ID do diretório porque você precisará desse valor ao criar ou modificar seu cluster de banco de dados do Aurora MySQL.
ID do diretório na página Detalhes do diretório

Etapa 2: (Opcional) Criar uma confiança para um Active Directory on-premises

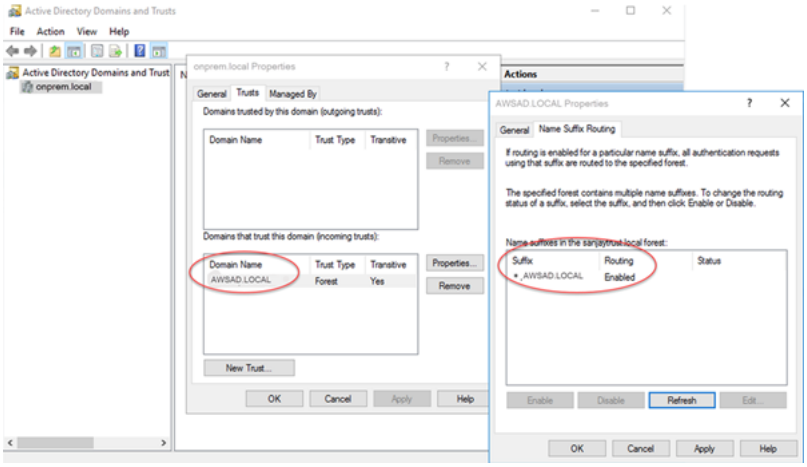
Se você não planeja usar seu próprio Microsoft Active Directory on-premises, vá para Etapa 3: Criar um perfil do IAM para ser usada pelo Amazon Aurora.

Para obter a autenticação Kerberos usando o Active Directory on-premises, é necessário criar uma relação de domínio confiável usando uma confiança de floresta entre o Microsoft Active Directory on-premises e o diretório AWS Managed Microsoft AD (criado em Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD). A relação de confiança pode ser unidirecional, onde o diretório AWS Managed Microsoft AD confia no Microsoft Active Directory on-premises. A confiança também pode ser bidirecional, onde os dois Active Directories confiam um no outro. Para ter mais informações sobre como configurar confianças usando o Directory Service, consulte Quando criar uma relação de confiança no Guia de administração do AWS Directory Service.

> **Note**
>
> Se você usa um Microsoft Active Directory on-premises:
>
> - Os clientes do Windows não podem se conectar usando endpoints personalizados do Aurora. Para saber mais, consulte Conexões de endpoints do Amazon Aurora.
> - Para bancos de dados globais:

- Os clientes do Windows podem se conectar usando endpoints de instância ou endpoints do cluster somente na Região da AWS primária do banco de dados global.
- Os clientes do Windows não podem se conectar usando endpoints do cluster em Regiões da AWS secundárias.

Verifique se o nome de domínio do Microsoft Active Directory on-premises inclui um roteamento de sufixo DNS que corresponde à relação de confiança recém-criada. A captura de tela a seguir mostra um exemplo.



Etapa 3: Criar um perfil do IAM para ser usada pelo Amazon Aurora

Para que o Amazon Aurora chame o Directory Service para você, é necessário ter um perfil do AWS Identity and Access Management (IAM) que use a política gerenciada `AmazonRDSDirectoryServiceAccess` do IAM. Este perfil permite que o Aurora faça chamadas ao Directory Service.

Quando você cria um cluster de banco de dados usando o Console de gerenciamento da AWS e tem a permissão `iam:CreateRole`, o console cria esse perfil automaticamente. Nesse caso, o nome da função é `rds-directoryservice-kerberos-access-role`. Caso contrário, é necessário criar a função do IAM manualmente. Ao criar essa função do IAM, escolha `Directory Service` e associe a AWS política gerenciada da `AmazonRDSDirectoryServiceAccess` a ela.

Para ter mais informações sobre como criar funções do IAM para um serviço, consulte o tópico sobre como Criar uma função para delegar permissões a um serviço da AWS, no Guia do usuário do IAM.

Se preferir, você poderá criar políticas com as permissões exigidas em vez de usar a política gerenciada do IAM `AmazonRDSDirectoryServiceAccess`. Nesse caso, o perfil do IAM deve ter a política de confiança do IAM a seguir.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

A função também deve ter a política de perfil do IAM a seguir.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
```

```
  ]
}
```

Etapa 4: Criar e configurar usuários

Você pode criar usuários com a ferramenta Usuários e computadores do Active Directory. Essa ferramenta faz parte das ferramentas do Active Directory Domain Services e do Active Directory Lightweight Directory Services. Os usuários representam pessoas ou entidades individuais que têm acesso ao seu diretório.

Para criar usuários em um diretório Directory Service, você usa uma instância on-premises ou do Amazon EC2 baseada no Microsoft Windows conectada seu diretório Directory Service. É necessário estar conectado à instância como um usuário que tenha privilégios para criar usuários. Para ter mais informações, consulte o tópico sobre como Gerenciar usuários e grupos AWS Managed Microsoft AD, no Guia de administração do AWS Directory Service.

Etapa 5: Criar ou modificar um cluster de banco de dados do Aurora MySQL

Crie ou modifique um cluster de banco de dados MySQL para usar com o diretório. É possível usar o console, a AWS CLI ou a API do RDS para associar um cluster de banco de dados a um diretório. Você pode realizar essa tarefa por meio de uma das seguintes maneiras:

- Crie um cluster de banco de dados do Aurora MySQL usando o console, o comando da CLI create-db-cluster ou a operação da API do RDS CreateDBCluster.

  Para instruções, consulte Criar um cluster de bancos de dados do Amazon Aurora.
- Modifique um cluster de banco de dados do Aurora MySQL existente usando o console, o comando da CLI modify-db-cluster ou a operação da API do RDS ModifyDBCluster.

  Para instruções, consulte Modificar um cluster de bancos de dados Amazon Aurora.
- Restaure um cluster de banco de dados do Aurora MySQL de um snapshot de banco de dados usando o console, o comando da CLI restore-db-cluster-from-snapshot ou a operação de API do RDS RestoreDBClusterFromSnapshot.

  Para instruções, consulte Restauração de um snapshot de um cluster de banco de dados.
- Restaure um cluster de banco de dados do Aurora MySQL para um momento específico usando o console, o comando da CLI restore-db-cluster-to-point-in-time ou a operação da API do RDS RestoreDBClusterToPointInTime.

Para instruções, consulte Restaurar um cluster de banco de dados para um horário especificado.

A autenticação Kerberos só é compatível com clusters de banco de dados do Aurora MySQL em uma VPC. O cluster de banco de dados pode estar na mesma VPC do diretório ou em uma VPC diferente. A VPC do cluster de banco de dados deve ter um grupo de segurança de VPC que permita a comunicação de saída com seu diretório.

Console

Ao usar o console para criar, modificar ou restaurar um cluster de banco de dados, escolha Kerberos authentication (Autenticação Kerberos) na seção Database authentication (Autenticação de banco de dados). Escolha Browse Directory (Procurar diretório) e selecione o diretório ou escolha Create a new directory (Criar um diretório).
Configuração da autenticação Kerberos ao criar um cluster de banco de dados

AWS CLI

Ao usar a AWS CLI ou a API do RDS, associe um cluster de banco de dados a um diretório. Os seguintes parâmetros são necessários para que o cluster de banco de dados utilize o diretório de domínio que você criou:

- Para o parâmetro `--domain`, use o identificador de domínio (identificador "d-*") gerado quando o diretório foi criado.
- Para o parâmetro `--domain-iam-role-name`, use a função criada que usa a política gerenciada `AmazonRDSDirectoryServiceAccess` do IAM.

Por exemplo, o comando da CLI a seguir modifica um cluster de banco de dados para usar um diretório.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Se você modificar um cluster de banco de dados para ativar a autenticação Kerberos, reinicialize as instâncias de banco de dados do leitor após a alteração.

Etapa 6: Criar usuários do Aurora MySQL que usam a autenticação Kerberos

O cluster de banco de dados é unido ao domínio AWS Managed Microsoft AD. Assim, é possível criar usuários do Aurora MySQL de usuários do Active Directory em seu domínio. As permissões de banco de dados são gerenciadas por meio de permissões padrão do Aurora MySQL que são concedidas e revogadas desses usuários.

Você pode permitir que um usuário do Active Directory seja autenticado pelo Aurora MySQL. Para fazer isso, primeiro use as credenciais do usuário primário do Amazon RDS para se conectar ao cluster de banco de dados do Aurora MySQL como você faz com qualquer outro cluster de banco de dados. Após o login, crie um usuário autenticado externamente com a autenticação Kerberos no Aurora MySQL como mostrado aqui:

```
CREATE USER user_name@'host_name' IDENTIFIED WITH 'authentication_kerberos' BY
 'realm_name';
```

- Substitua *user_name* pelo nome de usuário. Agora, os usuários (humanos e aplicações) de seu domínio poderão se conectar ao cluster de banco de dados de uma máquina cliente conectada ao domínio usando a autenticação Kerberos.
- Substitua *host_name* pelo nome do host. Você pode usar `%` como curinga. Você também pode usar endereços IP específicos para o nome do host.
- Substitua *realm_name* pelo nome do realm do diretório do domínio. O nome do realm geralmente é igual ao nome do domínio DNS em letras maiúsculas, como `CORP.EXAMPLE.COM`. Um realm é um grupo de sistemas que usam o mesmo Centro de Distribuição de Chaves do Kerberos.

O exemplo a seguir cria um usuário de banco de dados com o nome `Admin` que é autenticado no Active Directory com o nome do realm `MYSQL.LOCAL`.

```
CREATE USER Admin@'%' IDENTIFIED WITH 'authentication_kerberos' BY 'MYSQL.LOCAL';
```

Modificar um login existente do Aurora MySQL

Você também pode modificar um login existente do Aurora MySQL para usar a autenticação Kerberos utilizando a seguinte sintaxe:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Etapa 7: Configurar um cliente MySQL

Para configurar um cliente MySQL, execute as seguintes etapas:

1. Crie um arquivo `krb5.conf` (ou equivalente) para apontar para o domínio.
2. Verifique se o tráfego pode fluir entre o host do cliente e o Directory Service. Use um utilitário de rede, como o Netcat, para o seguinte:
   - Verifique o tráfego pelo DNS da porta 53.
   - Verifique o tráfego pelo TCP/UDP da porta 53 e do Kerberos, que inclui as portas 88 e 464 do Directory Service.
3. Verifique se o tráfego pode fluir entre o host do cliente e a instância de banco de dados pela porta do banco de dados. Por exemplo, use `mysql` para conectar e acessar o banco de dados.

Veja a seguir um exemplo de conteúdo `krb5.conf` para o AWS Managed Microsoft AD.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Veja a seguir um exemplo de conteúdo `krb5.conf` para o Microsoft Active Directory on-premises.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Etapa 8: (Opcional) Configurar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas

Por padrão, a capitalização das letras do nome de usuário do banco de dados MySQL deve corresponder à capitalização usada no login do Active Directory. No entanto, agora você pode usar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas com o plug-in `authentication_kerberos`. Para fazer isso, defina o parâmetro `authentication_kerberos_caseins_cmp` de cluster de banco de dados como `true`.

Como usar a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas

1. Crie um grupo de parâmetros de cluster de banco de dados personalizado. Siga os procedimentos em Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.
2. Edite o novo grupo de parâmetros para definir o valor de `authentication_kerberos_caseins_cmp` como `true`. Siga os procedimentos em Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

3. Associe o grupo de parâmetros de cluster de banco de dados ao cluster de banco de dados do Aurora MySQL. Siga os procedimentos em Associar um grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados no Amazon Aurora.
4. Reinicialize o cluster de banco de dados.

Conectar-se ao Aurora MySQL com a autenticação Kerberos

Para evitar erros, utilize um cliente MySQL com a versão 8.0.26 ou posterior em plataformas Unix, 8.0.27 ou posterior no Windows.

Utilizar o login do Kerberos do Aurora MySQL para se conectar ao cluster de banco de dados

Para se conectar ao Aurora MySQL com a autenticação Kerberos, você faz login como um usuário do banco de dados que você criou seguindo as instruções em Etapa 6: Criar usuários do Aurora MySQL que usam a autenticação Kerberos.

No prompt de comando, conecte-se a um dos endpoints associados ao cluster de banco de dados do Aurora MySQL. Quando a senha for solicitada, insira a senha do Kerberos associada a esse nome de usuário.

Quando você se autentica com o Kerberos, um tíquete de concessão de tíquetes (TGT) é gerado, caso ainda não exista. O plug-in da `authentication_kerberos` utiliza o TGT para obter um tíquete de serviço, que é então apresentado ao servidor de banco de dados do Aurora MySQL.

Você pode usar o cliente MySQL para se conectar ao Aurora MySQL com a autenticação Kerberos utilizando Windows ou Unix.

Unix

Você pode se conectar utilizando um dos seguintes métodos:

- Obtenha o TGT manualmente. Nesse caso, você não precisa fornecer a senha para o cliente MySQL.
- Forneça a senha para o login do Active Directory diretamente ao cliente MySQL.

O plug-in do lado do cliente é compatível com as plataformas Unix para as versões 8.0.26 e versões posteriores do cliente MySQL.

Como se conectar obtendo o TGT manualmente

1. Na interface de linha de comando, utilize o comando a seguir para obter o TGT.

```
kinit user_name
```

2. Utilize o comando `mysql` a seguir para fazer login no endpoint de instância de banco de dados de seu cluster de banco de dados.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name -p
```

> **Note**
>
> A autenticação poderá falhar se o keytab for alternado na instância de banco de dados. Nesse caso, obtenha um novo TGT executando novamente `kinit`.

Como se conectar diretamente

1. Na interface de linha de comando, utilize o comando `mysql` a seguir para fazer login no endpoint de instância de banco de dados de seu cluster de banco de dados.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name -p
```

2. Insira a senha do usuário do Active Directory.

Windows

No Windows, a autenticação geralmente é feita no momento do login; portanto, você não precisa obter o TGT manualmente para se conectar ao cluster de banco de dados do Aurora MySQL. A capitalização das letras do nome de usuário do banco de dados deve corresponder à capitalização das letras do usuário no Active Directory. Por exemplo, se o usuário no Active Directory for exibido como `Admin`, o nome de usuário do banco de dados deverá ser `Admin`.

O plug-in do lado do cliente é compatível com o Windows para as versões 8.0.27 e posteriores do cliente MySQL.

Como se conectar diretamente

- Na interface de linha de comando, utilize o comando `mysql` a seguir para fazer login no endpoint de instância de banco de dados de seu cluster de banco de dados.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name
```

Autenticação Kerberos com bancos de dados globais Aurora

A autenticação Kerberos para o Aurora MySQL é compatível com bancos de dados globais do Aurora. Para autenticar usuários no cluster de banco de dados secundário utilizando o Active Directory do cluster de banco de dados primário, replique o Active Directory para a Região da AWS secundária. Você ativa a autenticação Kerberos no cluster secundário utilizando o mesmo ID de domínio do cluster primário. A replicação AWS Managed Microsoft AD é compatível somente com a versão Enterprise do Active Directory. Para ter mais informações, consulte Replicação em várias regiões no Guia de administração do AWS Directory Service.

Migração do RDS para MySQL para o Aurora MySQL

Depois de migrar do RDS para MySQL com a autenticação Kerberos habilitada para o Aurora MySQL, modifique os usuários criados com o plug-in `auth_pam` para usar o plug-in `authentication_kerberos`. Por exemplo:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Evitar o armazenamento em cache de tíquetes

Se um TGT válido não existir quando a aplicação cliente MySQL for iniciada, a aplicação poderá obter o TGT e armazená-lo em cache. Se você quiser evitar que o TGT seja armazenado em cache, defina um parâmetro de configuração no arquivo `/etc/krb5.conf`.

> **Note**
>
> Essa configuração só se aplica a hosts de clientes que executam Unix, não Windows.

Para evitar o armazenamento em cache do TGT

- Adicione uma seção `[appdefaults]` a `/etc/krb5.conf` da seguinte forma:

```
[appdefaults]
  mysql = {
    destroy_tickets = true
  }
```

Registro em log da autenticação Kerberos

A variável de ambiente `AUTHENTICATION_KERBEROS_CLIENT_LOG` define o nível de registro em log para a autenticação Kerberos. Você pode usar os logs para depuração do lado do cliente.

Os valores permitidos são 1 a 5. As mensagens de log são gravadas na saída de erro padrão. A tabela a seguir descreve cada nível de registro em log.

Nível de registro	Descrição
1 ou não definido	Sem registro em log
2	Mensagens de erro
3	Mensagens de erro e aviso
4	Mensagens de erro, aviso e informação
5	Mensagens de erro, aviso, informação e depuração

Gerenciar um cluster de banco de dados em um domínio

É possível utilizar a AWS CLI ou a API do RDS para gerenciar o cluster de banco de dados e as respectivas relações com o Active Directory gerenciado. Por exemplo, você pode associar um Active Directory para autenticação Kerberos e desassociar um Active Directory para desativar a autenticação Kerberos. Também é possível mover um cluster de banco de dados para ser autenticado externamente por um Active Directory para outro.

Por exemplo, usando a API do Amazon RDS, você pode fazer o seguinte:

- Para tentar ativar novamente a autenticação Kerberos para uma associação com falha, utilize a operação `ModifyDBInstance` da API e especifique o ID do diretório da associação atual.

- Para atualizar o nome da função do IAM para a associação, use a operação `ModifyDBInstance` da API e especifique o ID do diretório da associação atual e a nova função do IAM.
- Para desativar a autenticação Kerberos em um cluster de banco de dados, utilize a operação `ModifyDBInstance` da API e especifique `none` como o parâmetro de domínio.
- Para mover um cluster de banco de dados de um domínio para outro, utilize a operação `ModifyDBInstance` da API e especifique o identificador do novo domínio como o parâmetro do domínio.
- Para listar as associações de cada cluster de banco de dados, utilize a operação `DescribeDBInstances` da API.

Compreensão da associação de domínio

Depois que você cria ou modifica o cluster de banco de dados, ele se torna membro do domínio. É possível visualizar o status da associação do domínio do cluster de banco de dados executando o comando describe-db-clusters da CLI. O status do cluster de banco de dados pode ser um dos seguintes:

- `kerberos-enabled`: o cluster de banco de dados tem a autenticação Kerberos ativada.
- `enabling-kerberos`: a AWS está no processo de ativar a autenticação Kerberos nesse cluster de bancos de dados.
- `pending-enable-kerberos`: a ativação da autenticação Kerberos está pendente nesse cluster de bancos de dados.
- `pending-maintenance-enable-kerberos`: a AWS tentará ativar a autenticação Kerberos no cluster de banco de dados durante a próxima janela de manutenção programada.
- `pending-disable-kerberos`: a desativação da autenticação Kerberos está pendente nesse cluster de bancos de dados.
- `pending-maintenance-disable-kerberos`: a AWS tentará desativar a autenticação Kerberos no cluster de banco de dados durante a próxima janela de manutenção programada.
- `enable-kerberos-failed`: um problema de configuração impediu que a AWS ativasse a autenticação Kerberos no cluster de banco de dados. Confira e corrija sua configuração antes de emitir novamente o comando modify do cluster de banco de dados.
- `disabling-kerberos`: a AWS está no processo de desativar a autenticação Kerberos nesse cluster de bancos de dados.

Uma solicitação para ativar a autenticação Kerberos pode falhar por conta de um problema de conectividade de rede ou de um perfil do IAM incorreto. Por exemplo, suponha que você crie um cluster de banco de dados ou modifique um existente e ocorra uma falha na tentativa de ativar a autenticação Kerberos. Se esse for o caso, emita o comando modify novamente ou modifique o cluster de banco de dados recém-criado para ingressar no domínio.

Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL

Você tem várias opções para migrar dados do seu banco de dados existente para um cluster de banco de dados MySQL do Amazon Aurora. Suas opções de migração também dependem do banco de dados do qual você está migrando e do tamanho dos dados que você está migrando.

Existem dois tipos diferentes de migração: física e lógica. A migração física significa que as cópias físicas dos arquivos de banco de dados serão usadas para migrar o banco de dados. A migração lógica significa que a migração será realizada aplicando mudanças de banco de dados lógicas, como inserções, atualizações e exclusões.

A migração física tem as seguintes vantagens:

- A migração física é mais rápida do que a migração lógica, especialmente para grandes bancos de dados.
- O performance do banco de dados não é afetado quando um backup é feito por migração física.
- A migração física pode migrar tudo no banco de dados de origem, incluindo componentes de banco de dados complexos.

A migração física tem as seguintes limitações:

- O parâmetro `innodb_page_size` precisa ser definido com o valor padrão (`16KB`).
- O parâmetro `innodb_data_file_path` deve ser configurado com apenas um arquivo de dados que usa o nome de arquivo de dados padrão `"ibdata1:12M:autoextend"`. Bancos de dados com dois arquivos de dados ou com um arquivo de dados com um nome diferente não podem ser migrados usando esse método.

  Veja a seguir, exemplos de nomes de arquivos que não são permitidos: `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` e `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- O parâmetro `innodb_log_files_in_group` precisa ser definido com o valor padrão (`2`).

A migração lógica tem as seguintes vantagens:

- Você pode migrar subconjuntos do banco de dados, como tabelas específicas ou partes de uma tabela.

- Os dados podem ser migrados independentemente da estrutura de armazenamento físico.

A migração lógica tem as seguintes limitações:

- Geralmente, a migração lógica é mais lenta do que a migração física.
- Os componentes complexos do banco de dados podem atrasar o processo de migração lógica. Em alguns casos, os componentes complexos do banco de dados podem até bloquear a migração lógica.

A tabela a seguir detalha as opções disponíveis e o tipo de migração para cada uma delas.

Migrar a partir de	Migration type	Solução
Uma instância de banco de dados RDS for MySQL	Físico	É possível migrar de uma instância de banco de dados RDS for MySQL criando antes uma réplica de leitura do Aurora MySQL de uma instância de banco de dados MySQL. Quando o atraso da réplica entre a instância de banco de dados MySQL e a réplica de leitura do Aurora MySQL é 0, você pode direcionar seus aplicativos clientes para leitura da réplica de leitura do Aurora e interromper a replicação, a fim de que a réplica de leitura do Aurora MySQL se torne um cluster de banco de dados autônomo do Aurora MySQL para leitura e gravação. Para obter mais detalhes, consulte Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora.
Um snapshot de banco de dados RDS for MySQL	Físico	Você pode migrar dados diretamente de um snapshot de banco de dados RDS for MySQL para um cluster de banco de dados do Amazon Aurora MySQL. Para obter mais detalhes, consulte Migrar de um snapshot do RDS for MySQL para o Aurora.
Um banco de dados MySQL externo para o Amazon RDS	Lógico	Você pode criar um despejo dos seus dados usando o utilitário `mysqldump` e importando os dados para um

Migrar a partir de	Migration type	Solução
		cluster existente de banco de dados do Amazon Aurora MySQL. Para obter mais detalhes, consulte Migração lógica do MySQL para o Amazon Aurora MySQL usando o mysqldump. Para exportar metadados para usuários de banco de dados durante a migração de um banco de dados externo do MySQL, também é possível usar um comando do MySQL Shell em vez de `mysqldump` . Consulte mais informaçõ es em Instance Dump Utility, Schema Dump Utility, and Table Dump Utility. **Note** O utilitário mysqlpump foi descontinuado desde o MySQL 8.0.34.
Um banco de dados MySQL externo para o Amazon RDS	Físico	Você pode copiar arquivos de backup do seu banco de dados em um bucket do Amazon Simple Storage Service (Amazon S3) e restaurar um cluster de banco de dados do Amazon Aurora MySQL desses arquivos. Essa opção pode ser considerada mais rápida do que migrar dados usando `mysqldump` . Para obter mais detalhes, consulte Migração física do MySQL usando o Percona XtraBackup e o Amazon S3.

Migrar a partir de	Migration type	Solução
Um banco de dados MySQL externo para o Amazon RDS	Lógico	Você pode economizar dados do banco de dados como arquivos de texto e copiar esses arquivos para um bucket do Amazon S3. Em seguida, você pode carregar os dados para um cluster existente de banco de dados do Aurora MySQL usando o comando `LOAD DATA FROM S3` do MySQL. Para ter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.
Um banco de dados que não compatível com MySQL	Lógico	Você pode usar AWS Database Migration Service (AWS DMS) para migrar dados de um banco de dados incompatível com MySQL. Para obter mais informações sobre o AWS DMS, consulte O que é o serviço de migração de banco de dados da AWS?

Note

Se você estiver migrando um banco de dados externo do MySQL para o Amazon RDS, as opções de migração descritas na tabela serão aceitas somente se seu banco de dados for compatível os espaços de tabela do InnoDB ou MyISAM.

Se o banco de dados MySQL que você está migrando para o Aurora MySQL usar `memcached`, remova o `memcached` antes de migrá-lo.

Não é possível migrar de algumas versões anteriores a 8.0 do MySQL para o Aurora MySQL 3.05, incluindo 8.0.11, 8.0.13 e 8.0.15. Recomendamos que você atualize para a versão 8.0.28 do MySQL antes da migração.

Migrar dados de um banco de dados MySQL externo para um cluster de banco de dados do Amazon Aurora MySQL

Se o seu banco de dados é compatível com tablespaces InnoDB ou MyISAM, estas são as opções para migrar os dados para um cluster de banco de dados Amazon Aurora MySQL:

- Você pode criar um despejo dos seus dados usando o utilitário `mysqldump` e importando os dados para um cluster existente de banco de dados do Amazon Aurora MySQL. Para obter mais informações, consulte Migração lógica do MySQL para o Amazon Aurora MySQL usando o mysqldump.
- Você pode copiar arquivos de backup completos e incrementais do banco de dados para um bucket do Amazon S3 e restaurar para um cluster de banco de dados do Amazon Aurora MySQL usando esses arquivos. Essa opção pode ser considerada mais rápida do que migrar dados usando `mysqldump`. Para obter mais informações, consulte Migração física do MySQL usando o Percona XtraBackup e o Amazon S3.

Tópicos

- Migração física do MySQL usando o Percona XtraBackup e o Amazon S3
- Migração lógica do MySQL para o Amazon Aurora MySQL usando o mysqldump

Migração física do MySQL usando o Percona XtraBackup e o Amazon S3

Você pode copiar os arquivos de backup completos e incrementais do banco de dados MySQL de origem, versão 5.7 ou 8.0, para um bucket do Amazon S3. Em seguida, você pode restaurar para um cluster de banco de dados do Amazon Aurora MySQL com a mesma versão principal do mecanismo de banco de dados desses arquivos.

Essa opção pode ser considerada mais rápida do que migrar dados usando `mysqldump`, pois o uso de `mysqldump` repete todos os comandos para recriar o esquema e os dados do seu banco de dados de origem no novo cluster de banco de dados Aurora MySQL. Ao copiar os arquivos de dados de origem MySQL, o Aurora MySQL pode usá-los imediatamente como dados para um cluster de banco de dados do Aurora MySQL.

Você pode também minimizar o tempo de inatividade usando a replicação de log binário durante o processo de migração. Se você usar a replicação de log binário, ao banco de dados externo de MySQL permanece aberta para transações enquanto os dados estão sendo migrados para o

cluster de banco de dados Aurora MySQL. Depois do cluster de banco de dados do Aurora MySQL ter sido criado, você usa a replicação de log binário para sincronizar o cluster de banco de dados do Aurora MySQL com as transações que aconteceram depois do backup. Quando o cluster de banco de dados do Aurora MySQL é acessado com o banco de dados do MySQL, você conclui a migração trocando completamente para o cluster de banco de dados do Aurora MySQL para novas transações. Para obter mais informações, consulte Sincronizar o cluster do banco de dados do Amazon Aurora MySQL com banco de dados MySQL usando a replicação.

Sumário

- Limitações e considerações
- Antes de começar
  - Instalação do Percona XtraBackup
  - Permissões obrigatórias
  - Criar a função de serviço do IAM
- Fazer backup de arquivos a serem restaurados como um cluster de banco de dados do Amazon Aurora MySQL
  - Criar um backup completo com o Percona XtraBackup
  - Usar backups incrementais com o Percona XtraBackup
  - Considerações sobre backup
- Restaurar um cluster de banco de dados do Amazon Aurora MySQL de um bucket do Amazon S3
- Sincronizar o cluster do banco de dados do Amazon Aurora MySQL com banco de dados MySQL usando a replicação
  - Configurar seu banco de dados MySQL externo e seu cluster de banco de dados do Aurora MySQL para replicação criptografada
  - Sincronizar o cluster de banco de dados do Amazon Aurora MySQL com o banco de dados MySQL externo
- Reduzir o tempo de migração física para o Amazon Aurora MySQL
  - Tipos de tabela não compatíveis
  - Contas de usuário com privilégios não compatíveis
  - Privilégios dinâmicos no Aurora MySQL versão 3
  - Objetos armazenados com ‘rdsadmin'@'localhost’ como definidor

Limitações e considerações

As limitações e considerações a seguir se aplicam à restauração para um cluster de banco de dados do Amazon Aurora MySQL usando um bucket do Amazon S3:

- Você pode migrar os dados somente para um novo cluster de banco de dados e não para um cluster existente.
- Use o Percona XtraBackup para fazer backup de dados no S3. Para obter mais informações, consulte Instalação do Percona XtraBackup.
- O bucket do Amazon S3 e o cluster de banco de dados do Aurora MySQL devem estar localizados na mesma região da AWS.
- Não é possível restaurar nos seguintes casos:
  - De uma exportação de snapshot de cluster de banco de dados para o Amazon S3. Também não é possível migrar dados de uma exportação de snapshot de cluster de banco de dados para o bucket do S3.
  - De um banco de dados de origem criptografado. Mas é possível criptografar os dados que estão sendo migrados. Também é possível deixar os dados não criptografados durante o processo de migração.
  - De um banco de dados MySQL 5.5 ou 5.6.
- O Percona Server para MySQL não é aceito como banco de dados de origem porque ele pode conter tabelas `compression_dictionary*` no esquema `mysql`.
- Não é possível realizar uma restauração para um cluster de banco de dados do Aurora Serverless.
- A reversão de migrações não é uma operação compatível com versões principais nem com secundárias. Por exemplo, não é possível migrar do MySQL versão 8.0 para o Aurora MySQL versão 2 (compatível com o MySQL 5.7) nem do MySQL versão 8.0.32 para o Aurora MySQL versão 3.03, que é compatível com a versão 8.0.26 da comunidade do MySQL.
- Não é possível migrar de algumas versões anteriores a 8.0 do MySQL para o Aurora MySQL 3.05, incluindo 8.0.11, 8.0.13 e 8.0.15. Recomendamos que você atualize para a versão 8.0.28 do MySQL antes da migração.
- Não é possível fazer importações do Amazon S3 na classe de instância de banco de dados db.t2.micro. Contudo, é possível restaurar para outra classe de instância de banco de dados e alterar a instância de banco de dados posteriormente. Para ter mais informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.

- O Amazon S3 limita o tamanho de um arquivo carregado para um bucket do S3 a 5 TB. Se um arquivo de backup exceder 5 TB, você deverá dividi o arquivo de backup em arquivos menores.
- O Amazon RDS limita a 1 milhão o número de arquivos carregados para um bucket do S3. Se os dados de backup do banco de dados, incluindo todos os backups completos e incrementais, exceder 1 milhão de arquivos, use um arquivo Gzip (.gz), tar (.tar.gz) ou Percona xbstream (.xbstream) para armazenar arquivos de backup completos e incrementais no bucket do S3. O Percona XtraBackup 8.0 oferece suporte apenas ao Percona xbstream para compactação.
- Para fornecer serviços de gerenciamento para cada cluster de banco de dados, o usuário `rdsadmin` é criado quando o cluster de banco de dados é criado. Como esse é um usuário reservado no RDS, as seguintes limitações se aplicam:
  - Funções, procedimentos, visualizações, eventos e acionadores com o `'rdsadmin'@'localhost'` como definidor não são importados. Para obter mais informações, consulte Objetos armazenados com 'rdsadmin'@'localhost' como definidor e Privilégios de usuário mestre com Amazon Aurora MySQL..
  - Quando o cluster de banco de dados do Aurora MySQL é criado, um usuário principal é criado com os privilégios máximos aceitos. Na restauração por meio de backup, quaisquer privilégios incompatíveis atribuídos aos usuários que estão sendo importados são removidos automaticamente durante a importação.

    Para identificar usuários que possam ser afetados por isso, consulteContas de usuário com privilégios não compatíveis. Para obter mais informações sobre privilégios compatíveis no Aurora MySQL, consulte Modelo de privilégios baseados em funções.
- No Aurora MySQL versão 3, os privilégios dinâmicos não são importados. Os privilégios dinâmicos aceitos pelo Aurora podem ser importados após a migração. Para obter mais informações, consulte Privilégios dinâmicos no Aurora MySQL versão 3.
- Tabelas criadas pelo usuário no esquema do `mysql` não são migradas.
- O parâmetro `innodb_data_file_path` deve ser configurado com apenas um arquivo de dados que usa o nome de arquivo de dados padrão `ibdata1:12M:autoextend`. Bancos de dados com dois arquivos de dados ou com um arquivo de dados com um nome diferente não podem ser migrados usando esse método.

  Veja a seguir, exemplos de nomes de arquivos que não são permitidos: `innodb_data_file_path=ibdata1:50M`, `ibdata2:50M:autoextend` e `innodb_data_file_path=ibdata01:50M:autoextend`.

- Não é possível migrar de um banco de dados de origem que tenha tabelas definidas fora do diretório de dados MySQL padrão.
- No momento, o tamanho máximo aceito para backups não compactados usando esse método é 64 TiB. Para backups compactados, esse limite é menor para levar em conta os requisitos de espaço sem compactação. Nesses casos, o tamanho máximo de backup aceito seria `64 TiB – compressed backup size`.
- O Aurora MySQL não é compatível com a importação do MySQL e de outros componentes e plug-ins externos.
- O Aurora MySQL não restaura tudo do seu banco de dados. Recomendamos salvar o esquema do banco de dados e os valores dos itens do banco de dados MySQL de origem relacionados abaixo e adicioná-los ao cluster de banco de dados do Aurora MySQL restaurado depois que ele for criado:
  - Contas de usuário
  - Funções
  - Procedimentos armazenados
  - Informações de fuso horário. As informações de fuso horário são carregadas do sistema operacional local do cluster de banco de dados do Aurora MySQL. Para obter mais informações, consulte Fuso horário local para os clusters de bancos de dados Amazon Aurora.

Antes de começar

Antes de copiar os dados para um bucket do Amazon S3 e restaurar um cluster de banco de dados desses arquivos, faça o seguinte:

- Instale o Percona XtraBackup no seu servidor local.
- Permita que o Aurora MySQL acesse o seu bucket do Amazon S3 em seu nome.

Instalação do Percona XtraBackup

O Amazon Aurora pode restaurar um cluster de banco de dados a partir de arquivos criados com o Percona XtraBackup. Você pode instalar o Percona XtraBackup em Software Downloads - Percona.

Para migração do MySQL 5.7, use o Percona XtraBackup 2.4.

Para migração do MySQL 8.0, use o Percona XtraBackup 8.0. Verifique se a versão do Percona XtraBackup é compatível com a versão do mecanismo do banco de dados de origem.

Permissões obrigatórias

Para migrar os dados do MySQL para um cluster de banco de dados do Amazon Aurora MySQL, são necessárias várias permissões:

- O usuário que está solicitando que o Aurora crie um novo cluster de um bucket do Amazon S3 deve ter permissão para listar os buckets da sua conta da AWS. Você concede essa permissão ao usuário usando uma política do AWS Identity and Access Management (IAM).
- O Aurora exige permissão para atuar em seu nome e acessar o bucket do Amazon S3 em que você armazena os arquivos usados para criar o cluster de banco de dados do Amazon Aurora MySQL. Conceda ao Aurora as permissões necessárias usando uma função de serviço do IAM.
- O usuário que faz a solicitação também deve ter permissão para indicar as funções do IAM de sua conta da AWS.
- Se o usuário que faz a solicitação for criar a função de serviço do IAM ou solicitar que o Aurora crie a função de serviço do IAM (usando o console), ele deverá ter permissão para criar uma função do IAM para sua conta da AWS.
- Se você pretende criptografar os dados durante o processo de migração, atualize a política do IAM do usuário responsável pela migração para conceder acesso do RDS às AWS KMS keys usadas para criptografar os backups. Para instruções, consulte Criar uma política do IAM para acessar recursos do AWS KMS.

Por exemplo, a política do IAM a seguir concede a um usuário as permissões mínimas exigidas para usar o console para listar funções do IAM, criar uma função do IAM, listar os buckets do Amazon S3 para a sua conta e listar as chaves do KMS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "s3:ListBucket",
```

```
                "kms:ListKeys"
            ],
            "Resource": "*"
        }
    ]
}
```

Além disso, para um usuário associar uma função do IAM a um bucket do Amazon S3, o usuário do IAM deve ter a permissão `iam:PassRole` para aquela função do IAM. Essa permissão autoriza que um administrador restrinja as funções do IAM que um usuário pode associar a buckets do Amazon S3.

Por exemplo, a seguinte política do IAM permite que um usuário associe a função chamada `S3Access` a um bucket do Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

Para obter mais informações sobre as permissões de usuário do IAM, consulte Gerenciamento do acesso usando políticas.

Criar a função de serviço do IAM

Para solicitar que o Console de gerenciamento da AWS crie uma função para você, selecione a opção Create a New Role (Criar uma nova função) (mostrada posteriormente neste tópico). Se você selecionar essa opção e especificar um nome para a nova função, o Aurora criará a função de serviço do IAM necessária para o Aurora acessar o bucket do Amazon S3 com o nome fornecido.

Você também pode criar a função manualmente usando o procedimento seguinte.

Como criar uma função do IAM para o Aurora acessar o Amazon S3

1. Siga as etapas em Criar uma política do IAM para acessar recursos do Amazon S3.
2. Siga as etapas em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.
3. Siga as etapas em Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.

Fazer backup de arquivos a serem restaurados como um cluster de banco de dados do Amazon Aurora MySQL

Você pode criar um backup completo de seus arquivos de banco de dados MySQL usando o Percona XtraBackup e fazer upload dos arquivos de backup em um bucket do Amazon S3. Ou, se você já usa o Percona XtraBackup para fazer o backup dos arquivos do banco de dados MySQL, pode fazer upload dos arquivos e diretórios de backup completos e incrementais em um bucket do Amazon S3.

Tópicos

- Criar um backup completo com o Percona XtraBackup
- Usar backups incrementais com o Percona XtraBackup
- Considerações sobre backup

Criar um backup completo com o Percona XtraBackup

Para criar um backup completo dos arquivos do banco de dados MySQL que podem ser restaurados do Amazon S3 para criar um cluster de banco de dados do Aurora MySQL, use o utilitário Percona XtraBackup (`xtrabackup`) para fazer backup do banco de dados.

Por exemplo, o seguinte comando cria um backup de um banco de dados MySQL e armazena os arquivos na pasta `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

Se você deseja compactar o backup em um único arquivo (que pode ser dividido, se necessário), use a opção `--stream` para salvar o backup em um dos seguintes formatos:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

O comando a seguir cria um backup do seu banco de dados MySQL dividido em vários arquivos Gzip.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

O comando a seguir cria um backup do seu banco de dados MySQL dividido em vários arquivos tar.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

O comando a seguir cria um backup do seu banco de dados MySQL dividido em vários arquivos xbstream.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> Se você vir o erro a seguir, saiba que ele pode ser causado pela mistura de formatos de arquivo em seu comando:
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Após fazer o backup do seu banco de dados MySQL usando o utilitário Percona XtraBackup, você poderá copiar os arquivos e diretórios de backup para um bucket do Amazon S3.

Para obter informações sobre a criação e o upload de um arquivo para um bucket do Amazon S3, consulte Conceitos básicos do Amazon Simple Storage Service, no Guia de conceitos básicos do Amazon S3.

Usar backups incrementais com o Percona XtraBackup

O Amazon Aurora MySQL oferece suporte a backups completos e incrementais criados com o Percona XtraBackup. Se você já usa o Percona XtraBackup para fazer backups completos e incrementais de seus arquivos de banco de dados MySQL, não precisa criar um backup completo e fazer upload dos arquivos de backup no Amazon S3. Em vez disso, você pode economizar muito tempo copiando os diretórios e arquivos de backup existentes para seus backups completos e incrementais para um bucket do Amazon S3. Para obter mais informações, consulte Create an incremental backup no site da Percona.

Quando copiar os arquivos existentes de backup completo e incremental para um bucket do Amazon S3, copie recursivamente o conteúdo do diretório de base. Esse conteúdo inclui o backup completo e também todo o backup incremental dos diretórios e arquivos. Essa cópia deve preservar a estrutura de diretórios no bucket do Amazon S3. O Aurora percorre todos os arquivos e diretórios. O Aurora usa o arquivo `xtrabackup-checkpoints` incluído em cada backup incremental para identificar o diretório de base e ordenar os backups incrementais por intervalo de número de sequência de log (LSN).

Para obter informações sobre a criação e o upload de um arquivo para um bucket do Amazon S3, consulte Conceitos básicos do Amazon Simple Storage Service, no Guia de conceitos básicos do Amazon S3.

Considerações sobre backup

O Aurora não oferece suporte a backups parciais criados com o Percona XtraBackup. Você não pode usar as seguintes opções para criar um backup parcial quando faz backup dos arquivos de origem de seu banco de dados: `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` ou `--databases-file`.

Para ter mais informações sobre como fazer backup do banco de dados com o Percona XtraBackup, consulte Percona XtraBackup - Documentation e Work with binary logs no site da Percona.

O Aurora oferece suporte a backups incrementais criados com o Percona XtraBackup. Para obter mais informações, consulte Create an incremental backup no site da Percona.

O Aurora consome seus arquivos de backup com base no nome do arquivo. Renomeie seus arquivos de backup com a extensão de arquivo apropriada com base no formato do arquivo — por exemplo, `.xbstream` para arquivos armazenados usando o formato Percona xbstream.

O Aurora consome os arquivos de backup em ordem alfabética assim como na ordem numérica natural. Sempre use a opção `split` ao emitir o comando `xtrabackup` para garantir que os arquivos de backup sejam gravados e nomeados na ordem apropriada.

O Amazon S3 limita o tamanho de um arquivo carregado para um bucket do Amazon S3 a 5 TB. Se os dados de backup do seu banco de dados ultrapassarem 5 TB, use o comando `split` para dividir os arquivos de backup em vários arquivos com menos de 5 TB cada.

O Aurora limita a 1 milhão o número de arquivos de origem enviados por upload para um bucket do Amazon S3. Em alguns casos, os dados de backup de seu banco de dados, incluindo todos os backups completos e incrementais, podem aumentar para um grande número de arquivos. Nesses casos, use um arquivo de tarball (.tar.gz) para armazenar arquivos completos e incrementais de backup no bucket Amazon S3.

Quando você carrega um arquivo no bucket Amazon S3, você pode usar criptografia por parte do servidor para criptografar os dados. Então você pode restaurar o cluster de banco de dados do Amazon Aurora MySQL a partir desses arquivos criptografados. O Amazon Aurora MySQL pode restaurar um cluster de banco de dados com arquivos criptografados usando os seguintes tipos de criptografia do lado do servidor:

- Criptografia de servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) – cada objeto é criptografado com uma chave única empregando uma criptografia multifator forte.
- Criptografia do lado do servidor com chaves gerenciadas pelo AWS KMS (SSE-KMS): semelhante à SSE-S3, mas com a opção de você mesmo criar e gerenciar chaves de criptografia, além de outras diferenças.

Para obter informações sobre como usar a criptografia do lado do servidor ao carregar arquivos para um bucket do Amazon S3, consulte Proteger dados usando a criptografia do lado do servidor no Guia de desenvolvedor do Amazon S3.

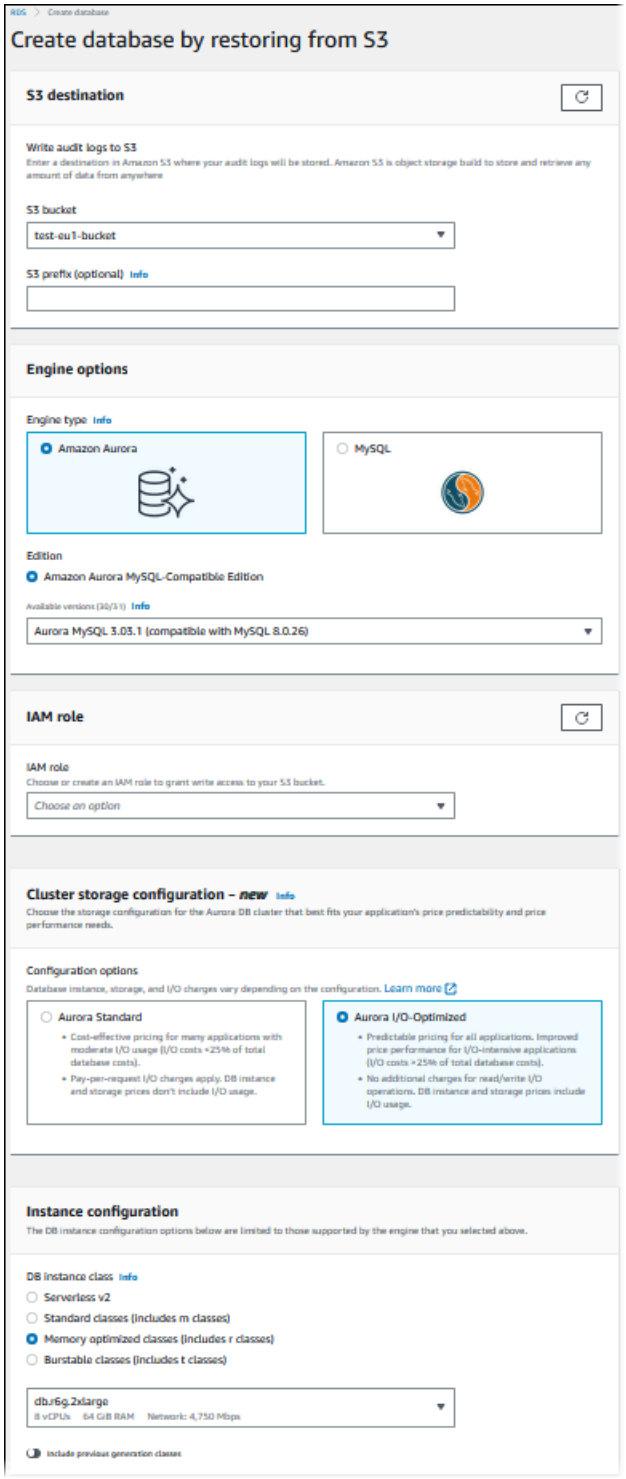
Restaurar um cluster de banco de dados do Amazon Aurora MySQL de um bucket do Amazon S3

É possível restaurar os arquivos de backup do bucket do Amazon S3 para criar um novo cluster de banco de dados do Amazon Aurora MySQL usando o console do Amazon RDS.

Para restaurar um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos em um bucket do Amazon S3

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do console do console do Amazon RDS, escolha a região da AWS na qual deseja criar o cluster de banco de dados. Escolha o mesmo nome da região da AWS que o bucket Amazon S3 que contém o backup do banco de dados.
3. No painel de navegação, escolha Databases (Bancos de dados) e depois escolha Restore from S3 (Restaurar de S3).
4. Escolha Restore from S3 (Restaurar do S3).

   A página Create database by restoring from S3 (Criar banco de dados restaurando a partir do S3) é exibida.



5. Em S3 destination (destino do S3):

   a. Escolha o bucket do S3 que contém seus arquivos backup.

   b. (Opcional) Em S3 folder path prefix (Prefixo do caminho da pasta do S3), digite um prefixo de caminho de arquivo para os arquivos armazenados no bucket do Amazon S3.

Se você não especificar um prefixo, o RDS criará a instância de banco de dados usando todos os arquivos e as pastas na pasta raiz do bucket do S3. Se você especificar um prefixo, o RDS criará a instância de banco de dados usando os arquivos e as pastas no bucket do S3 no qual o caminho para o arquivo começa com o prefixo especificado.

Por exemplo, suponha que você armazene seus arquivos de backup no S3 em uma subpasta denominada backups e que você tenha vários conjuntos de arquivos de backup, cada um em seu próprio diretório (gzip_backup1, gzip_backup2 e assim por diante). Nesse caso, especifique um prefixo de backups/gzip_backup1 para restaurar dos arquivos na pasta gzip_backup1.

6. Em Engine options (Opções de mecanismo):

   a. Para Engine type (Tipo de mecanismo), escolha Amazon Aurora.

   b. Em Version (Versão), escolha a versão do Aurora MySQL mecanismo para sua instância de banco de dados restaurada.

7. Para IAM role (Função do IAM), é possível escolher uma função existente do IAM.

8. (Opcional) Você também pode ter uma nova função do IAM escolhendo Create a New Role (Criar uma nova função). Em caso afirmativo:

   a. Insira o IAM role name (Nome da função do IAM).

   b. Escolha se deseja Allow access to KMS key (Permitir acesso à chave do KMS):

      - Se você não criptografou os arquivos de backup, escolha No (Não).
      - Se você criptografou os arquivos de backup com AES-256 (SSE-S3) ao fazer upload para o Amazon S3, escolha No (Não). Nesse caso, os dados são descriptografados automaticamente.
      - Se você criptografou os arquivos de backup com criptografia do lado do servidor do AWS KMS (SSE-KMS) ao fazer upload para o Amazon S3, escolha Yes (Sim). A seguir, escolha a chave do KMS correta para AWS KMS key.

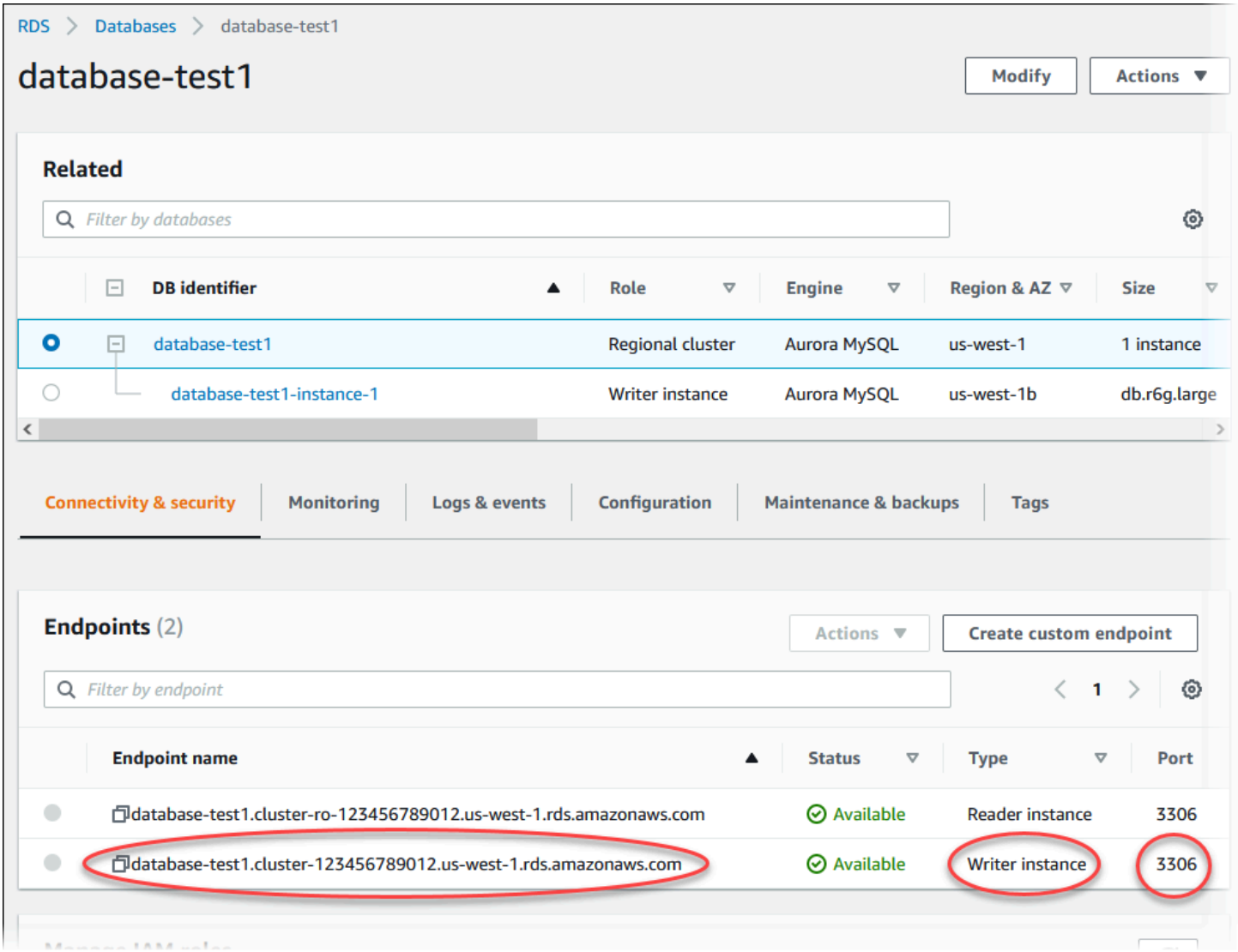
        O Console de gerenciamento da AWS cria uma política do IAM que habilita o Aurora para descriptografar os dados.

   Para obter mais informações, consulte Proteger dados usando criptografia do lado do servidor no Guia do desenvolvedor do Amazon S3.

9. Escolha configurações para o cluster de banco de dados, como a configuração do armazenamento do cluster de banco de dados, a classe de instância de banco de dados, o identificador do cluster de banco de dados e as credenciais de login. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.
10. Personalize configurações adicionais para o cluster de banco de dados do Aurora MySQL conforme necessário.
11. Escolha Create database (Criar banco de dados) para executar sua instância de banco de dados de Aurora.

No console do Amazon RDS, a nova instância de banco de dados é exibida na lista de instâncias de banco de dados. A instância de banco de dados fica com o status creating (criando) até que esteja criada e pronta para uso. Quando o status for alterado para available, você poderá se conectar à instância primária do seu cluster de banco de dados. Dependendo da classe da instância de banco de dados e do armazenamento alocado, pode levar alguns minutos até que a nova instância fique disponível.

Para visualizar o cluster recém-criado, escolha a visualização Databases (Bancos de dados) no console do Amazon RDS e escolha o cluster de banco de dados. Para obter mais informações, consulte Visualizar um cluster de bancos de dados Amazon Aurora.



Anote a porta e o endpoint do gravador do cluster do banco de dados. Use o endpoint do gravador e a porta do cluster do banco de dados em suas strings de conexão JDBC e ODBC para qualquer aplicativo que realize operações de gravação ou leitura.

Sincronizar o cluster do banco de dados do Amazon Aurora MySQL com banco de dados MySQL usando a replicação

Para alcançar pouco ou quase nenhum tempo de inatividade durante a migração, você pode replicar transações que foram confirmadas em seu banco de dados do MySQL para seu cluster de banco de dados do Aurora MySQL. A replicação permite que o cluster do banco de dados recupere as transações no banco de dados do MySQL que aconteceram durante a migração. Quando o cluster do banco de dados é completamente recuperado, você pode interromper a replicação e terminar a migração para Aurora MySQL.

Tópicos

- Configurar seu banco de dados MySQL externo e seu cluster de banco de dados do Aurora MySQL para replicação criptografada
- Sincronizar o cluster de banco de dados do Amazon Aurora MySQL com o banco de dados MySQL externo

Configurar seu banco de dados MySQL externo e seu cluster de banco de dados do Aurora MySQL para replicação criptografada

Para replicar dados de forma segura, você pode usar a replicação criptografada.

> **Note**
>
> Se você não precisar usar a replicação criptografada, você pode passar essas etapas e seguir para as instruções em Sincronizar o cluster de banco de dados do Amazon Aurora MySQL com o banco de dados MySQL externo.

Veja a seguir os pré-requisitos para usar a replicação criptografada:

- O Secure Sockets Layer (SSL) deve ser habilitado no banco de dados primário do MySQL.
- Uma chave e certificado de cliente devem estar preparados para o cluster do banco de dados do Aurora MySQL.

Durante a replicação criptografada, o cluster do banco de dados de Aurora MySQL age como um cliente para o servidor de banco de dados do MySQL. Os certificados e chaves do cliente Aurora MySQL estão nos arquivos em formato .pem.

Para configurar seu banco de dados MySQL externo e seu cluster de banco de dados do Aurora MySQL para replicação criptografada

1. Certifique-se de que você estará preparado para a replicação criptografada:
    - Se você não tem o SSL habilitado no banco de dados primário externo do MySQL e não tem uma chave de cliente e um certificado de cliente preparados, habilite o SSL no servidor de banco de dados MySQL e gere a chave de cliente e o certificado de cliente necessários.

- Se o SSL estiver habilitado no primário externo, forneça uma chave e um certificado de cliente para o cluster de banco de dados do Aurora MySQL. Se você não os tiver, gere uma nova chave e certificado para o cluster de banco de dados do Aurora MySQL. Para assinar o certificado de cliente, é necessário ter a chave de autoridade de certificado usada para configurar o SSL no banco de dados primário externo do MySQL.

Para obter mais informações, consulte Creating SSL certificates and keys using openssl na documentação do MySQL.

Você precisa do certificado de autoridade de certificação, a chave do cliente e o certificado do cliente.

2. Conecte-se ao cluster de banco de dados do Aurora MySQL como o usuário primário usando SSL.

   Para obter informações sobre como se conectar ao cluster de banco de dados do Aurora MySQL com SSL, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

3. Execute o procedimento armazenado mysql.rds_import_binlog_ssl_material para importar as informações de SSL para o cluster de banco de dados do Aurora MySQL.

   Para o parâmetro `ssl_material_value`, insira as informações dos arquivos em formato .pem no cluster de banco de dados do Aurora MySQL, na carga JSON correta.

   O exemplo a seguir importa informações SSL em um cluster de banco de dados Aurora MySQL. Em arquivos de formato .pem, o código do corpo geralmente é maior que o código de corpo exibido no exemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
```

```
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Para obter mais informações, consulte mysql.rds_import_binlog_ssl_material e Conexões do TLS com clusters de banco de dados do Aurora MySQL.

> **Note**
>
> Após executar o procedimento, os segredos são armazenados em arquivos. Para apagar os arquivos posteriormente, você pode executar o procedimento armazenado mysql.rds_remove_binlog_ssl_material.

Sincronizar o cluster de banco de dados do Amazon Aurora MySQL com o banco de dados MySQL externo

Você pode sincronizar seu cluster de banco de dados do Amazon Aurora MySQL com o banco de dados de MySQL usando replicação.

Para sincronizar seu cluster de banco de dados do Aurora MySQL com o banco de dados de MySQL usando replicação

1. Certifique-se de que o arquivo /etc/my.cnf do banco de dados MySQL externo tem as entradas relevantes.

   Se a replicação criptografada não for necessária, assegure-se de que o banco de dados MySQL externo é iniciado com logs binários (binlogs) habilitados e SSL desabilitado. A seguir veja as entradas relevantes no arquivo /etc/my.cnf para obter dados não criptografados.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

Se a replicação criptografada for necessária, assegure-se de que o banco de dados MySQL externo é iniciado SSL e logs binários estão habilitados. As entradas no arquivo /etc/my.cnf incluem os locais de arquivos .pem para servidor de banco de dados MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

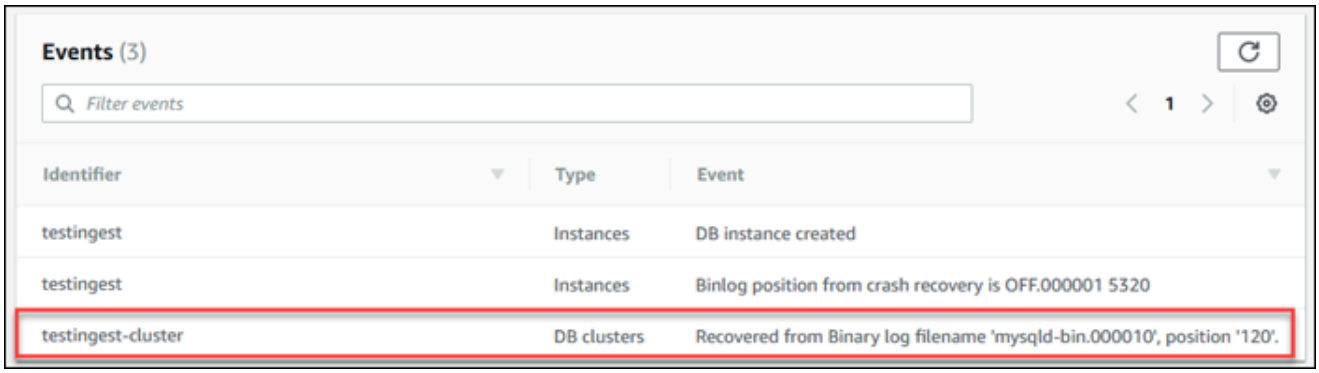
Você pode verificar que o SSL está habilitado com o comando a seguir.

```
mysql> show variables like 'have_ssl';
```

Sua saída deve ser similar à seguinte.

```
+~-~-~-
```

c. Na lista Events (Eventos), anote a posição no evento Recovered from Binary log filename (Recuperado do nome de arquivo de log binário).



Usar o AWS CLI

Você também pode obter o nome e a posição do arquivo de log binário usando o comando describe-events da AWS CLI. O seguinte mostra um exemplo de comando `describe-events`.

```
PROMPT> aws rds describe-events
```

Na saída, identifique o evento que mostra a posição do log binário.

3. Quando conectado ao banco de dados MySQL externo, crie um usuário para ser usado na replicação. Esta conta é usada unicamente para replicação e deve estar restrita ao seu domínio para melhorar a segurança. Veja um exemplo a seguir.

```
mysql> CREATE USER '<user_name>'@'<domain_name>' IDENTIFIED BY '<password>';
```

O usuário requer os privilégios `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concede esses privilégios ao usuário.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
 '<user_name>'@'<domain_name>';
```

Se você precisar usar replicação criptografada, exija conexões SSL para o usuário de replicação. Você pode usar o seguinte comando, por exemplo, para solicitar conexões SSL na conta de usuário *<user_name>*.

```
GRANT USAGE ON *.* TO '<user_name>'@'<domain_name>' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` não estiver incluído, a conexão de replicação pode silenciosamente cair de volta para uma conexão não criptografada.

4. No console do Amazon RDS, adicione o endereço IP do servidor que hospeda o banco de dados MySQL externo ao grupo de segurança da VPC para o cluster de banco de dados do Aurora MySQL. Para obter mais informações sobre como modificar um grupo de segurança da VPC, consulte Grupos de segurança para a VPC no Manual do usuário da Amazon Virtual Private Cloud.

   Você também pode precisar configurar sua rede local para permitir conexões com o endereço IP de seu cluster de banco de dados Aurora MySQL, para que ele possa se comunicar com seu banco de dados MySQL externo. Para encontrar o endereço IP do cluster de banco de dados do Aurora MySQL, use o comando `host`.

   ```
   host <db_cluster_endpoint>
   ```

   O nome do host é o nome de DNS do endpoint do cluster de banco de dados Aurora MySQL.

5. Habilite a replicação de log binário executando o procedimento mysql.rds_reset_external_master (Aurora MySQL versão 2) ou mysql.rds_reset_external_source (Aurora MySQL versão 3) armazenado. Esse procedimento armazenado tem a seguinte sintaxe.

   ```
   CALL mysql.rds_set_external_master (
     host_name
     , host_port
     , replication_user_name
     , replication_user_password
     , mysql_binary_log_file_name
     , mysql_binary_log_file_location
     , ssl_encryption
   );

   CALL mysql.rds_set_external_source (
   ```

```
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Para obter informações sobre os parâmetros, consulte mysql.rds_reset_external_master (Aurora MySQL versão 2) e mysql.rds_reset_external_source (Aurora MySQL versão 3).

Para `mysql_binary_log_file_name` e `mysql_binary_log_file_location`, use a posição no evento Recovered from Binary log filename (Recuperado do nome de arquivo de log binário) que você anotou anteriormente.

Se os dados no cluster de banco de dados do Aurora MySQL não estiverem criptografados, o parâmetro `ssl_encryption` deverá ser definido como `0`. Se os dados são criptografados, o parâmetro `ssl_encryption` deve ser definido como `1`.

O exemplo a seguir executa o procedimento para um cluster de banco de dados de Aurora MySQL que tem dados criptografados.

```
CALL mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);

CALL mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);
```

Esse procedimento armazenado define os parâmetros que o cluster de banco de dados de Aurora MySQL usa para conectar ao banco de dados MySQL externo e ler seu log binário. Se os dados são criptografados, ele também baixa o certificado de autoridade de certificação SSL, certificado e chave de cliente ao disco local.

6. Inicie a replicação de log binário executando o procedimento armazenadomysql.rds_start_replication.

```
CALL mysql.rds_start_replication;
```

7. Monitore a distância do cluster do banco de dados de Aurora MySQL que está atrás do banco de dados MySQL primário de replicação. Para fazer isso, conecte ao cluster de banco de dados do Aurora MySQL e execute o comando a seguir.

```
Aurora MySQL version 2:
SHOW SLAVE STATUS;

Aurora MySQL version 3:
SHOW REPLICA STATUS;
```

   Na saída do comando, o campo `Seconds Behind Master` mostra a distância o cluster de banco de dados do Aurora MySQL está em relação ao MySQL primário. Quando esse valor for `0` (zero), o cluster de banco de dados do Aurora MySQL terá alcançado o primário e você poderá movê-lo para a próxima etapa, a fim de parar a replicação.

8. Conecte-se ao banco de dados primário de replicação do MySQL e pare a replicação. Para isso, execute o procedimento mysql.rds_stop_replication armazenado.

```
CALL mysql.rds_stop_replication;
```

Reduzir o tempo de migração física para o Amazon Aurora MySQL

Você pode fazer as modificações a seguir no banco de dados para acelerar o processo de migração do banco de dados para o Amazon Aurora MySQL.

> **Important**
>
> Realize essas atualizações em uma cópia do banco de dados de produção, e não no banco de dados de produção em si. Em seguida, você poderá fazer backup da cópia e restaurá-la no cluster de banco de dados do Aurora MySQL para evitar interrupções no serviço no banco de dados de produção.

Tipos de tabela não compatíveis

O Aurora MySQL comporta somente o mecanismo InnoDB para tabelas de banco de dados. Se houver tabelas MyISAM no banco de dados, elas deverão ser convertidas antes da migração para o Aurora MySQL. O processo de conversão exige espaço adicional para a conversão de MyISAM para InnoDB durante o procedimento de migração.

Para reduzir as chances de ficar sem espaço ou para agilizar o processo de migração, converta todas as tabelas MyISAM para tabelas InnoDB antes de migrá-las. O tamanho da tabela InnoDB resultante é equivalente ao tamanho exigido pelo Aurora MySQL para aquela tabela. Para converter uma tabela MyISAM para InnoDB, execute o seguinte comando:

```
ALTER TABLE schema.table_name engine=innodb, algorithm=copy;
```

O Aurora MySQL não comporta tabelas ou páginas compactadas (isto é, tabelas criadas com `ROW_FORMAT=COMPRESSED` ou `COMPRESSION = {"zlib"|"lz4"}`).

Para reduzir as chances de ficar sem espaço ou para agilizar o processo de migração, expanda suas tabelas compactadas definindo `ROW_FORMAT` como `DEFAULT`, `COMPACT`, `DYNAMIC` ou `REDUNDANT`. Para páginas compactadas, defina `COMPRESSION="none"`.

Para ter mais informações, consulte InnoDB row formats e InnoDB table and page compression na documentação do MySQL.

Use o seguinte script SQL na sua instância de banco de dados MySQL existente para listar as tabelas no seu banco de dados que são MyISAM ou compactadas.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Aurora MySQL.
-- It must be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.
```

```
-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
```

```
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Contas de usuário com privilégios não compatíveis

As contas de usuário com privilégios não compatíveis com o Aurora MySQL são importadas sem os privilégios incompatíveis. Para ver a lista de privilégios compatíveis, consulte Modelo de privilégios baseados em funções.

É possível executar a consulta SQL a seguir em seu banco de dados de origem para listar as contas de usuário com privilégios incompatíveis.

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

Privilégios dinâmicos no Aurora MySQL versão 3

Os privilégios dinâmicos não são importados. O Aurora MySQL versão 3 comporta os seguintes privilégios dinâmicos.

```
'APPLICATION_PASSWORD_ADMIN',
'CONNECTION_ADMIN',
'REPLICATION_APPLIER',
'ROLE_ADMIN',
'SESSION_VARIABLES_ADMIN',
'SET_USER_ID',
'XA_RECOVER_ADMIN'
```

O script de exemplo a seguir concede os privilégios dinâmicos compatíveis com as contas de usuário no cluster de banco de dados do Aurora MySQL.

```
-- This script finds the user accounts that have Aurora MySQL supported dynamic
 privileges
-- and grants them to corresponding user accounts in the Aurora MySQL DB cluster.

/home/ec2-user/opt/mysql/8.0.26/bin/mysql -uusername -pxxxxx -P8026 -h127.0.0.1 -BNe
 "SELECT
  CONCAT('GRANT ', GRANTS, ' ON *.* TO ', GRANTEE ,';') AS grant_statement
  FROM (select GRANTEE, group_concat(privilege_type) AS GRANTS FROM
 information_schema.user_privileges
      WHERE privilege_type IN (
        'APPLICATION_PASSWORD_ADMIN',
        'CONNECTION_ADMIN',
        'REPLICATION_APPLIER',
        'ROLE_ADMIN',
        'SESSION_VARIABLES_ADMIN',
        'SET_USER_ID',
        'XA_RECOVER_ADMIN')
      AND GRANTEE NOT IN (\"'mysql.session'@'localhost'\",
\"'mysql.infoschema'@'localhost'\",\"'mysql.sys'@'localhost'\") GROUP BY GRANTEE)
      AS PRIVGRANTS; " | /home/ec2-user/opt/mysql/8.0.26/bin/mysql -u master_username -
p master_password -h DB_cluster_endpoint
```

Objetos armazenados com ‘rdsadmin'@'localhost’ como definidor

Funções, procedimentos, visualizações, eventos e acionadores com `'rdsadmin'@'localhost'` como definidor não são importados.

Você pode usar o script SQL a seguir, no banco de dados MySQL de origem, a fim de listar os objetos armazenados que têm o definidor incompatível.

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
```

```
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
FROM
    information_schema.events
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

Migração lógica do MySQL para o Amazon Aurora MySQL usando o mysqldump

Como o Amazon Aurora MySQL é um banco de dados compatível com MySQL, você pode usar o utilitário `mysqldump` para copiar dados do banco de dados MySQL ou o utilitário `mariadb-dump` para copar os dados do banco de dados MariaDB em um cluster de banco de dados existente do Aurora MySQL.

Para ver uma discussão sobre como fazer isso com bancos de dados MySQL ou MariaDB que são muito grandes, consulte os seguintes tópicos no Guia do usuário do Amazon Relational Database Service:

- MySQL: importar dados para um banco de dados Amazon RDS para MySQL com tempo de inatividade reduzido.
- MariaDB: importar dados para um banco de dados Amazon RDS para MariaDB com tempo de inatividade reduzido.

Em relação a bancos de dados MySQL ou MariaDB que têm volumes menores de dados, consulte os seguintes tópicos no Guia do usuário do Amazon Relational Database Service:

- MySQL: importar dados de um banco de dados externo MySQL para uma instância de banco de dados do Amazon RDS para MySQL.
- MariaDB: importar dados de um banco de dados externo MariaDB para uma instância de banco de dados do Amazon RDS para MariaDB.

Migração de dados de um instância de banco de dados RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL.

É possível migrar (copiar) dados de um cluster de banco de dados do Amazon Aurora MySQL para uma instância de banco de dados do RDS para MySQL.

Tópicos

- Migrar de um snapshot do RDS for MySQL para o Aurora
- Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora

> **Note**
>
> Como o Amazon Aurora MySQL é compatível com MySQL, você pode migrar dados do seu banco de dados MySQL configurando uma replicação entre o banco de dados MySQL e um cluster de banco de dados do Amazon Aurora MySQL. Para obter mais informações, consulte Replicação com o Amazon Aurora.

Migrar de um snapshot do RDS for MySQL para o Aurora

Você pode migrar um snapshot de banco de dados de uma instância de banco de dados RDS for MySQL para criar um cluster de banco de dados Aurora MySQL. O novo cluster de banco de dados do Aurora MySQL é preenchido com os dados da instância de banco de dados RDS for MySQL original. O snapshot de banco de dados precisa ter sido criado de uma instância de banco de dados do Amazon RDS que esteja executando uma versão do MySQL compatível com o Aurora MySQL.

É possível migrar um snapshot de banco de dados manual ou automático. Após a criação do cluster de banco de dados, você pode criar réplicas opcionais do Aurora.

> **Note**
>
> Você também pode migrar uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Aurora MySQL criando uma réplica de leitura do Aurora da sua instância de banco de dados do RDS para MySQL de origem. Para obter mais informações, consulte Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora.

Não é possível migrar de algumas versões anteriores a 8.0 do MySQL para o Aurora MySQL 3.05, incluindo 8.0.11, 8.0.13 e 8.0.15. Recomendamos que você atualize para a versão 8.0.28 do MySQL antes da migração.

Veja as etapas gerais que você deve seguir:

1. Determine a quantidade de espaço para provisionar ao seu cluster de banco de dados Aurora MySQL. Para obter mais informações, consulte De quanto espaço eu preciso?
2. Use o console para criar o snapshot na região da AWS em que se encontra a instância MySQL do Amazon RDS. Para obter informações sobre a criação de um snapshot de banco de dados, consulte Criar um snapshot de banco de dados.
3. Se o snapshot do banco de dados não estiver na mesma região da AWS que o cluster de banco de dados, use o console do Amazon RDS para copiar o snapshot do banco de dados para essa região da AWS. Para obter informações sobre a cópia de um snapshot de banco de dados, consulte Cópia de um snapshot de banco de dados.
4. Use o console para migrar o snapshot de banco de dados e criar um cluster de banco de dados Aurora MySQL com os mesmos bancos de dados que a instância de banco de dados MySQL original.

⚠ Warning

O Amazon RDS limita cada conta da AWS a uma cópia de snapshot para cada região da AWS por vez.

De quanto espaço eu preciso?

Ao migrar um snapshot de uma instância de banco de dados MySQL para um cluster de banco de dados do Aurora MySQL, o Aurora usa um volume do Amazon Elastic Block Store (Amazon EBS) para formatar os dados do snapshot antes de migrá-los. Em alguns casos, um espaço adicional é necessário para formatar os dados para migração.

As tabelas que não sejam MyISAM e não estejam compactadas podem ter até 16 TB de tamanho. Se você tiver tabelas MyISAM, o Aurora deverá usar espaço adicional no volume para converter as tabelas e deixá-las compatíveis com o Aurora MySQL. Se você compactar as tabelas, o Aurora deverá usar espaço adicional no volume para expandi-las antes de armazená-las no volume de

cluster do Aurora. Devido a esse requisito de espaço adicional, você deve garantir que nenhuma das tabelas MyISAM e das tabelas compactadas sendo migradas da sua instância de banco de dados MySQL exceda 8 TB de tamanho.

Reduzir a quantidade de espaço necessário para migrar dados para o Amazon Aurora MySQL

Talvez você queira modificar o esquema de banco de dados antes de migrá-lo para o Amazon Aurora. Essa modificação pode ser útil nos seguintes casos:

- Você deseja agilizar o processo de migração.
- Você não sabe quanto espaço deve ser provisionado.
- Você tentou migrar os dados e a migração falhou devido à falta de espaço provisionado.

É possível fazer as seguintes alterações para melhorar o processo de migração de um banco de dados para o Amazon Aurora.

> **Important**
>
> Lembre-se de realizar essas atualizações em uma nova instância de banco de dados restaurada do snapshot de um banco de dados de produção, em vez de em uma instância de produção. Em seguida, você poderá migrar os dados do snapshot da sua nova instância de banco de dados para o cluster de banco de dados do Aurora a fim de evitar interrupções no serviço do seu banco de dados de produção.

Tipo de tabela	Limitação ou diretriz
Tabelas MyISAM	O Aurora MySQL oferece suporte somente a tabelas InnoDB. Se houver tabelas MyISAM no seu banco de dados, elas deverão ser convertidas antes da migração para o Aurora MySQL. O processo de conversão exige espaço adicional para a conversão de MyISAM para InnoDB durante o procedimento de migração. Para reduzir as chances de ficar sem espaço ou para agilizar o processo de migração, converta todas as tabelas MyISAM para tabelas InnoDB antes de migrá-las. O tamanho da tabela InnoDB resultante é equivalente ao tamanho exigido pelo Aurora MySQL

Tipo de tabela	Limitação ou diretriz
	para aquela tabela. Para converter uma tabela MyISAM para InnoDB, execute o seguinte comando: `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Tabelas compactadas	O Aurora MySQL não oferece suporte a tabelas compactadas (ou seja, tabelas criadas com `ROW_FORMAT=COMPRESSED`). Para reduzir as chances de ficar sem espaço ou para agilizar o processo de migração, expanda suas tabelas compactadas definindo `ROW_FORMAT` como `DEFAULT`, `COMPACT`, `DYNAMIC` ou `REDUNDANT`. Para obter mais informações, consulte Formatos de linha do InnoDB na documentação do MySQL.

Use o seguinte script SQL na sua instância de banco de dados MySQL existente para listar as tabelas no seu banco de dados que são MyISAM ou compactadas.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Amazon Aurora.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;
```

```
-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

O script gera uma saída semelhante à saída do seguinte exemplo. O exemplo mostra duas tabelas que devem ser convertidas de MyISAM para InnoDB. A saída também inclui o tamanho aproximado de cada tabela em megabytes (MB).

```
+---------------------------------+------------------+
| ==> MyISAM or Compressed Tables | Approx size (MB) |
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
```

```
2 rows in set (0.01 sec)
```

Migrar um snapshot de banco de dados do RDS para MySQL para um cluster de banco de dados do Aurora MySQL

Você pode migrar um snapshot de banco de dados de uma instância de banco de dados do RDS para MySQL para criar um cluster de banco de dados do Aurora MySQL usando o Console de gerenciamento da AWS ou a AWS CLI. O novo cluster de banco de dados do Aurora MySQL é preenchido com os dados da instância de banco de dados RDS for MySQL original. Para obter informações sobre a criação de um snapshot de banco de dados, consulte Criar um snapshot de banco de dados.

Se o snapshot do banco de dados não estiver na região da AWS em que você deseja colocar os seus dados, copie o snapshot do banco de dados para essa região da AWS. Para obter informações sobre a cópia de um snapshot de banco de dados, consulte Cópia de um snapshot de banco de dados.

Console

Quando você migra o snapshot de banco de dados usando o Console de gerenciamento da AWS, o console realiza as ações necessárias para criar somente o cluster de banco de dados.

Também é possível determinar que o seu novo cluster de banco de dados do Aurora MySQL seja criptografado em repouso usando uma AWS KMS key.

Para migrar um snapshot de banco de dados MySQL usando o Console de gerenciamento da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Comece a migração a partir da instância de banco de dados do MySQL ou do snapshot:

   Para iniciar a migração a partir da instância de banco de dados:

   1. No painel de navegação, selecione Databases (Bancos de dados) e selecione a instância de banco de dados MySQL.
   2. Em Actions (Ações), escolha Migrate latest snapshot (Migrar o último snapshot).

   Para iniciar a migração a partir do snapshot:

   1. Escolha Snapshots.

2. Na página Snapshots, escolha o snapshot que você deseja migrar para um cluster de banco de dados do Aurora MySQL.
3. Escolha Snapshot Actions (Ações do snapshot) e, em seguida, escolha Migrate Snapshot (Migrar snapshot).

A página Migrate Database (Migrar banco de dados) é exibida.

3. Defina os seguintes valores na página Migrate Database:

   - Migrate to DB Engine: Selecione `aurora`.
   - DB Engine Version (Versão do mecanismo de banco de dados): selecione a versão do mecanismo de banco de dados para o cluster de banco de dados do Aurora MySQL.
   - DB Instance Class (Classe de instância de banco de dados): selecione uma categoria de instância de banco de dados que tenha o armazenamento e a capacidade necessários para seu banco de dados, por exemplo `db.r3.large`. Os volumes de cluster do Aurora crescem automaticamente à medida em que aumenta a quantidade de dados em seu banco de dados. Um volume de cluster do Aurora pode aumentar até o tamanho máximo de 128 tebibytes (TiB). Assim, só será necessário selecionar uma classe de instância de banco de dados que atenda aos requisitos atuais de armazenamento. Para obter mais informações, consulte Visão geral do armazenamento do Amazon Aurora.
   - Identificador da instância do banco de dados: digite um nome para o cluster de banco de dados exclusivo à sua conta na região da AWS selecionada. Esse identificador é usado em endereços de endpoint para as instâncias no cluster de banco de dados. É possível optar por adicionar informações ao nome, como incluir a região da AWS e o mecanismo de banco de dados selecionados, por exemplo, **`aurora-cluster1`**.

     O DB instance identifier tem as seguintes restrições:

     - Deve conter de 1 a 63 caracteres alfanuméricos ou hífens.
     - O primeiro caractere deve ser uma letra.
     - Não pode terminar com um hífen ou conter dois hifens consecutivos.
     - Deve ser exclusivo para todas as instâncias de banco de dados por conta do AWS, por região da AWS.
   - Virtual Private Cloud (VPC) (Nuvem privada virtual): se você tiver uma VPC existente, poderá usá-la com o seu cluster de banco de dados do Aurora MySQL selecionando o identificador de VPC, por exemplo `vpc-a464d1c1`. Para obter informações sobre como criar uma VPC,

consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

De outro modo, é possível optar para que o Aurora crie uma VPC para você selecionando Create a new VPC (Criar uma VPC).

- DB subnet group (Grupo de sub-redes do banco de dados): se você tiver um grupo de sub-redes existente, poderá usá-lo com o cluster de banco de dados MySQL do Aurora selecionando o identificador do grupo de sub-redes, por exemplo, `gs-subnet-group1`.

  De outro modo, é possível optar para que o Aurora crie um grupo de sub-redes para você selecionando Create a new subnet group (Criar um grupo de sub-redes).

- Public accessibility: selecione No para especificar que as instâncias no seu cluster de banco de dados só podem ser acessadas por recursos dentro da sua VPC. Selecione Yes para especificar que as instâncias no seu cluster de banco de dados podem ser acessadas por recursos na rede pública. O padrão é Yes (Sim).

  > **Note**
  >
  > O cluster de banco de dados de produção talvez não precise estar em uma sub-rede pública, porque apenas os seus servidores de aplicativo precisarão acessá-lo. Se o cluster de banco de dados não precisar estar em uma sub-rede pública, defina Publicly Accessible como No.

- Availability Zone (Zona de disponibilidade): selecione a zona de disponibilidade para hospedar a instância primária do seu cluster de banco de dados Aurora MySQL. Para fazer com que o Aurora selecione uma zona de disponibilidade para você, escolha No Preference (Sem preferência).
- Database Port (Porta de banco de dados): digite a porta padrão para ser usada ao se conectar a instâncias no cluster de banco de dados do Aurora MySQL. O padrão é `3306`.

  > **Note**
  >
  > Pode ser que haja um firewall corporativo que impeça o acesso a portas padrão, como a porta padrão do MySQL, 3306. Nesse caso, forneça um valor de porta permitido pelo seu firewall corporativo. Lembre-se desse valor de porta mais tarde ao se conectar ao cluster de banco de dados do Aurora MySQL.

- Encryption (Criptografia): selecione Enable Encryption (Habilitar criptografia) para que o novo cluster de banco de dados do Aurora MySQL seja criptografado em repouso. Se você selecionar Enable Encryption (Habilitar criptografia), deverá escolher uma chave do KMS como o valor de AWS KMS key.

  Se o seu snapshot de banco de dados não for criptografado, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso.

  Se o seu snapshot de banco de dados for criptografado, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso usando-a. Você pode especificar a chave de criptografia usada pelo snapshot de banco de dados ou por uma chave diferente. Você não pode criar um cluster de banco de dados não criptografado de um snapshot de banco de dados criptografado.

- Auto Minor Version Upgrade (Atualização automática de versão secundária): essa configuração não se aplica aos clusters de banco de dados do Aurora MySQL.

  Para obter mais informações sobre atualizações de mecanismos para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

4. Selecione Migrate para migrar o snapshot de banco de dados.
5. Selecione Instances e o ícone de seta para mostrar os detalhes do cluster de banco de dados e monitorar o andamento da migração. Na página de detalhes, você vê o endpoint do cluster usado para se conectar à instância primária do cluster de banco de dados. Para obter mais informações sobre a conexão com um cluster de banco de dados do Aurora MySQL, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

AWS CLI

Você pode criar um cluster de banco de dados do Aurora a partir de um snapshot de banco de dados de uma instância de banco de dados RDS for MySQL usando o `restore-db-cluster-from-snapshot` comando com os seguintes parâmetros:

- `--db-cluster-identifier`: o nome do cluster de banco de dados a ser criado.
- `--engine aurora-mysql`: para um cluster de banco de dados compatível com o MySQL 5.7 ou 8.0.
- `--kms-key-id`: a AWS KMS key para criptografar opcionalmente o cluster de banco de dados, de acordo com a criptografia do seu snapshot de banco de dados.

- Se o seu snapshot de banco de dados não for criptografado, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso. Caso contrário, seu cluster de banco de dados não será criptografado.
- Se o seu snapshot de banco de dados for criptografado, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso usando-a. Caso contrário, seu cluster de banco de dados será criptografado em repouso usando a chave de criptografia do snapshot de banco de dados.

> **Note**
>
> Você não pode criar um cluster de banco de dados não criptografado de um snapshot de banco de dados criptografado.

- `--snapshot-identifier` – O nome do recurso da Amazon (ARN) do snapshot do banco de dados a ser migrado. Para obter mais informações sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

Quando você migra o snapshot de banco de dados usando o comando `RestoreDBClusterFromSnapshot`, o comando cria o cluster de banco de dados e a instância primária.

Neste exemplo, você cria um cluster de banco de dados compatível com o MySQL 5.7, denominado *`mydbcluster`*, de um snapshot de banco de dados com um ARN definido como *`mydbsnapshotARN`*.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Neste exemplo, você cria um cluster de banco de dados compatível com o MySQL 5.7, denominado *mydbcluster*, de um snapshot de banco de dados com um ARN definido como *mydbsnapshotARN*.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora

O Aurora usa a funcionalidade de replicação de log binário dos mecanismos de banco de dados MySQL para criar um tipo especial de cluster de banco de dados, chamado réplica de leitura do Aurora, para uma instância de banco de dados do RDS para MySQL. As atualizações feitas na instância de banco de dados do RDS para MySQL de origem são replicadas de forma assíncrona na réplica de leitura do Aurora.

Recomendamos usar essa funcionalidade para migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Aurora MySQL criando uma réplica de leitura do Aurora da instância de banco de dados do RDS para MySQL de origem. Quando o atraso da réplica entre a instância de banco de dados do RDS para MySQL e a réplica de leitura do Aurora é 0, você pode direcionar aplicações clientes para a réplica de leitura do Aurora e interromper a replicação, a fim de que a réplica de leitura do Aurora se torne um cluster de banco de dados autônomo do Aurora MySQL. Prepare-se. A migração pode demorar um pouco. Estima-se várias horas por tebibyte (TiB) de dados.

Para obter uma lista de regiões onde o Aurora está disponível, consulte Amazon Aurora na Referência geral da AWS.

Quando você cria uma réplica de leitura do Aurora usando uma instância de banco de dados do RDS para MySQL, o Amazon RDS cria um snapshot de banco de dados da instância de banco de dados do RDS para MySQL de origem (privado para o Amazon RDS sem qualquer custo). Em seguida, o Amazon RDS migra os dados do snapshot do banco de dados para a réplica de leitura do Aurora. Depois que os dados do snapshot de banco de dados são migrados para o novo cluster de banco de dados do Aurora MySQL, o Amazon RDS inicia a replicação entre a instância de banco de dados do RDS para MySQL e o cluster de banco de dados do Aurora MySQL. Caso a instância de banco de dados do RDS para MySQL contenha tabelas que usam mecanismos de armazenamento diferentes do InnoDB ou que usam formato compacto de linha, você pode agilizar o processo de criação de uma réplica de leitura do Aurora. Para isso, basta alterar essas tabelas para que usem o mecanismo de armazenamento InnoDB e o formato dinâmico de linha antes de criar a réplica de leitura do Aurora. Para ter mais informações sobre o processo de cópia de um snapshot de banco de dados MySQL para um cluster de banco de dados do Aurora MySQL, consulte Migração de dados de um instância de banco de dados RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL..

Você só pode ter uma réplica de leitura do Aurora para uma instância de banco de dados do MySQL.

> **Note**
>
> Podem ocorrer problemas de replicação devido a diferenças de recurso entre o Aurora MySQL e a versão do mecanismo de banco de dados do MySQL da instância de banco de dados do RDS para MySQL, que é a original da replicação. Se você encontrar um erro, tente obter ajuda no fórum da comunidade do Amazon RDS ou entre em contato com o AWS Support.
> Não será possível criar uma réplica de leitura do Aurora se a instância de banco de dados do RDS para MySQL já for a origem de uma réplica de leitura entre regiões.
> Não é possível migrar de algumas versões anteriores a 8.0 do RDS para MySQL para o Aurora MySQL 3.05, incluindo 8.0.11, 8.0.13 e 8.0.15. Recomendamos que você atualize para a versão 8.0.28 do RDS pra MySQL antes da migração.

Para ter mais informações sobre réplicas de leitura do MySQL, consulte Como trabalhar com réplicas de leitura do MariaDB, do MySQL e de instâncias de banco de dados PostgreSQL.

Criar uma réplica de leitura do Aurora

Você pode criar uma réplica de leitura do Aurora para uma instância de banco de dados do RDS para MySQL usando o console, a AWS CLI ou a API do RDS.

Console

Como criar uma réplica de leitura do Aurora usando uma instância de banco de dados de origem do RDS para MySQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha a instância de banco de dados MySQL que você deseja usar como a origem da réplica de leitura do Aurora.
4. Em Actions (Ações), selecione Create Aurora read replica (Criar réplica de leitura do Aurora).
5. Escolha as especificações do cluster de banco de dados que deseja usar para a réplica de leitura do Aurora, como descrito na tabela a seguir.

Opção	Descrição
Classe da instância de banco de dados	Escolha uma classe de instância de banco de dados que defina os requisitos de processamento e memória para a instância principal no cluster de banco de dados. Para ter mais informações sobre as opções de classe de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.
implantação multi-AZ	Escolha Criar réplica em diferentes zonas para criar uma réplica em espera do novo cluster de banco de dados em outra Zona de disponibilidade na região de destino da AWS para suporte a failover. Para ter mais informações sobre várias Zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.

Opção	Descrição
Identificador de instância de banco de dados	Digite um nome para a instância primária no seu cluster de banco de dados de réplica de leitura do Aurora. Esse identificador é usado no endereço do endpoint da instância principal do novo cluster de banco de dados. O DB instance identifier tem as seguintes restrições: • Deve conter de 1 a 63 caracteres alfanuméricos ou hífens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todas as instâncias de banco de dados para cada conta da AWS por região da AWS. Como o cluster de banco de dados da réplica de leitura do Aurora é criado de um snapshot da instância do banco de dados de origem, o nome de usuário principal e a senha principal para a réplica de leitura do Aurora são os mesmos nome e senha principais da instância do banco de dados de origem.
Nuvem privada virtual (VPC)	Selecione o VPC que hospeda o cluster de banco de dados. Selecione Create a new VPC (Criar uma VPC) para fazer com que o Aurora crie uma VPC para você. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.

Opção	Descrição
DB subnet group (Grupo de subredes do banco de dados)	Selecione o grupo de sub-rede de banco de dados para usar para o cluster de banco de dados. Selecione Create new DB subnet group (Criar grupo de sub-redes de banco de dados) para fazer com que o Aurora crie um grupo de sub-redes de banco de dados para você. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.
Public accessibility	Selecione `Yes` para dar um endereço IP público ao cluster de banco de dados; do contrário, selecione `No`. As instâncias no seu cluster de banco de dados podem ser uma combinação de instâncias de banco de dados públicas e privadas. Para ter mais informações sobre como ocultar instâncias do acesso público, consulte Ocultar um cluster de banco de dados em uma VPC da Internet.
Availability zone	Determine se você deseja especificar uma Zona de disponibilidade específica. Para ter mais informações sobre zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.
VPC security group (firewall) [Grupo de segurança da VPC (firewall)]	Selecione Create new VPC security group (Criar novo grupo de segurança de VPC) para fazer com que o Aurora crie um grupo de segurança de VPC para você. Selecione Select existing VPC security groups (Selecionar grupos de segurança de VPC existentes) para especificar um ou mais grupos de segurança de VPC e proteger o acesso de rede ao cluster de banco de dados. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.

Opção	Descrição
Database port	Especifique a porta que os aplicativos e os utilitários usam para acessar o banco de dados. Os clusters de banco de dados Aurora MySQL são definidos para a porta padrão do MySQL, 3306. Em algumas empresas, os firewalls bloqueiam conexões a esta porta. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o novo cluster de banco de dados.
Grupo de parâmetros de banco de dados	Selecione um grupo de parâmetros de banco de dados para o cluster de banco de dados do Aurora MySQL. O Aurora conta com um grupo de parâmetro s de banco de dados padrão que você pode usar, ou é possível criar seu próprio grupo de parâmetros de banco de dados. Para ter mais informações sobre os parameter groups de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.
Grupo de parâmetros do cluster de banco de dados	Selecione um grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados do Aurora MySQL. O Aurora conta com um grupo de parâmetros de cluster de banco de dados que você pode usar, ou é possível criar seu próprio grupo de parâmetros. Para ter mais informações sobre os parameter groups do cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Opção	Descrição
Criptografia	Escolha Disable encryption se você não quiser que seu novo cluster de banco de dados Aurora seja criptografado. Escolha Enable encryption para que o novo cluster de banco de dados Aurora seja criptografado em repouso. Se você selecionar Enable encryption (Habilitar criptografia), deverá escolher uma chave do KMS como o valor de AWS KMS key. Se a sua instância de banco de dados MySQL não for criptografada, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso. Se a sua instância de banco de dados MySQL for criptografada, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso usando-a. Você pode especificar a chave de criptografia usada pela instância de banco de dados MySQL ou por uma chave diferente. Você não pode criar um cluster de banco de dados não criptografado de uma instância de banco de dados MySQL criptografada.
Priority	Escolha uma prioridade de failover para o cluster de banco de dados. Se você não selecionar um valor, o padrão será tier-1. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Para ter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

Opção	Descrição
Backup retention period (Período de retenção de backup	Selecione o tempo, de 1 a 35 dias, que o Aurora retém cópias de backup do banco de dados. As cópias de backup podem ser usadas para restaurações point-in-time (PITR) do banco de dados, contabilizando até os segundos.
Monitoramento avançado	Escolha Enable enhanced monitoring (Habilitar monitoramento avançado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para ter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Monitoring Role (Monitoramento de perfis	Disponível apenas quando Enhanced Monitoring estiver definido como Enable enhanced monitorin g. Escolha a função do IAM que você criou para permitir que o Aurora se comunique com o Amazon CloudWatch Logs para você. Ou escolha Default (Padrão) para que o Aurora crie uma função para você chamada `rds-monitoring-role` . Para ter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Granularity	Disponível apenas quando Enhanced Monitoring estiver definido como Enable enhanced monitoring. Defina o intervalo, em segundos, em que as métricas são coletadas para o seu cluster de banco de dados.
Atualização da versão secundária automática	Essa configuração não é aplicável aos clusters de banco de dados do Aurora MySQL. Para ter mais informações sobre atualizações de mecanismos para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

Opção	Descrição
Janela de manutenção	Selecione Select window e especifique o período semanal durante o qual a manutenção do sistema pode ser realizada. Ou selecione No preference (Sem preferência) para que o Aurora atribua um período aleatoriamente.

6. Escolha Create read replica (Criar réplica de leitura).

AWS CLI

Para criar uma réplica de leitura do Aurora usando uma instância de banco de dados do MySQL de origem, use os comandos da `create-db-cluster` e `create-db-instance` da AWS CLI para criar um cluster de banco de dados do Aurora MySQL. Quando você chamar o comando `create-db-cluster`, inclua o parâmetro `--replication-source-identifier` para identificar o Amazon Resource Name (ARN – Nome de recurso da Amazon) da instância do banco de dados MySQL de origem. Para ter mais informações sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

Não especifique o nome de usuário principal, a senha principal ou o nome do banco de dados já que a réplica de leitura do Aurora usa o mesmo nome de usuário principal, senha principal e nome de banco de dados que a instância de banco de dados do MySQL de origem.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Se você usar o console para criar uma réplica de leitura do Aurora, o Aurora criará automaticamente a instância primária para a sua réplica de leitura do cluster de banco de dados do Aurora. Se você usar a AWS CLI para criar uma réplica de leitura do Aurora, deverá criar explicitamente a instância primária para o seu cluster de banco de dados. A instância principal é a primeira instância criada em um cluster de banco de dados.

Você pode criar uma instância primária para seu cluster de banco de dados usando o comando `create-db-instance` da AWS CLI com os parâmetros a seguir.

- `--db-cluster-identifier`

  O nome de seu cluster de banco de dados.
- `--db-instance-class`

  O nome da classe da instância de banco de dados a ser usada para sua instância primária.
- `--db-instance-identifier`

  O nome da sua instância primária.
- `--engine aurora`

Neste exemplo, você cria uma instância primária denominada *myreadreplicainstance* para o cluster de banco de dados denominado *myreadreplicacluster* usando a classe da instância de banco de dados especificada em *myinstanceclass*.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass \
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

Para Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier myreadreplicacluster ^
    --db-instance-class myinstanceclass ^
```

```
    --db-instance-identifier myreadreplicainstance ^
    --engine aurora
```

API do RDS

Para criar uma réplica de leitura do Aurora de uma instância de banco de dados do RDS para MySQL de origem, use os comandos `CreateDBCluster` e `CreateDBInstance` da API do Amazon RDS para criar um cluster de banco de dados do Aurora e uma instância primária. Não especifique o nome de usuário principal, a senha principal ou o nome do banco de dados, pois a réplica de leitura do Aurora usa o mesmo nome de usuário principal, senha principal e nome de banco de dados que a instância de banco de dados do RDS para MySQL de origem.

Você pode criar um cluster de banco de dados do Aurora para uma réplica de leitura do Aurora com base em uma instância de banco de dados do RDS para MySQL de origem usando o comando `CreateDBCluster` da API do Amazon RDS com os seguintes parâmetros:

- `DBClusterIdentifier`

  O nome do cluster de banco de dados a ser criado.

- `DBSubnetGroupName`

  O nome do grupo de sub-redes de banco de dados a ser associado a esse cluster de banco de dados.

- `Engine=aurora`
- `KmsKeyId`

  A AWS KMS key a ser usada para criptografar o cluster de banco de dados opcionalmente, de acordo com a criptografia da sua instância de banco de dados do MySQL.

  - Se a sua instância de banco de dados MySQL não for criptografada, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso. Caso contrário, seu cluster de banco de dados será criptografado em repouso usando a chave de criptografia padrão da sua conta.
  - Se a sua instância de banco de dados MySQL for criptografada, especifique uma chave de criptografia para criptografar seu cluster de banco de dados em repouso usando-a. Caso contrário, seu cluster de banco de dados será criptografado em repouso usando a chave de criptografia da instância de banco de dados MySQL.

> **Note**
>
> Você não pode criar um cluster de banco de dados não criptografado de uma instância de banco de dados MySQL criptografada.

- `ReplicationSourceIdentifier`

  O Nome de recurso da Amazon (ARN) para a instância de banco de dados MySQL de origem. Para ter mais informações sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

- `VpcSecurityGroupIds`

  A lista de security groups de VPC do EC2 a ser associada a esse cluster de banco de dados.

Neste exemplo, você cria um cluster de banco de dados denominado *myreadreplicacluster* de uma instância de banco de dados MySQL de origem com um ARN definido como *mysqlprimaryARN* associado a um grupo do sub-redes de banco de dados denominado *mysubnetgroup*, e um grupo de segurança da VPC denominado *mysecuritygroup*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlprimaryARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

Se você usar o console para criar uma réplica de leitura do Aurora, o Aurora criará automaticamente a instância primária para a sua réplica de leitura do cluster de banco de dados do Aurora. Se você

usar a AWS CLI para criar uma réplica de leitura do Aurora, deverá criar explicitamente a instância primária para o seu cluster de banco de dados. A instância principal é a primeira instância criada em um cluster de banco de dados.

Você pode criar uma instância primária para seu cluster de banco de dados usando o comando `CreateDBInstance` da API do Amazon RDS com os parâmetros a seguir:

- `DBClusterIdentifier`

  O nome de seu cluster de banco de dados.

- `DBInstanceClass`

  O nome da classe da instância de banco de dados a ser usada para sua instância primária.

- `DBInstanceIdentifier`

  O nome da sua instância primária.

- `Engine=aurora`

Neste exemplo, você cria uma instância primária denominada *`myreadreplicainstance`* para o cluster de banco de dados denominado *`myreadreplicacluster`* usando a classe da instância de banco de dados especificada em *`myinstanceclass`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
    &DBClusterIdentifier=myreadreplicacluster
    &DBInstanceClass=myinstanceclass
    &DBInstanceIdentifier=myreadreplicainstance
    &Engine=aurora
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
    &X-Amz-Date=20140424T194844Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

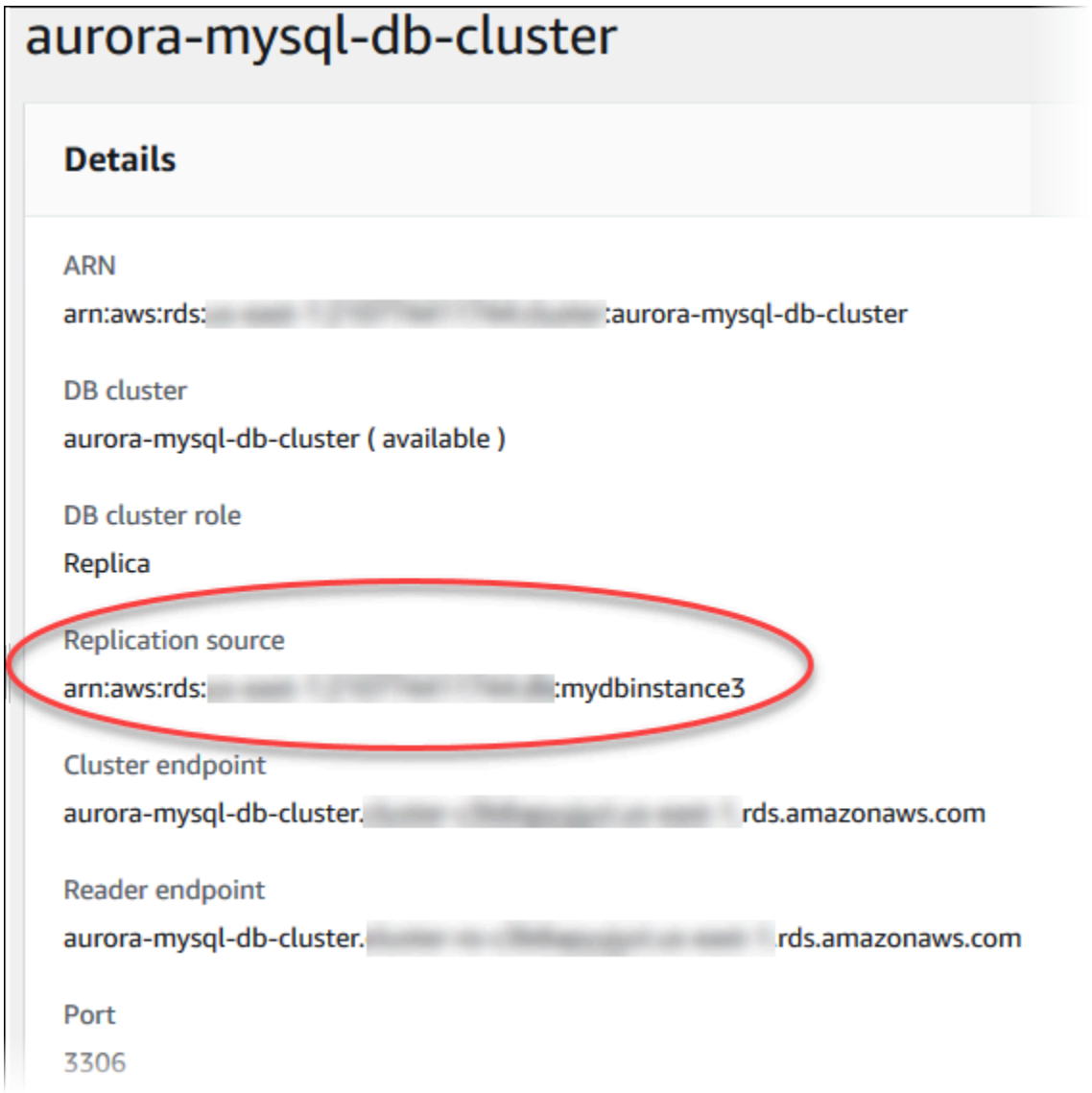
Visualizar uma réplica de leitura do Aurora

Você pode visualizar as relações de replicação do MySQL para o Aurora MySQL de seus clusters de banco de dados do Aurora MySQL usando o Console de gerenciamento da AWS ou a AWS CLI.

Console

Como visualizar a instância de banco de dados do MySQL para uma réplica de leitura do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados da réplica de leitura do Aurora para exibir seus detalhes. As informações da instância do banco de dados MySQL primária estará no campo Replication source (Fonte de replicação).



AWS CLI

Para visualizar as relações de replicação do MySQL com o Aurora MySQL para clusters de banco de dados do Aurora MySQL utilizando a AWS CLI, use os comandos `describe-db-clusters` e `describe-db-instances`.

Para determinar qual instância de banco de dados MySQL é a primária, use o `describe-db-clusters` e especifique o identificador do cluster de réplica de leitura do Aurora para a opção `--db-cluster-identifier`. Consulte o elemento `ReplicationSourceIdentifier` na saída para o ARN da instância de banco de dados que é a primária da replicação.

Para determinar qual cluster de banco de dados é a réplica de leitura do Aurora, use o `describe-db-instances` e especifique o identificador da instância de banco de dados MySQL para a opção `--db-instance-identifier`. Consulte o elemento `ReadReplicaDBClusterIdentifiers` na saída para o identificador de cluster de banco de dados da réplica de leitura do Aurora.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlprimary
```

Para Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlprimary
```

Promover uma réplica de leitura do Aurora

Depois que a migração for concluída, você poderá promover a réplica de leitura do Aurora para um cluster de banco de dados autônomo usando o Console de gerenciamento da AWS ou a AWS CLI.

Depois, você poderá direcionar suas aplicações cliente ao endpoint para a réplica de leitura do Aurora. Para ter mais informações sobre os endpoints do Aurora, consulte Conexões de endpoints do Amazon Aurora. A promoção deve ser bem rápida. É possível fazer operações de leitura e gravação na réplica de leitura do Aurora durante a promoção. No entanto, você não pode excluir a instância primária do banco de dados MySQL ou desvincular a instância de banco de dados e a réplica de leitura do Aurora neste momento.

Antes de promover a sua réplica de leitura do Aurora interrompa quaisquer transações de gravação na instância do banco de dados MySQL de origem. Em seguida, aguarde que o atraso da réplica de leitura do Aurora chegue a 0. É possível visualizar o atraso da réplica para uma réplica de leitura do Aurora chamando o comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA`

`STATUS` (Aurora MySQL versão 3) na sua réplica de leitura do Aurora. Verifique o valor Seconds behind master (Segundos atrás do mestre).

Você pode iniciar a gravação na réplica de leitura do Aurora depois que as transações de gravação para o primário tiverem parado e o atraso da réplica for 0. Se você gravar na réplica de leitura do Aurora antes disso e alterar as tabelas que também estão sendo modificadas no MySQL primário, você se arriscará a interromper a replicação para o Aurora. Se isso acontecer, você deve excluir e recriar sua réplica de leitura do Aurora.

Console

Como promover uma réplica de leitura do Aurora a um cluster de bancos de dados Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados para a réplica de leitura do Aurora.
4. Em Actions (Ações), selecione Promote (Promover).
5. Escolha Promote read replica (Promover réplica de leitura).

Depois de promover, confirme se a promoção foi concluída usando o procedimento a seguir.

Para confirmar que a réplica de leitura do Aurora foi promovida

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Events.
3. Na página Eventos, verifique se há um evento do `Promoted Read Replica cluster to a stand-alone database cluster` para o cluster que você promoveu.

Depois que promoção estiver concluída, a instância do banco de dados MySQL primária e a réplica de leitura do Aurora serão desvinculados e, se desejar, você poderá excluir a instância de banco de dados com segurança.

AWS CLI

Para promover uma réplica de leitura do Aurora a um cluster de banco de dados autônomo, use o comando `promote-read-replica-db-cluster` da AWS CLI.

Example

Para Linux, macOS ou Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Para Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Como gerenciar o Amazon Aurora MySQL

As seguintes seções abordam como gerenciar um cluster de banco de dados do Amazon Aurora MySQL.

Tópicos

- Gerenciar a performance e a escalabilidade do Amazon Aurora MySQL
- Retroceder um cluster de banco de dados Aurora
- Testar o Amazon Aurora MySQL usando consultas de injeção de falhas
- Alterar tabelas no Amazon Aurora usando a DDL rápida
- Exibir o status do volume para um cluster de banco de dados Aurora MySQL

Gerenciar a performance e a escalabilidade do Amazon Aurora MySQL

Dimensionar instâncias de bancos de dados Aurora MySQL

Você pode escalar instâncias de banco de dados Aurora MySQL de duas maneiras: com escalabilidade de instância e escalabilidade de leitura. Para obter mais informações sobre a escalabilidade de leitura, consulte Escalabilidade de leitura.

Você pode escalar o cluster de bancos de dados Aurora MySQL modificando a classe da instância de banco de dados para cada instância de banco de dados do cluster. O Aurora MySQL é compatível com várias classes de instância de banco de dados otimizada para o Aurora. Não use classes de instância db.t2 ou db.t3 para clusters do Aurora maiores com tamanho superior a 40 TB. Para obter as especificações das classes de instância de banco de dados compatíveis com o Aurora MySQL, consulte Classes de instâncias de banco de dados do Amazon Aurora.

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Uso de classes de instância T para desenvolvimento e testes.

Número máximo de conexões com uma instância de bancos de dados Aurora MySQL

O número máximo de conexões permitido para uma instância de bancos de dados Aurora MySQL é determinado pelo parâmetro `max_connections` no grupo de parâmetros do nível da instância para a instância de banco de dados.

A tabela a seguir lista o valor padrão resultante de `max_connections` para cada classe de instância de banco de dados disponível para o Aurora MySQL. É possível aumentar o número máximo de conexões da instância de bancos de dados Aurora MySQL dimensionando a instância até uma classe de instância de banco de dados com mais memória ou definindo um valor maior para o parâmetro `max_connections` no grupo de parâmetros de banco de dados da instância, de até 16.000.

> **Tip**
>
> Se suas aplicações abrem e fecham conexões com frequência ou mantêm um grande número de conexões de longa duração abertas, recomendamos usar o Amazon RDS Proxy. O RDS Proxy é um proxy de banco de dados totalmente gerenciado e altamente disponível que usa grupos de conexões para compartilhar conexões de banco de dados de forma segura e eficiente. Para saber mais sobre o RDS Proxy, consulte Amazon RDS Proxypara Aurora.

Para obter detalhes sobre como as instâncias do Aurora Serverless v2 lidam com esse parâmetro, consulte Conexões máximas do Aurora Serverless v2.

Classe de instância	Valor padrão de max_connections		
db.t2.small	45		
db.t2.medium	90		
db.t3.small	45		
db.t3.medium	90		
db.t3.large	135		

Classe de instância	Valor padrão de max_connections		
db.t4g.medium	90		
db.t4g.large	135		
db.r3.large	1000		
db.r3.xlarge	2000		
db.r3.2xlarge	3000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		
db.r4.large	1000		
db.r4.xlarge	2000		
db.r4.2xlarge	3000		
db.r4.4xlarge	4000		
db.r4.8xlarge	5000		
db.r4.16xlarge	6000		
db.r5.large	1000		
db.r5.xlarge	2000		
db.r5.2xlarge	3000		
db.r5.4xlarge	4000		
db.r5.8xlarge	5000		

Classe de instância	Valor padrão de max_connections		
db.r5.12xlarge	6000		
db.r5.16xlarge	6000		
db.r5.24xlarge	7000		
db.r6g.large	1000		
db.r6g.xlarge	2000		
db.r6g.2xlarge	3000		
db.r6g.4xlarge	4000		
db.r6g.8xlarge	5000		
db.r6g.12xlarge	6000		
db.r6g.16xlarge	6000		
db.r6i.large	1000		
db.r6i.xlarge	2000		
db.r6i.2xlarge	3000		
db.r6i.4xlarge	4000		
db.r6i.8xlarge	5000		
db.r6i.12xlarge	6000		
db.r6i.16xlarge	6000		
db.r6i.24xlarge	7000		

Classe de instância	Valor padrão de max_connections		
db.r6i.32xlarge	7000		
db.r7g.large	1000		
db.r7g.xlarge	2000		
db.r7g.2xlarge	3000		
db.r7g.4xlarge	4000		
db.r7g.8xlarge	5000		
db.r7g.12xlarge	6000		
db.r7g.16xlarge	6000		
db.r7i.large	1000		
db.r7i.xlarge	2000		
db.r7i.2xlarge	3000		
db.r7i.4xlarge	4000		
db.r7i.8xlarge	5000		
db.r7i.12xlarge	6000		
db.r7i.16xlarge	6000		
db.r7i.24xlarge	7000		
db.r7i.48xlarge	8000		
db.r8g.large	1000		

Classe de instância	Valor padrão de max_connections		
db.r8g.xlarge	2000		
db.r8g.2xlarge	3000		
db.r8g.4xlarge	4000		
db.r8g.8xlarge	5000		
db.r8g.12xlarge	6000		
db.r8g.16xlarge	6000		
db.r8g.24xlarge	7000		
db.r8g.48xlarge	8000		
db.x2g.large	2000		
db.x2g.xlarge	3000		
db.x2g.2xlarge	4000		
db.x2g.4xlarge	5000		
db.x2g.8xlarge	6000		
db.x2g.12xlarge	7000		
db.x2g.16xlarge	7000		

Tip

O cálculo do parâmetro `max_connections` usa o log base 2 (diferente do logaritmo natural) e o valor `DBInstanceClassMemory` em bytes para a classe de instância selecionada do

Aurora MySQL. O parâmetro aceita somente valores inteiros, com partes decimais truncadas dos cálculos. A fórmula implementa os limites de conexão da seguinte forma:

- Incremento de 1.000 conexões para instâncias R3, R4 e R5 maiores
- Incremento de 45 conexões para variantes de memória de instância T2 e T3

Exemplo: para db.r6g.large, enquanto a fórmula calcula 1069,2, o sistema implementa 1000 para manter padrões incrementais consistentes.

Se você criar um novo grupo de parâmetros para personalizar seu próprio padrão para o limite de conexão, verá que o limite da conexão padrão é derivado usando uma fórmula baseada no valor `DBInstanceClassMemory`. Como mostra a tabela anterior, a fórmula gera limites de conexão que aumentam em 1000 à medida que a memória é duplicada entre as instâncias progressivamente maiores R3, R4 e R5, e em 45 para tamanhos de memórias diferentes das instâncias T2 e T3.

Consulte Especificação de parâmetros de banco de dados para obter mais detalhes sobre como é feito o cálculo de `DBInstanceClassMemory`.

As instâncias de banco de dados Aurora MySQL e RDS para MySQL têm diferentes quantidades de sobrecarga de memória. Portanto, o valor de `max_connections` pode ser diferente para instâncias de banco de dados Aurora MySQL e RDS para MySQL que usam a mesma classe de instância. Os valores na tabela só se aplicam a instâncias de banco de dados Aurora MySQL.

> **Note**
>
> Os limites de conectividade bem mais baixos das instâncias T2 e T3 ocorrem porque com o Aurora essas classes de instâncias são destinadas apenas para cenários de desenvolvimento e teste, e não para cargas de trabalho de produção.

Os limites de conexão padrão são ajustados pelos sistemas que usam os valores padrão para outros grandes consumidores de memória, como grupo de buffers e cache de consulta. Se você alterar essas outras configurações de seu cluster, considere o ajuste do limite de conexão para levar em conta o aumento ou a redução da memória disponível nas instâncias de banco de dados.

Limites de armazenamento temporário para o Aurora MySQL

O Aurora MySQL armazena tabelas e índices no subsistema de armazenamento do Aurora. O Aurora MySQL usa armazenamento temporário separado ou local para arquivos temporários não persistentes e tabelas temporárias não InnoDB. O armazenamento local também inclui arquivos que são usados para fins como classificar grandes conjuntos de dados durante o processamento de consultas ou para operações de compilação de índice. Ele não inclui tabelas temporárias do InnoDB.

Para obter mais informações sobre tabelas temporárias no Aurora MySQL versão 3, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3. Para obter mais informações sobre tabelas temporárias na versão 2, consulte Comportamento de espaço de tabela temporário no Aurora MySQL versão 2.

Os dados e os arquivos temporários nesses volumes são perdidos ao iniciar e interromper a instância de banco de dados e durante a substituição do host.

Esses volumes de armazenamento local são apoiados pelo Amazon Elastic Block Store (EBS) e podem ser estendidos utilizando uma classe de instância de banco de dados maior. Para ter mais informações sobre armazenamento, consulte Armazenamento do Amazon Aurora.

O armazenamento local também é usado para importar dados do Amazon S3 usando `LOAD DATA FROM S3` ou `LOAD XML FROM S3` e exportar dados para o S3 usando SELECT INTO OUTFILE S3. Para ter mais informações sobre como importar e exportar dados para o S3, consulte o seguinte:

- Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3
- Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3

O Aurora MySQL usa armazenamento permanente separado para logs de erros, logs gerais, logs de consultas lentas e logs de auditoria para a maioria das classes de instância de banco de dados do Aurora MySQL (sem incluir tipos de classe de instância de desempenho expansível, como db.t2, db.t3 e db.t4g). Os dados nesse volume são retidos ao iniciar e interromper a instância de banco de dados e durante a substituição do host.

Esse volume de armazenamento permanente também conta com o suporte do Amazon EBS e tem um tamanho fixo de acordo com a classe de instância de banco de dados. Não é possível ampliá-lo utilizando uma classe de instância de banco de dados maior.

A tabela a seguir mostra a quantidade máxima de armazenamento temporário e permanente disponível para cada classe de instância de banco de dados do Aurora MySQL. Para ter mais informações sobre o suporte a classes de instância de banco de dados para o Aurora, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Classe de instância de banco de dados	Armazenamento temporário/ local máximo disponível (GiB)	Armazenamento máximo adicional disponível para arquivos de log (GiB)
db.x2g.16xlarge	1.280	500
db.x2g.12xlarge	960	500
db.x2g.8xlarge	640	500
db.x2g.4xlarge	320	500
db.x2g.2xlarge	160	60
db.x2g.xlarge	80	60
db.x2g.large	40	60
db.r8g.48xlarge	3840	500
db.r8g.24xlarge	1920	500
db.r8g.16xlarge	1.280	500
db.r8g.12xlarge	960	500
db.r8g.8xlarge	640	500
db.r8g.4xlarge	320	500
db.r8g.2xlarge	160	60
db.r8g.xlarge	80	60
db.r8g.large	32	60

Classe de instância de banco de dados	Armazenamento temporário/ local máximo disponível (GiB)	Armazenamento máximo adicional disponível para arquivos de log (GiB)
db.r7i.48xlarge	3840	500
db.r7i.24xlarge	1920	500
db.r7i.16xlarge	1.280	500
db.r7i.12xlarge	960	500
db.r7i.8xlarge	640	500
db.r7i.4xlarge	320	500
db.r7i.2xlarge	160	60
db.r7i.xlarge	80	60
db.r7i.large	32	60
db.r7g.16xlarge	1.280	500
db.r7g.12xlarge	960	500
db.r7g.8xlarge	640	500
db.r7g.4xlarge	320	500
db.r7g.2xlarge	160	60
db.r7g.xlarge	80	60
db.r7g.large	32	60
db.r6i.32xlarge	2560	500
db.r6i.24xlarge	1920	500
db.r6i.16xlarge	1.280	500

Classe de instância de banco de dados	Armazenamento temporário/ local máximo disponível (GiB)	Armazenamento máximo adicional disponível para arquivos de log (GiB)
db.r6i.12xlarge	960	500
db.r6i.8xlarge	640	500
db.r6i.4xlarge	320	500
db.r6i.2xlarge	160	60
db.r6i.xlarge	80	60
db.r6i.large	32	60
db.r6g.16xlarge	1.280	500
db.r6g.12xlarge	960	500
db.r6g.8xlarge	640	500
db.r6g.4xlarge	320	500
db.r6g.2xlarge	160	60
db.r6g.xlarge	80	60
db.r6g.large	32	60
db.r5.24xlarge	1920	500
db.r5.16xlarge	1.280	500
db.r5.12xlarge	960	500
db.r5.8xlarge	640	500
db.r5.4xlarge	320	500
db.r5.2xlarge	160	60

Classe de instância de banco de dados	Armazenamento temporário/ local máximo disponível (GiB)	Armazenamento máximo adicional disponível para arquivos de log (GiB)
db.r5.xlarge	80	60
db.r5.large	32	60
db.r4.16xlarge	1.280	500
db.r4.8xlarge	640	500
db.r4.4xlarge	320	500
db.r4.2xlarge	160	60
db.r4.xlarge	80	60
db.r4.large	32	60
db.t4g.large	32	–
db.t4g.medium	32	–
db.t3.large	32	–
db.t3.medium	32	–
db.t3.small	32	–
db.t2.medium	32	–
db.t2.small	32	–

Important

Esses valores representam a quantidade teórica máxima de armazenamento gratuito em cada instância de banco de dados. O armazenamento local real disponível para você pode ser menor. O Aurora usa um pouco do armazenamento local para seus processos de gerenciamento, e a instância de banco de dados usa um pouco do armazenamento

local antes mesmo de você carregar dados. É possível monitorar o armazenamento temporário disponível para uma instância de banco de dados específica com a métrica `FreeLocalStorage` do CloudWatch, descrita em Métricas do Amazon CloudWatch para o Amazon Aurora. Você pode verificar a quantidade de armazenamento gratuito no momento atual. Você também pode mapear a quantidade de armazenamento gratuito ao longo do tempo. Monitorar o armazenamento gratuito ao longo do tempo ajuda você a determinar se o valor está aumentando ou diminuindo, ou a encontrar os valores mínimo, máximo ou médio. (Isso não se aplica ao Aurora Serverless v2.)

Retroceder um cluster de banco de dados Aurora

Com o Amazon Aurora Edição compatível com MySQL, é possível retroceder um cluster de banco de dados a um período específico, sem restaurar os dados de um backup.

Sumário

- Visão geral do retrocesso
  - Janela de retrocesso
  - Tempo de retrocesso
  - Limitações de retrocesso
- Disponibilidade de regiões e versões
- Considerações de atualização para clusters habilitados para o retrocesso
- Configurar o retrocesso de um cluster de banco de dados do Aurora MySQL
- Realizar um retrocesso de um cluster de banco de dados do Aurora MySQL
- Monitorar o retrocesso de um cluster de banco de dados do Aurora MySQL
- Assinar um evento de retrocesso com o console
- Recuperar retrocessos existentes
- Desativar o retrocesso de um cluster de banco de dados do Aurora MySQL

Visão geral do retrocesso

O retrocesso retrocede o cluster de banco de dados ao período que você especificar. O retrocesso não é um substituto para o backup do cluster de banco de dados, para que você possa restaurá-lo para um período específico. No entanto, o retrocesso fornece as seguintes vantagens em relação ao backup e à restauração tradicionais:

- É possível desfazer erros facilmente. Se, por engano, você executar uma ação destrutiva, como, por exemplo, um DELETE sem uma cláusula WHERE, é possível retroceder o cluster de banco de dados a um período anterior à ação destrutiva, com interrupção mínima do serviço.
- É possível retroceder um cluster de banco de dados rapidamente. Restaurar um cluster de banco de dados para um período específico ativa um novo cluster de banco de dados e o restaura a partir de dados de backup ou de um snapshot de cluster de banco de dados, o que pode levar horas. O retrocesso de um cluster de banco de dados não requer um novo cluster de banco de dados e o retrocede em minutos.
- É possível explorar alterações de dados anteriores. Você pode voltar e avançar o retrocesso de um cluster de banco de dados repetidamente para ajudar a determinar quando uma alteração de dados específica ocorreu. Por exemplo, você pode retroceder um cluster de banco de dados três horas e, em seguida, avançar uma hora. Nesse caso, o período do retrocesso é duas horas antes do período original.

> **Note**
>
> Para obter informações sobre como restaurar um cluster de banco de dados para um período específico, consulte Visão geral do backup e da restauração de um cluster de banco de dados do Aurora.

Janela de retrocesso

Com o retrocesso, há uma janela de retrocesso de destino e uma janela de retrocesso real:

- A janela de retrocesso de destino é a quantidade de tempo que você deseja poder retroceder seu cluster de banco de dados. Ao habilitar o retrocesso, você especifica uma janela de retrocesso de destino. Por exemplo, você pode especificar uma janela de retrocesso de destino de 24 horas se quiser retroceder o cluster de banco de dados um dia.
- A janela de retrocesso real é a quantidade real de tempo para a qual você pode retroceder seu cluster de banco de dados, que pode ser menor que a janela de retrocesso de destino. A janela de retrocesso real é baseada em sua workload e no armazenamento disponível para armazenar informações sobre alterações no banco de dados, denominadas registros de alterações.

Ao atualizar seu cluster de banco de dados Aurora com o retrocesso habilitado, você gera registros de alterações. O Aurora retém os registros de alterações da janela de retrocesso de destino, e você

paga uma taxa por hora para armazená-los. A janela de retrocesso de destino e a workload em seu cluster de banco de dados determinam o número de registros de alterações armazenados. A workload é o número de alterações feitas em seu cluster de banco de dados em um determinado período de tempo. Se sua workload for pesada, você armazenará mais registros de alterações em sua janela de retrocesso do que se sua workload for leve.

Você pode pensar em sua janela de retrocesso de destino como a meta para o período máximo de tempo que deseja retroceder seu cluster de banco de dados. Na maioria dos casos, você pode retroceder o tempo máximo que especificou. No entanto, em alguns casos, o cluster de banco de dados não pode armazenar registros de alterações suficientes para retroceder a quantidade máxima de tempo, além da janela de retrocesso real ser menor do que a meta. Normalmente, a janela de retrocesso real é menor que a meta quando você tem uma workload extremamente pesada em seu cluster de banco de dados. Quando a janela de retrocesso atual é menor que sua meta, enviamos uma notificação.

Quando o retrocesso é ativado para um cluster de banco de dados e você exclui uma tabela armazenada nele, o Aurora mantém essa tabela nos registros de alterações de retrocesso. Ele faz isso para que você possa reverter a um período anterior à exclusão da tabela. Se você não tiver espaço suficiente na sua janela de retrocesso para armazenar a tabela, ela poderá ser removida dos registros de alterações de retrocesso eventualmente.

Tempo de retrocesso

O Aurora sempre retrocede a um período que seja consistente para o cluster de banco de dados. Isso elimina a possibilidade de transações não confirmadas quando o retrocesso é concluído. Quando você especifica um período para um retrocesso, o Aurora escolhe automaticamente o período consistente mais próximo possível. Essa abordagem significa que o retrocesso concluído pode não corresponder exatamente ao período especificado, mas você pode determinar o período exato para um retrocesso usando o comando describe-db-cluster-backtracks da CLI da AWS. Para obter mais informações, consulte Recuperar retrocessos existentes.

Limitações de retrocesso

As limitações a seguir se aplicam ao retrocesso:

- O retrocesso somente está disponível em clusters de banco de dados que foram criados com o recurso Retrocesso habilitado. Não é possível modificar um cluster de banco de dados para habilitar o recurso Backtrack. É possível habilitar o recurso Retrocesso ao criar um cluster de banco de dados ou restaurar um snapshot de um cluster de banco de dados.

- O limite para uma janela de retrocesso é de 72 horas.
- O retrocesso afeta todo o cluster de banco de dados. Por exemplo, você não pode seletivamente retroceder uma única tabela ou uma única atualização de dados.
- Não é possível criar réplicas de leitura entre regiões usando um cluster habilitado para retrocesso, mas você pode habilitar a replicação de log binário (binlog) no cluster. Além disso, quando você tenta retroceder um cluster de banco de dados para o qual o registro em log binário está habilitado, normalmente ocorre um erro, a menos que você opte por forçar o retrocesso. Qualquer tentativa de forçar um retrocesso interromperá as réplicas de leitura posteriores e interferirá em outras operações, como implantações azul/verde.
- Não é possível retroceder um clone do banco de dados a um período anterior à criação dele. No entanto, você pode usar o banco de dados original para retroceder a um período anterior à criação do clone. Para obter mais informações sobre clonagem de banco de dados, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora.
- O retrocesso causa uma breve interrupção da instância de banco de dados. É necessário parar ou pausar seus aplicativos antes de iniciar uma operação de retrocesso para garantir que não haja novas solicitações de leitura ou gravação. Durante a operação de retrocesso, o Aurora pausa o banco de dados, fecha todas as conexões abertas e elimina leituras e gravações não confirmadas. Em seguida, aguarda a conclusão da operação de retrocesso.
- Não é possível restaurar um snapshot entre regiões de um cluster habilitado para retrocesso em uma região da AWS que não oferece suporte ao retrocesso.
- Se você executar um upgrade local para um cluster habilitado para retrocesso do Aurora MySQL versão 2 para a versão 3, não poderá retroceder a um ponto no tempo antes do upgrade.

Disponibilidade de regiões e versões

O backtrack não está disponível para o Aurora PostgreSQL.

Veja a seguir os mecanismos compatíveis e a disponibilidade de regiões para o retrocesso com o Aurora MySQL.

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Norte da Virgínia)	Todas as versões	Todas as versões

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Leste dos EUA (Ohio)	Todas as versões	Todas as versões
Oeste dos EUA (Norte da Califórnia)	Todas as versões	Todas as versões
Oeste dos EUA (Oregon)	Todas as versões	Todas as versões
Africa (Cape Town)	–	–
Ásia-Pacífico (Hong Kong)	–	–
Ásia-Pacífico (Jacarta)	–	–
Ásia-Pacífico (Malásia)	–	–
Ásia-Pacífico (Melbourne)	–	–
Ásia-Pacífico (Mumbai)	Todas as versões	Todas as versões
Ásia-Pacífico (Nova Zelândia)	–	–
Ásia-Pacífico (Osaka)	Todas as versões	Versão 2.07.3 e posterior
Ásia-Pacífico (Seul)	Todas as versões	Todas as versões

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Ásia-Pacífico (Singapura)	Todas as versões	Todas as versões
Ásia-Pacífico (Sydney)	Todas as versões	Todas as versões
Ásia-Pacífico (Taipei)	–	–
Ásia-Pacífico (Tailândia)	–	–
Ásia-Pacífico (Tóquio)	Todas as versões	Todas as versões
Canadá (Central)	Todas as versões	Todas as versões
Oeste do Canadá (Calgary)	–	–
China (Beijing)	–	–
China (Ningxia)	–	–
Europa (Frankfurt)	Todas as versões	Todas as versões
Europa (Irlanda)	Todas as versões	Todas as versões
Europa (Londres)	Todas as versões	Todas as versões
Europe (Milan)	–	–
Europe (Paris)	Todas as versões	Todas as versões

Região	Aurora MySQL versão 3	Aurora MySQL versão 2
Europa (Espanha)	–	–
Europa (Estocolmo)	–	–
Europa (Zurique)	–	–
Israel (Tel Aviv)	–	–
México (Central)	–	–
Oriente Médio (Bahrein)	–	–
Oriente Médio (Emirados Árabes Unidos)	–	–
South America (São Paulo)	–	–
AWS GovCloud (Leste dos EUA)	–	–
AWS GovCloud (Oeste dos EUA)	–	–

Considerações de atualização para clusters habilitados para o retrocesso

Você pode fazer upgrade de um cluster de banco de dados habilitar para retrocesso do Aurora MySQL versão 2 para a versão 3, porque todas as versões secundárias do Aurora MySQL versão 3 são compatíveis com o recurso de retrocesso.

Configurar o retrocesso de um cluster de banco de dados do Aurora MySQL

Para usar o recurso Retrocesso, é necessário ativar o retrocesso e especificar uma janela de retrocesso de destino. Caso contrário, o retrocesso será desativado.

Na janela de retrocesso de destino, especifique o tempo que você deseja retroceder em seu banco de dados usando Backtrack (Retrocesso). O Aurora tenta reter registros de alteração suficientes para suportar essa janela de tempo.

Console

Você pode usar o console para configurar o retrocesso ao criar um novo cluster de banco de dados. Você também pode modificar um cluster de banco de dados para alterar a janela de retrocesso de um cluster habilitado para o retrocesso. Se você desativar totalmente o retrocesso em um cluster definindo a janela de retrocesso como 0, não poderá ativar o retrocesso novamente nesse cluster.

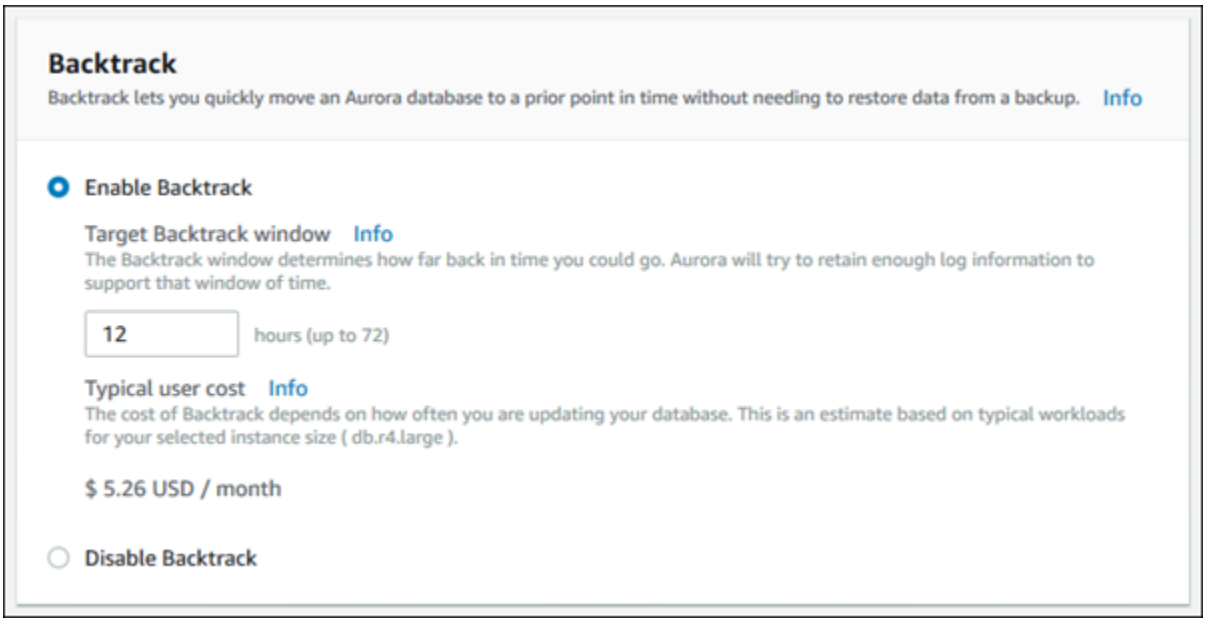
Tópicos

- Configurar o retrocesso com o console ao criar um cluster de banco de dados
- Configurar o retrocesso com o console ao modificar um cluster de banco de dados

Configurar o retrocesso com o console ao criar um cluster de banco de dados

Quando você cria um novo cluster de banco de dados Aurora MySQL, o retrocesso é configurado quando você escolhe Enable Backtrack (Habilitar retrocesso) e especifica um valor na Target Backtrack window (Janela de retrocesso de destino) que seja maior que zero na seção Backtrack (Retrocesso).

Para criar um cluster de banco de dados, siga as instruções em Criar um cluster de bancos de dados do Amazon Aurora. A imagem a seguir mostra a seção Retrocesso.



Ao criar um novo cluster de banco de dados, o Aurora não possui dados para a workload do cluster de banco de dados. Portanto, não é possível estimar um custo especificamente para o novo cluster de banco de dados. Em vez disso, o console apresenta um custo de usuário típico na janela de retrocesso de destino especificada com base em uma workload comum. O custo típico serve para fornecer uma referência geral para o custo do recurso Retrocesso.

> **Important**
>
> Seu custo real pode não corresponder ao custo típico, pois o custo real é baseado na workload do cluster de banco de dados.

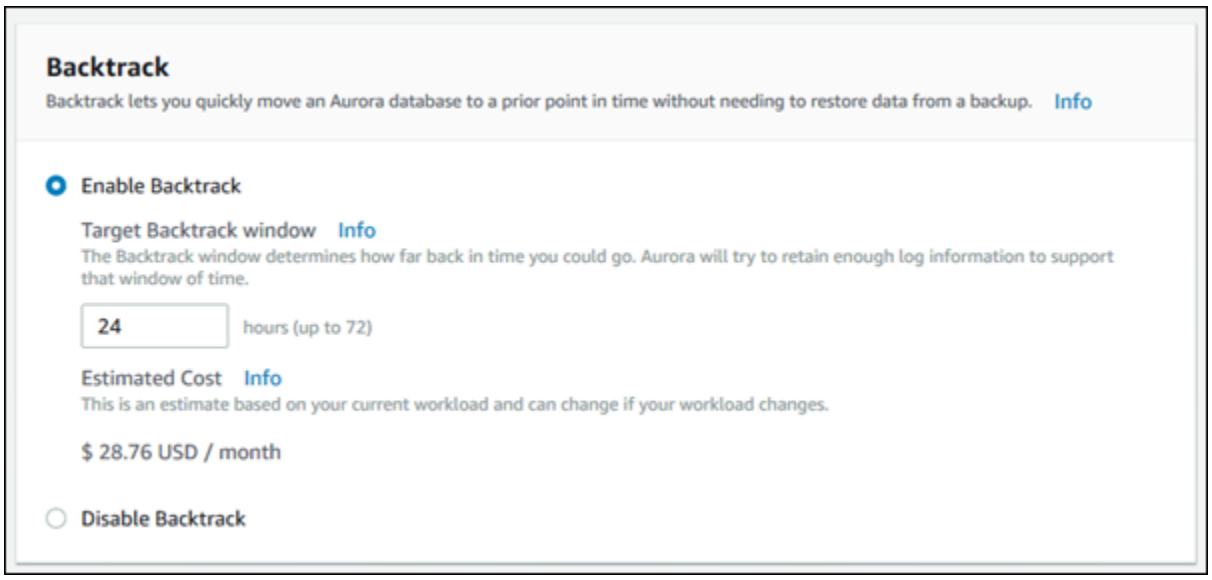
Configurar o retrocesso com o console ao modificar um cluster de banco de dados

Você pode modificar o retrocesso de um cluster de banco de dados usando o console.

> **Note**
>
> Atualmente, é possível modificar o retrocesso somente para um cluster de banco de dados que tenha o recurso Retrocesso habilitado. A seção Backtrack (Retrocesso) não é exibida para um cluster de banco de dados que foi criado com o recurso Retrocesso desabilitado ou se o recurso Retrocesso foi desabilitado para o cluster de banco de dados.

Para modificar o retrocesso de um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o cluster que deseja modificar e escolha Modify (Modificar).
4. A Target Backtrack window (Janela de retrocesso de destino), modifica a quantidade de tempo que você deseja retroceder seu cluster de banco de dados. O limite é de 72 horas.



O console mostra o custo estimado da quantidade de tempo especificada com base na workload anterior do cluster de banco de dados:

- Se o retrocesso foi desabilitado no cluster de banco de dados, a estimativa de custo é baseada na métrica `VolumeWriteIOPS` do cluster de banco de dados no Amazon CloudWatch.
- Se o retrocesso foi ativado anteriormente no cluster de banco de dados, a estimativa de custo é baseada na métrica `BacktrackChangeRecordsCreationRate` do cluster de banco de dados no Amazon CloudWatch.

5. Escolha Continue.
6. Em Scheduling of Modifications (Programação de modificações), selecione uma das seguintes opções:

   - Apply during the next scheduled maintenance window (Aplicar durante a próxima janela de manutenção programada) – esperar para aplicar a modificação da Target Backtrack window (Janela de retrocesso de destino) na próxima janela de manutenção.
   - Apply immediately (Aplicar imediatamente) – aplicar a modificação da Target Backtrack window (Janela de retrocesso de destino) o mais breve possível.

7. Selecione Modify Cluster (Modificar cluster).

AWS CLI

Ao criar um novo cluster de banco de dados Aurora MySQL usando o comando create-db-cluster da AWS CLI, o retrocesso é configurado quando você especifica um valor para `--backtrack-window` que seja maior que zero. O valor `--backtrack-window` especifica a janela de retrocesso de destino. Para obter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Você também pode especificar o valor de `--backtrack-window` usando os seguintes comandos da CLI da AWS:

- modify-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

O procedimento a seguir descreve como modificar a janela de retrocesso de destino para um cluster de banco de dados usando a AWS CLI.

Para modificar a janela de retrocesso de destino para um cluster de banco de dados usando a AWS CLI

- Chame o comando modify-db-cluster da CLI da AWS e forneça os seguintes valores:
  - `--db-cluster-identifier` – o nome do cluster de banco de dados.
  - `--backtrack-window` – o número máximo de segundos que você deseja retroceder o cluster de banco de dados.

  O exemplo a seguir define a janela de retrocesso de destino do `sample-cluster` como um dia (86.400 segundos).

  Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 86400
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 86400
```

> **Note**
>
> No momento, você pode ativar o retrocesso apenas para um cluster de banco de dados criado com o recurso Retrocesso ativado.

API do RDS

Ao criar um novo cluster de banco de dados Aurora MySQL usando a operação CreateDBCluster da API do Amazon RDS, o retrocesso é configurado ao especificar um valor para `BacktrackWindow` que seja maior que zero. O valor `BacktrackWindow` especifica a janela de retrocesso de destino para o cluster de banco de dados especificado no valor do `DBClusterIdentifier`. Para obter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Também é possível especificar o valor `BacktrackWindow` usando as seguintes operações da API:

- ModifyDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

> **Note**
>
> No momento, você pode ativar o retrocesso apenas para um cluster de banco de dados criado com o recurso Retrocesso ativado.

Realizar um retrocesso de um cluster de banco de dados do Aurora MySQL

Você pode retroceder um cluster de banco de dados a um time stamp de retrocesso especificado. Se o time stamp do retrocesso não for anterior ao período do retrocesso mais antigo possível, e não for um período futuro, o cluster de banco de dados será retrocedido a esse time stamp.

Caso contrário, geralmente ocorre um erro. Além disso, se você tentar retroceder um cluster de banco de dados para o qual o log binário está habilitado, um erro normalmente ocorre, a menos que você tenha optado por forçar que o retrocesso ocorra. Forçar que um retrocesso ocorra pode interferir em outras operações que usam o log binário.

> **Important**
>
> O retrocesso não gera entradas de log binário para as alterações que ele faz. Se você tiver o log binário ativado para o cluster de banco de dados, o retrocesso poderá não ser compatível com sua implementação do log binário.

> **Note**
>
> Para clones de banco de dados, não é possível retroceder o cluster de banco de dados antes da data e da hora em que o clone foi criado. Para obter mais informações sobre clonagem de banco de dados, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora.

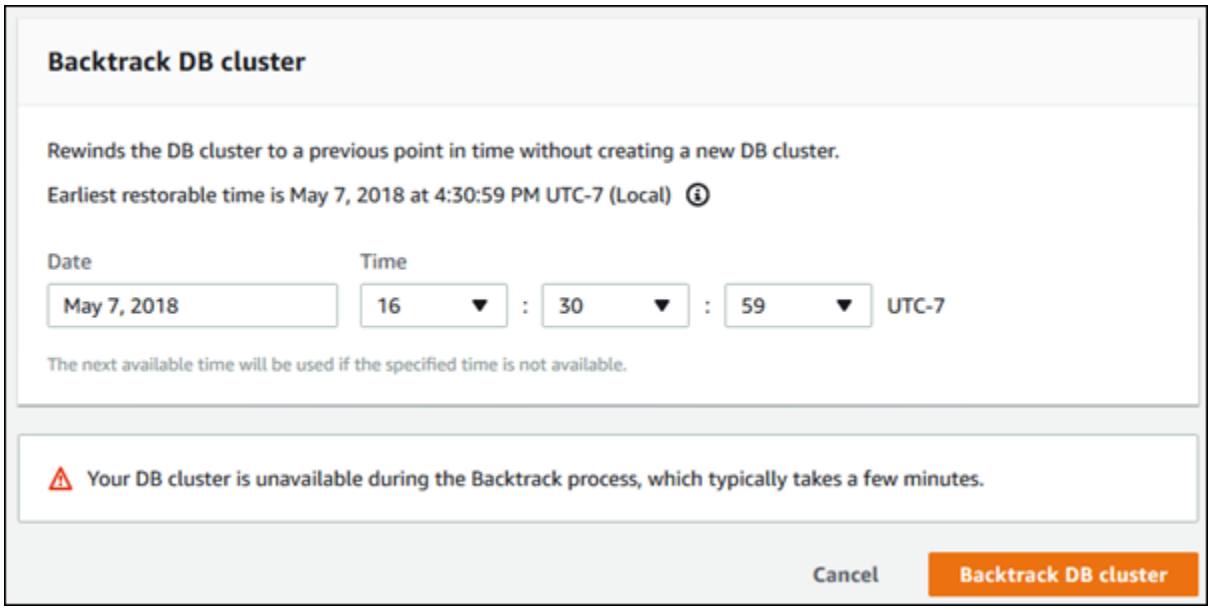
Console

O procedimento a seguir descreve como executar uma operação de retrocesso para um cluster de banco de dados usando o console.

Para executar uma operação de retrocesso usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Instances (Instâncias).
3. Selecione a instância primária do cluster de banco de dados que deseja retroceder.
4. Em Actions (Ações), escolha Backtrack DB cluster (Retroceder cluster de banco de dados).

5. Na página Backtrack DB cluster (Retroceder cluster de banco de dados), digite o timestamp do retrocesso para o qual retroceder o cluster de banco de dados.



6. Escolha Backtrack DB cluster (Retroceder cluster de banco de dados)

AWS CLI

O procedimento a seguir descreve como retroceder um cluster de banco de dados usando a AWS CLI.

Para retroceder um cluster de banco de dados usando a AWS CLI

- Chame o comando backtrack-db-cluster da CLI da AWS e forneça os seguintes valores:
  - `--db-cluster-identifier` – o nome do cluster de banco de dados.
  - `--backtrack-to` – o time stamp de retrocesso segundo o qual você deseja retroceder o cluster de banco de dados, especificado no formato ISO 8601.

  O exemplo a seguir retrocede o cluster de banco de dados `sample-cluster` a 19 de março de 2018, às 10h.

  Para Linux, macOS ou Unix:

```
aws rds backtrack-db-cluster \
    --db-cluster-identifier sample-cluster \
```

```
    --backtrack-to 2018-03-19T10:00:00+00:00
```

Para Windows:

```
aws rds backtrack-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-to 2018-03-19T10:00:00+00:00
```

API do RDS

Para retroceder um cluster de banco de dados usando a API do Amazon RDS, use a ação BacktrackDBCluster. Essa ação retrocede o cluster de banco de dados especificado no valor do `DBClusterIdentifier` ao período especificado.

Monitorar o retrocesso de um cluster de banco de dados do Aurora MySQL

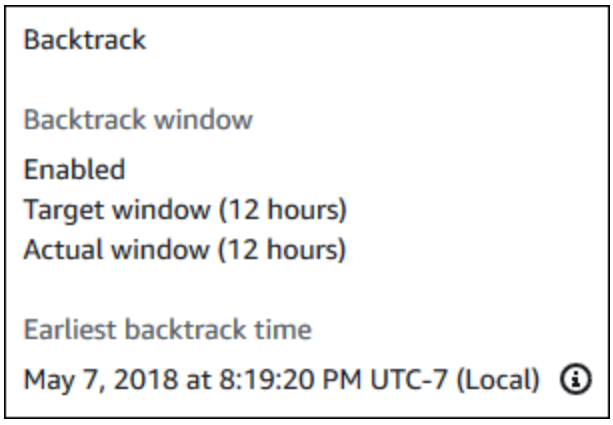
Você pode visualizar informações de retrocesso e monitorar métricas de retrocesso para um cluster de banco de dados.

Console

Para visualizar informações de retrocesso e monitorar métricas de retrocesso usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o nome do cluster de banco de dados para abrir informações sobre ele.

   As informações de retrocesso estão na seção Retrocesso.



Quando o retrocesso é ativado, as seguintes informações ficam disponíveis:

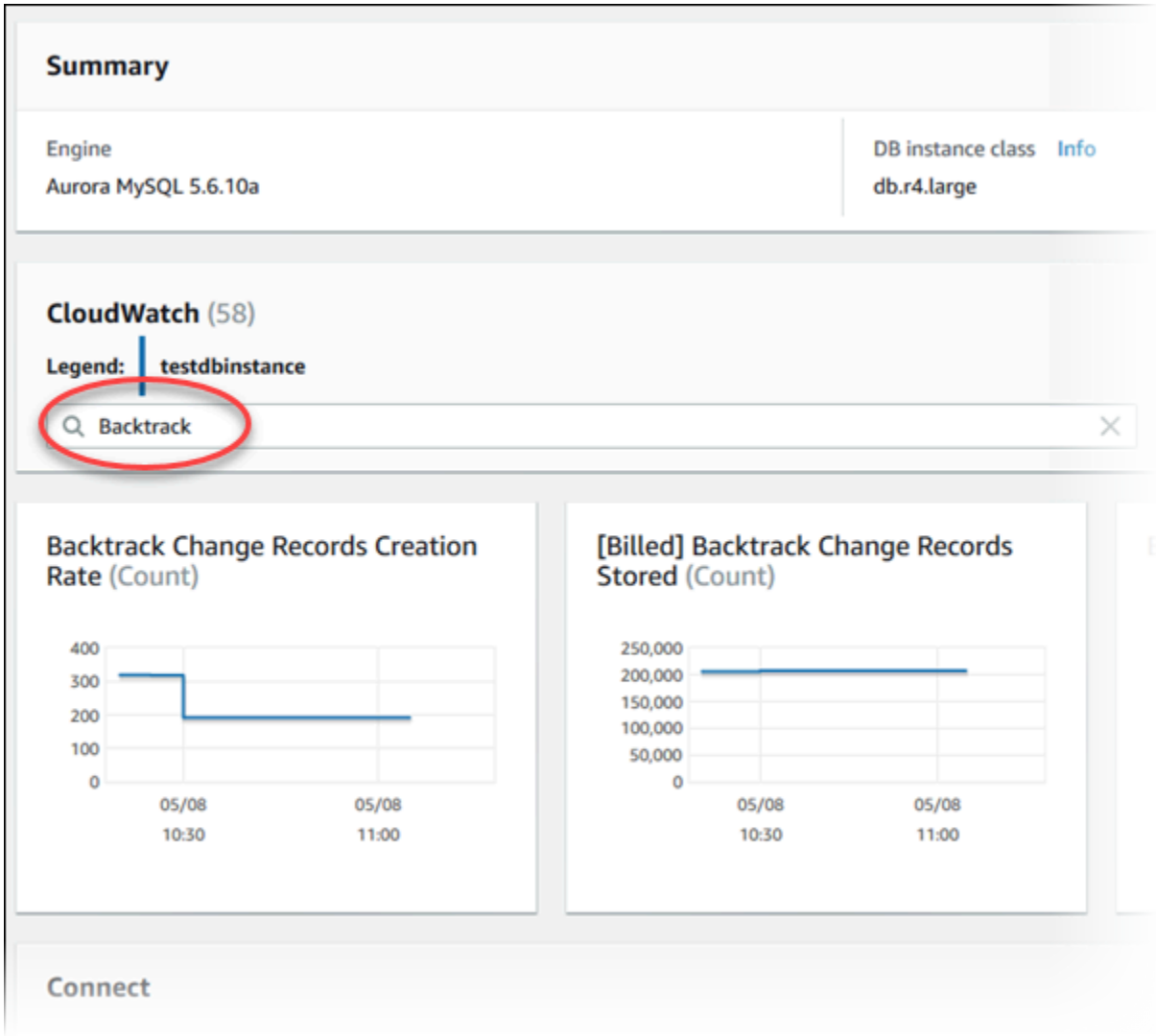
- Target window (Janela de destino) – o tempo especificado no momento para a janela de retrocesso de destino. O destino é o tempo máximo que você pode retroceder se houver armazenamento suficiente.
- Actual window (Janela real) – o tempo real que você pode retroceder, que pode ser menor que a janela de retrocesso de destino. A janela de retrocesso real é baseada em sua workload e no armazenamento disponível para reter registros de alterações de retrocesso.
- Earliest backtrack time (Período de retrocesso mais antigo) – o tempo de retrocesso mais antigo possível do cluster de banco de dados. Não é possível retroceder o cluster de banco de dados a um período anterior ao período exibido.

4. Faça o seguinte para visualizar métricas de retrocesso para o cluster de banco de dados:

   a. No painel de navegação, escolha Instances (Instâncias).

   b. Selecione o nome da instância primária do cluster de banco de dados para exibir seus detalhes.

   c. Na seção CloudWatch, digite **`Backtrack`** na caixa CloudWatch para mostrar apenas as métricas do Retrocesso.



As seguintes métricas são exibidas:

- Backtrack Change Records Creation Rate (Count) (Taxa de criação de registros de alterações de retrocesso (contagem)) – essa métrica mostra o número de registros de alterações de retrocesso criados no período de cinco minutos para o cluster de banco de dados. Você pode usar essa métrica para estimar o custo de retrocesso para a janela de retrocesso de destino.
- [Billed] Backtrack Change Records Stored (Count) ([Faturado] Registros de alterações de retrocesso armazenados (contagem)) – essa métrica mostra o número real de registros de alterações de retrocesso usados pelo cluster de banco de dados.
- Backtrack Window Actual (Minutes) (Janela de retrocesso real (minutos)) – essa métrica mostra se há uma diferença entre a janela de retrocesso de destino e a janela de retrocesso real. Por exemplo, se a janela de retrocesso de destino for de 2 horas (120

minutos) e essa métrica mostrar que a janela de retrocesso real é de 100 minutos, a janela de retrocesso real será menor que o destino.

- Backtrack Window Alert (Count) (Alerta da janela de retrocesso (contagem)) – essa métrica mostra com que frequência a janela de retrocesso real é menor que a janela de retrocesso de destino em um determinado período.

> **Note**
>
> As métricas a seguir podem atrasar o período atual:
>
> - Backtrack Change Records Creation Rate (Count) (Taxa de criação de registros de alterações de retrocesso (contagem)
> - [Billed] Backtrack Change Records Stored (Count) ([Faturado] Registros de alterações de retrocesso armazenados (contagem)

AWS CLI

O procedimento a seguir descreve como visualizar as informações de retrocesso para um cluster de banco de dados usando a AWS CLI.

Para visualizar informações de retrocesso para um cluster de banco de dados usando a AWS CLI

- Chame o comando describe-db-clusters da CLI da AWS e forneça os seguintes valores:
  - `--db-cluster-identifier` – o nome do cluster de banco de dados.

  O exemplo a seguir lista informações de retrocesso para o `sample-cluster`.

  Para Linux, macOS ou Unix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier sample-cluster
```

  Para Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier sample-cluster
```

API do RDS

Para visualizar informações de retrocesso para um cluster de banco de dados usando a API do Amazon RDS, use a operação DescribeDBClusters. Esta ação retorna informações de retrocesso para o cluster de banco de dados especificado no valor de `DBClusterIdentifier`.

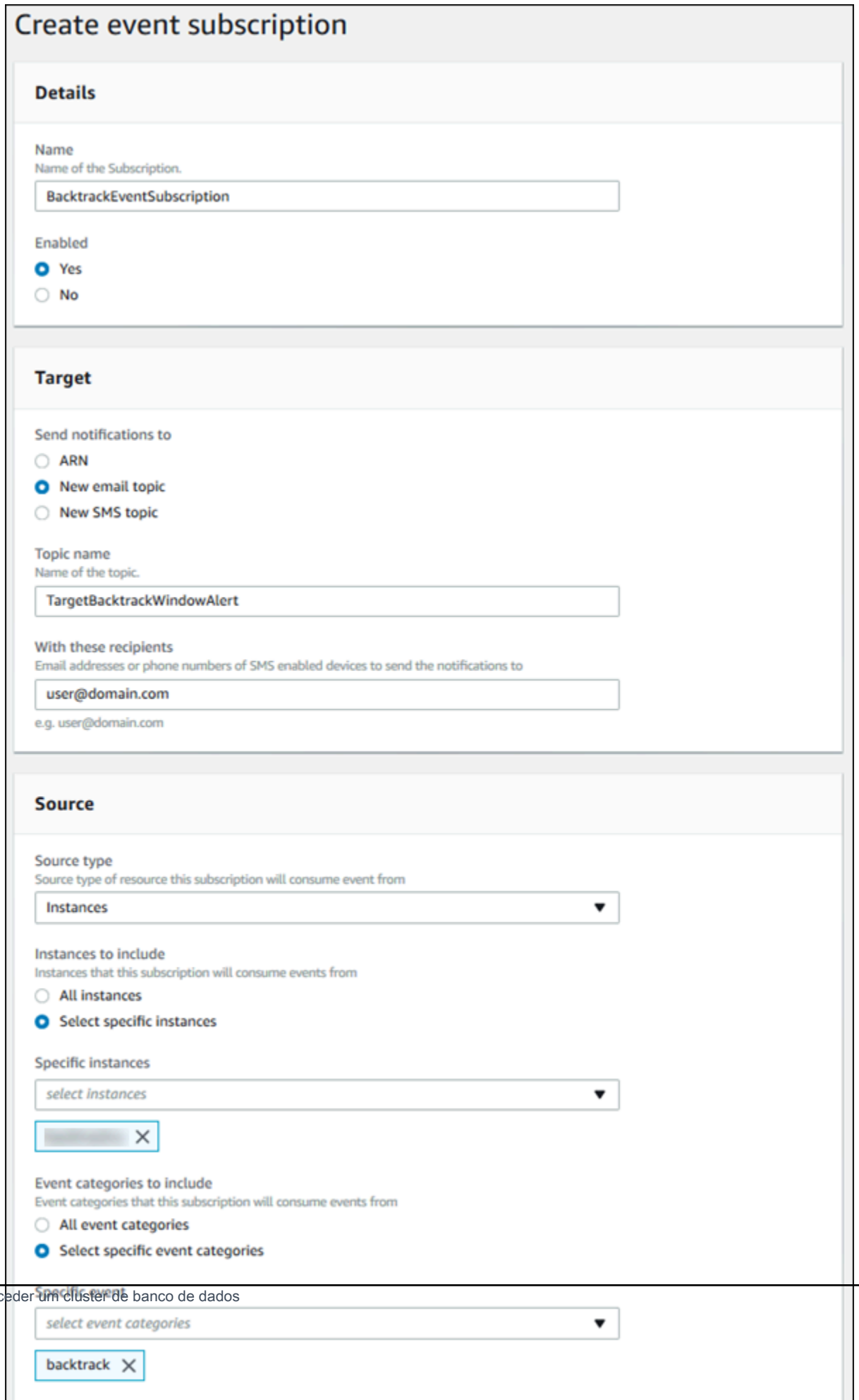
Assinar um evento de retrocesso com o console

O procedimento a seguir descreve como assinar um evento de retrocesso usando o console. O evento envia um email ou uma notificação de texto quando a janela de retrocesso real for menor que a janela de retrocesso de destino.

Para visualizar informações de retrocesso usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Selecione Event subscriptions (Assinaturas de eventos).
3. Selecione Criar assinatura de evento.
4. Na caixa Name (Nome), digite um nome para a assinatura do evento e verifique se Yes (Sim) está selecionado para Enabled (Ativado).
5. Na seção Target (Destino), selecione New email topic (Novo tópico de email).
6. Em Topic name (Nome do tópico), digite um nome para o tópico e, em With these recipients (Com estes destinatários), insira os endereços de e-mail ou números de telefone para receber as notificações.
7. Na seção Source (Origem), selecione Instances (Instâncias) para Source type (Tipo de origem).
8. Em Instances to include (Instâncias a serem incluídas), selecione Select specific instances (Selecionar instâncias específicas) e escolha a instância de banco de dados.
9. Em Event categories to include (Categorias de eventos a serem incluídas), selecione Select specific event categories (Selecionar categorias de eventos específicos) e escolha backtrack (retrocesso).

   A página deve ser semelhante à página a seguir.


Create event subscription
Details
Name
Name of the Subscription.
BacktrackEventSubscription
Enabled
Yes
No
Target
Send notifications to
ARN
New email topic
New SMS topic
Topic name
Name of the topic.
TargetBacktrackWindowAlert
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
e.g. user@domain.com
Source
Source type
Source type of resource this subscription will consume event from
Instances
Instances to include
Instances that this subscription will consume events from
All instances
Select specific instances
Specific instances
select instances
Event categories to include
Event categories that this subscription will consume events from
All event categories
Select specific event categories
select event categories
backtrack

10. Escolha Criar.

Recuperar retrocessos existentes

Você pode recuperar informações sobre retrocessos existentes para um cluster de banco de dados. Essas informações incluem o identificador exclusivo do retrocesso, a data e a hora para as quais e a partir das quais foi feito o retrocesso, a data e a hora em que o retrocesso foi solicitado e o status atual do retrocesso.

> **Note**
>
> Atualmente, não é possível recuperar retrocessos existentes usando o console.

AWS CLI

O procedimento a seguir descreve como recuperar retrocessos existentes para um cluster de banco de dados usando a AWS CLI.

Para recuperar retrocessos existentes usando a AWS CLI

- Chame o comando describe-db-cluster-backtracks da CLI da AWS e forneça os seguintes valores:
  - `--db-cluster-identifier` – o nome do cluster de banco de dados.

  O exemplo a seguir recupera retrocessos existentes para o `sample-cluster`.

  Para Linux, macOS ou Unix:

  ```
  aws rds describe-db-cluster-backtracks \
      --db-cluster-identifier sample-cluster
  ```

  Para Windows:

  ```
  aws rds describe-db-cluster-backtracks ^
      --db-cluster-identifier sample-cluster
  ```

API do RDS

Para recuperar informações sobre retrocessos para um cluster de banco de dados usando a API do Amazon RDS, use a operação DescribeDBClusterBacktracks. Esta operação retorna informações sobre retrocessos para o cluster de banco de dados especificado no valor do `DBClusterIdentifier`.

Desativar o retrocesso de um cluster de banco de dados do Aurora MySQL

É possível desativar o recurso Retrocesso para um cluster de banco de dados.

Console

Você pode desativar o retrocesso para um cluster de banco de dados usando o console. Depois de desativar totalmente o retrocesso em um cluster, não será possível ativá-lo novamente nesse cluster.

Para desativar o recurso de Retrocesso para um cluster de banco de dados usando o console.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o cluster que você deseja modificar e escolha Modify (Modificar).
4. Na seção Backtrack (Retrocesso), selecione Disable Backtrack (Desativar retrocesso).
5. Escolha Continue.
6. Em Scheduling of Modifications (Programação de modificações), selecione uma das seguintes opções:
   - Apply during the next scheduled maintenance window (Aplicar durante a próxima janela de manutenção programada) – esperar para aplicar a modificação na próxima janela de manutenção.
   - Apply immediately (Aplicar imediatamente) – aplicar a modificação o mais breve possível.
7. Escolha Modify Cluster.

AWS CLI

Você pode desabilitar o recurso Retrocesso para um cluster de banco de dados usando a AWS CLI ao definir a janela de retrocesso de destino como `0` (zero). Depois de desativar totalmente o retrocesso em um cluster, não será possível ativá-lo novamente nesse cluster.

Para modificar a janela de retrocesso de destino para um cluster de banco de dados usando a AWS CLI

- Chame o comando modify-db-cluster da CLI da AWS e forneça os seguintes valores:
  - `--db-cluster-identifier` – o nome do cluster de banco de dados.
  - `--backtrack-window` – especificar `0` para desativar o retrocesso.

  O exemplo a seguir desativa o recurso Retrocesso para o `sample-cluster` ao definir a `--backtrack-window` como `0`.

  Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 0
```

  Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 0
```

API do RDS

Para desabilitar o recurso Retrocesso para um cluster de banco de dados usando a API do Amazon RDS, use a operação ModifyDBCluster. Defina o valor da `BacktrackWindow` como `0` (zero) e especifique o cluster de banco de dados no valor do `DBClusterIdentifier`. Depois de desativar totalmente o retrocesso em um cluster, não será possível ativá-lo novamente nesse cluster.

Testar o Amazon Aurora MySQL usando consultas de injeção de falhas

Você pode testar a tolerância a falhas do cluster de banco de dados Aurora MySQL usando consultas de injeção de falha. Consultas de injeção de falhas são emitidas como comandos SQL para uma instância do Amazon Aurora. Eles permitem que você agende uma ocorrência simulada de um dos seguintes eventos:

- Uma falha de uma instância de banco de dados de gravação ou leitura
- Uma falha de uma réplica do Aurora
- Uma falha de disco
- Congestionamento de disco

Quando uma consulta de injeção de falha especifica uma falha, ela força uma falha da instância de banco de dados do Aurora MySQL. As outras consultas de injeção de falha resultam em simulações de eventos de falha, mas não desencadeiam o evento. Ao enviar uma consulta de injeção de falha, especifique também um período para a simulação do evento de falha.

Você pode enviar uma consulta de injeção de falha a uma das instâncias da réplica do Aurora conectando-se ao endpoint da réplica do Aurora. Para obter mais informações, consulte Conexões de endpoints do Amazon Aurora.

A execução de consultas de injeção de falhas requer todos os privilégios de usuário principal. Para obter mais informações, consulte Privilégios da conta de usuário mestre.

Teste de pane da instância

Você pode forçar uma pane em uma instância do Amazon Aurora usando a consulta de injeção de falha `ALTER SYSTEM CRASH`.

No que tange a essa consulta de injeção de falha, não ocorrerá um failover. Se desejar testar um failover, é possível escolher a ação de instância Failover para o cluster de banco de dados no console do RDS ou usar o comando failover-db-cluster da AWS CLI ou a operação de API FailoverDBCluster do RDS.

Sintaxe

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Opções

Esta consulta de injeção de falha considera um dos seguintes tipos de pane:

- **`INSTANCE`** — uma pane é simulada no banco de dados compatível com MySQL para a instância do Amazon Aurora.

- **`DISPATCHER`** — Uma falha no dispatcher é simulada na instância de gravador do cluster de banco de dados do Aurora. O dispatcher grava atualizações no volume do cluster para um cluster de banco de dados do Amazon Aurora.
- **`NODE`** — uma pane é simulada no banco de dados compatível com MySQL e no dispatcher para a instância do Amazon Aurora. No que tange a essa simulação de injeção de falha, o cache também é excluído.

O tipo de pane padrão é `INSTANCE`.

Teste de falha em uma réplica do Aurora

Você pode simular a falha de uma réplica do Aurora usando a consulta de injeção de falha `ALTER SYSTEM SIMULATE READ REPLICA FAILURE`.

Uma falha na réplica do Aurora bloqueia todas as solicitações feitas na instância do leitor a ela ou a todas as réplicas do Aurora no cluster de banco de dados durante um intervalo de tempo especificado. Quando o intervalo de tempo acabar, as réplicas do Aurora afetadas serão sincronizadas automaticamente com a instância do gravador.

Sintaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

- **`percentage_of_failure`** — A porcentagem de solicitações para obstruir durante o evento de falha. Esse valor pode ser um duplo entre 0 e 100. Se você especificar 0, nenhuma solicitação será bloqueada. Se você especificar 100, todas as solicitações serão bloqueadas.
- Tipo de falha — O tipo de falha a ser simulado. Especifique `TO ALL` para simular falhas para todas as réplicas do Aurora no cluster de banco de dados. Especifique `TO` e o nome da réplica do Aurora para simular uma falha de uma única réplica do Aurora. O tipo de falha padrão é `TO ALL`.
- **`quantity`** — o tempo durante o qual a falha da réplica do Aurora será simulada. O intervalo é uma quantidade seguida por uma unidade de tempo. A simulação ocorrerá durante essa

quantidade da unidade especificada. Por exemplo, `20 MINUTE` resultará na execução de uma simulação de 20 minutos.

> **Note**
>
> Especifique o intervalo de tempo do evento de falha da réplica do Aurora; com cautela. Se você especificar um intervalo de tempo muito longo e a instância de gravador gravar uma grande quantidade de dados durante o evento de falha, o cluster de banco de dados do Aurora poderá presumir que a réplica do Aurora falhou e substituí-la.

Teste de uma falha de disco

Você pode simular a falha de um disco para um cluster de banco de dados do Aurora usando a consulta de injeção de falha `ALTER SYSTEM SIMULATE DISK FAILURE`.

Durante uma simulação de falha de disco, o cluster de banco de dados do Aurora marca aleatoriamente segmentos do disco como falhos. As solicitações feitas a esses segmentos serão bloqueadas enquanto durar a simulação.

Sintaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

- **`percentage_of_failure`** — a porcentagem do disco para marcar como falha durante o evento de falha. Esse valor pode ser um duplo entre 0 e 100. Se você especificar 0, nenhum dos discos será marcado como falhos. Se você especificar 100, o disco todo será marcado como falho.
- **`DISK index`** — um bloco de dados lógico específico para simular o evento de falha. Se você exceder o intervalo de blocos lógicos de dados disponíveis, receberá um erro informando o valor máximo do índice que você pode especificar. Para obter mais informações, consulte Exibir o status do volume para um cluster de banco de dados Aurora MySQL.

- **`NODE index`** — um nó de armazenamento específico para simular o evento de falha. Se você exceder o intervalo de nós de armazenamento disponíveis, receberá um erro informando o valor máximo do índice que você pode especificar. Para obter mais informações, consulte Exibir o status do volume para um cluster de banco de dados Aurora MySQL.
- **`quantity`** — o tempo durante o qual a falha do disco será simulada. O intervalo é uma quantidade seguida por uma unidade de tempo. A simulação ocorrerá durante essa quantidade da unidade especificada. Por exemplo, `20 MINUTE` resultará na execução de uma simulação de 20 minutos.

Teste de congestionamento de disco

Você pode simular a falha de um disco para um cluster de banco de dados do Aurora usando a consulta de injeção de falha `ALTER SYSTEM SIMULATE DISK CONGESTION`.

Durante uma simulação de congestionamento de disco, o cluster de banco de dados do Aurora marca aleatoriamente segmentos do disco como congestionados. As solicitações feitas a esses segmentos serão atrasadas entre o tempo de atraso mínimo e máximo especificado enquanto durar a simulação.

Sintaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

- **`percentage_of_failure`** — a porcentagem do disco para marcar como congestionada durante o evento de falha. Esse valor pode ser um duplo entre 0 e 100. Se você especificar 0, nenhum dos discos será marcado como congestionados. Se você especificar 100, o disco todo será marcado como congestionado.
- **`DISK index`** ou **`NODE index`** — um disco ou nó específico para simular o evento de falha. Se você exceder o intervalo de índices de disco ou de nó, receberá um erro informando o valor máximo do índice que você pode especificar.

- **`minimum`** e **`maximum`** — a quantidade mínima e máxima de atraso de congestionamento em milissegundos. Os segmentos de disco marcados como congestionados serão atrasados durante um período aleatório de tempo dentro do intervalo mínimo e máximo de milissegundos enquanto durar a simulação.
- **`quantity`** — o tempo durante o qual o congestionamento do disco será simulado. O intervalo é uma quantidade seguida por uma unidade de tempo. A simulação ocorrerá durante essa quantidade da unidade de tempo especificada. Por exemplo, `20 MINUTE` resultará na execução de uma simulação de 20 minutos.

Alterar tabelas no Amazon Aurora usando a DDL rápida

O Amazon Aurora inclui otimizações para executar uma operação de `ALTER TABLE` localmente, de maneira quase instantânea. A operação é concluída sem exigir que a tabela seja copiada e sem causar impacto material em outras instruções de DML. Como a operação não consome armazenamento temporário para a cópia da tabela, as instruções de DDL são práticas mesmo para tabelas grandes em classes de instâncias pequenas.

O Aurora MySQL versão 3 é compatível com o recurso do MySQL 8.0 chamado DDL instantânea. O Aurora MySQL versão 2 utiliza uma implementação diferente, chamada DDL rápida.

Tópicos

- DDL instantânea (Aurora MySQL versão 3)
- DDL rápida (Aurora MySQL versão 2)

DDL instantânea (Aurora MySQL versão 3)

A otimização realizada pelo Aurora MySQL versão 3 para melhorar a eficiência de algumas operações de DDL é chamada de DDL instantânea.

O Aurora MySQL versão 3 é compatível com a DDL instantânea da comunidade do MySQL 8.0. Você realiza uma operação de DDL instantânea utilizando a cláusula `ALGORITHM=INSTANT` com a instrução `ALTER TABLE`. Para obter detalhes de sintaxe e uso sobre a DDL instantânea, consulte ALTER TABLE e Operações online de DDL na documentação do MySQL.

Os seguintes exemplos demonstram o recurso de DDL instantânea. As instruções `ALTER TABLE` adicionam colunas e modificam valores de colunas padrão. Os exemplos incluem colunas regulares

e virtuais, bem como tabelas regulares e particionadas. Em cada etapa, é possível ver os resultados emitindo instruções `SHOW CREATE TABLE` e `DESCRIBE`.

```
mysql> CREATE TABLE t1 (a INT, b INT, KEY(b)) PARTITION BY KEY(b) PARTITIONS 6;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t1 RENAME TO t2, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b SET DEFAULT 100, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b DROP DEFAULT, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN c ENUM('a', 'b', 'c'), ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 MODIFY COLUMN c ENUM('a', 'b', 'c', 'd', 'e'), ALGORITHM =
 INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN (d INT GENERATED ALWAYS AS (a + 1) VIRTUAL), ALGORITHM
 = INSTANT;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t2 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3 (a INT, b INT) PARTITION BY LIST(a)(
    ->   PARTITION mypart1 VALUES IN (1,3,5),
    ->   PARTITION MyPart2 VALUES IN (2,4,6)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE t3 ALTER COLUMN a SET DEFAULT 20, ALTER COLUMN b SET DEFAULT 200,
 ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t4 (a INT, b INT) PARTITION BY RANGE(a)
    ->   (PARTITION p0 VALUES LESS THAN(100), PARTITION p1 VALUES LESS THAN(1000),
    ->   PARTITION p2 VALUES LESS THAN MAXVALUE);
```

```
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE t4 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

/* Sub-partitioning example */
mysql> CREATE TABLE ts (id INT, purchased DATE, a INT, b INT)
    ->   PARTITION BY RANGE( YEAR(purchased) )
    ->     SUBPARTITION BY HASH( TO_DAYS(purchased) )
    ->     SUBPARTITIONS 2 (
    ->       PARTITION p0 VALUES LESS THAN (1990),
    ->       PARTITION p1 VALUES LESS THAN (2000),
    ->       PARTITION p2 VALUES LESS THAN MAXVALUE
    ->    );
Query OK, 0 rows affected (0.10 sec)

mysql> ALTER TABLE ts ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

DDL rápida (Aurora MySQL versão 2)

A DDL rápida no Aurora MySQL é uma otimização projetada para melhorar o desempenho de determinadas alterações de esquema, como adicionar ou remover colunas, reduzindo o tempo de inatividade e o uso de recursos. Ela permite que essas operações sejam concluídas com mais eficiência em comparação com os métodos tradicionais de DDL.

> **Important**
>
> No momento, é necessário habilitar o modo de laboratório do Aurora para usar a DDL rápida. Consulte informações sobre como habilitar o modo de laboratório em Modo de laboratório do Amazon Aurora MySQL.
>
> A otimização da DDL rápida foi introduzida inicialmente no modo de laboratório no Aurora MySQL versão 2 para melhorar a eficiência de certas operações de DDL. No Aurora MySQL versão 3, o modo de laboratório foi descontinuado, e a DDL rápida foi substituída pelo recurso de DDL instantânea do MySQL 8.0.

No MySQL, muitas operações de linguagem de definição de dados (DDL) exercem um impacto significativo na performance.

Por exemplo, suponha que você usará uma operação `ALTER TABLE` para adicionar uma coluna a uma tabela. Dependendo do algoritmo especificado para a operação, esta operação pode envolver o seguinte:

- A criação de uma cópia completa da tabela
- A criação de uma tabela temporária para processar operações simultâneas de linguagem de manipulação de dados (DML)
- A reconstrução de todos os índices da tabela
- A aplicação de bloqueios à tabela ao fazer alterações de DML simultâneos
- O desaceleramento da taxa de transferência de DML simultâneos

Esse impacto no desempenho pode ser particularmente desafiador em ambientes com tabelas grandes ou altos volumes de transações. A DDL rápida ajuda a mitigar esses desafios otimizando as alterações de esquema, o que permite operações mais rápidas e menos intensivas em recursos.

Limitações de DDL rápido

No momento, a DDL rápida apresenta as seguintes limitações:

- O DDL rápido oferece suporte somente à adição de colunas anuláveis, sem valores padrão, ao final de uma tabela existente.
- O Fast DDL não funciona com tabelas particionadas.
- O DDL rápido não funciona com tabelas do InnoDB que usam o formato de linha REDUNDANT.
- O Fast DDL não funciona com tabelas com índices de pesquisa de texto completo.
- Se o tamanho de registro possível máximo para a operação de DDL for muito grande, a DDL rápida não será utilizada. O registro será muito grande se for maior do que a metade do tamanho da página. O tamanho máximo de um registro é calculado adicionando os tamanhos máximos de todas as colunas. Para colunas de tamanho variado, de acordo com os padrões InnoDB, bytes extern não são incluídos para computação.

Sintaxe DDL rápido

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Esta instrução apresenta as seguintes opções:

- **`tbl_name`** — o nome da tabela a ser modificada.
- **`col_name`** — o nome da coluna a ser adicionada.
- **`col_definition`** — a definição da coluna a ser adicionada.

> **Note**
>
> Você deve especificar uma definição de coluna anulável sem um valor padrão. Caso contrário, a DDL rápida não será usada.

Exemplos DDL rápido

Os exemplos a seguir demonstram a aceleração de operações de DDL rápida. O primeiro exemplo SQL executa instruções `ALTER TABLE` em uma tabela grande sem usar a DDL rápida. Esta operação leva um tempo considerável. Um exemplo da CLI mostra como permitir a DDL rápida para o cluster. Em seguida, outro exemplo SQL executa as mesmas instruções `ALTER TABLE` em uma tabela idêntica. Com a DDL rápida habilitada, a operação é muito rápida.

Este exemplo usa a tabela de`ORDERS` do benchmark TPC-H, contendo 150 milhões de linhas. Este cluster usa intencionalmente uma classe de instância relativamente pequena, para demonstrar quanto tempo as instruções `ALTER TABLE` podem demorar quando você não pode usar a DDL rápida. O exemplo cria um clone da tabela original contendo dados idênticos. Verificar a configuração de `aurora_lab_mode` confirma que o cluster não pode usar a DDL rápida, porque o modo de laboratório não está habilitado. Em seguida, as instruções de `ALTER TABLE ADD COLUMN` levam um tempo considerável para adicionar novas colunas no final da tabela.

```
mysql> create table orders_regular_ddl like orders;
Query OK, 0 rows affected (0.06 sec)

mysql> insert into orders_regular_ddl select * from orders;
Query OK, 150000000 rows affected (1 hour 1 min 25.46 sec)

mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 0 |
+-------------------+
```

```
mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (40 min 31.41 sec)

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (40 min 44.45 sec)
```

Este exemplo faz a mesma preparação de uma tabela grande que o exemplo anterior. No entanto, você não pode simplesmente habilitar o modo de laboratório em uma sessão SQL interativa. Essa configuração deve ser ativada em um grupo de parâmetro personalizado. Isso requer a troca da sessão `mysql` e a execução de alguns comandos de CLI da AWS ou usar o Console de gerenciamento da AWS.

```
mysql> create table orders_fast_ddl like orders;
Query OK, 0 rows affected (0.02 sec)

mysql> insert into orders_fast_ddl select * from orders;
Query OK, 150000000 rows affected (58 min 3.25 sec)

mysql> set aurora_lab_mode=1;
ERROR 1238 (HY000): Variable 'aurora_lab_mode' is a read only variable
```

Habilitar o modo de laboratório para o cluster requer algum trabalho com um grupo de parâmetro. Este exemplo de AWS CLI usa um grupo de parâmetros de cluster, para garantir que todas as instâncias de banco de dados no cluster usem o mesmo valor para a configuração do modo de laboratório.

```
$ aws rds create-db-cluster-parameter-group \
  --db-parameter-group-family aurora5.7 \
    --db-cluster-parameter-group-name lab-mode-enabled-57 --description 'TBD'
$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].[ParameterName,ParameterValue]' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 0
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --parameters ParameterName=aurora_lab_mode,ParameterValue=1,ApplyMethod=pending-
reboot
{
    "DBClusterParameterGroupName": "lab-mode-enabled-57"
```

```
}

# Assign the custom parameter group to the cluster that's going to use Fast DDL.
$ aws rds modify-db-cluster --db-cluster-identifier tpch100g \
  --db-cluster-parameter-group-name lab-mode-enabled-57
{
  "DBClusterIdentifier": "tpch100g",
  "DBClusterParameterGroup": "lab-mode-enabled-57",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.2",
  "Status": "available"
}

# Reboot the primary instance for the cluster tpch100g:
$ aws rds reboot-db-instance --db-instance-identifier instance-2020-12-22-5208
{
  "DBInstanceIdentifier": "instance-2020-12-22-5208",
  "DBInstanceStatus": "rebooting"
}

$ aws rds describe-db-clusters --db-cluster-identifier tpch100g \
  --query '*[].[DBClusterParameterGroup]' --output text
lab-mode-enabled-57

$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 1
```

O exemplo a seguir mostra as etapas restantes depois que a alteração do grupo de parâmetro tem efeito. Ele testa a configuração de `aurora_lab_mode` para certificar-se de que o cluster pode usar a DDL rápida. Em seguida, ele executa as instruções de `ALTER TABLE` para adicionar colunas ao final de outra tabela grande. Desta vez, as instruções terminam muito rapidamente.

```
mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 1 |
+-------------------+

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_refunded boolean;
```

```
Query OK, 0 rows affected (1.51 sec)

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (0.40 sec)
```

Exibir o status do volume para um cluster de banco de dados Aurora MySQL

No Amazon Aurora, um volume de cluster de banco de dados consiste em um acervo de blocos lógicos. Cada um deles representa 10 gigabytes de armazenamento alocado. Esses blocos são chamados de grupos de proteção.

Os dados em cada grupo de proteção são replicados entre seis dispositivos de armazenamento físico, chamados de nós de armazenamento. Esses nós de armazenamento são alocados em três zonas de disponibilidade (AZs) na região AWS em que reside o cluster de banco de dados. Por sua vez, cada nó de armazenamento contém um ou mais blocos lógicos de dados para o volume do cluster de banco de dados. Para obter mais informações sobre os grupos de proteção e os nós de armazenamento, consulte Introducing the Aurora storage engine (Apresentação do mecanismo de armazenamento do Aurora) no blog de banco de dados da AWS.

Você pode simular a falha de um nó de armazenamento completo ou de um único bloco lógicos de dados dentro de um nó de armazenamento. Para fazer isso, use a instrução de injeção de falha `ALTER SYSTEM SIMULATE DISK FAILURE`. Para a instrução, especifique o valor do índice de um bloco lógico de dados ou de um nó de armazenamento específico. No entanto, se você especificar um valor de índice maior do que o número de blocos lógicos de dados ou de nós de armazenamento usados pelo volume do cluster de banco de dados, a instrução retornará um erro. Para obter mais informações sobre as consultas de injeção de falha, veja Testar o Amazon Aurora MySQL usando consultas de injeção de falhas.

Você pode evitar esse erro usando a instrução `SHOW VOLUME STATUS`. A instrução retorna duas variáveis de status do servidor, `Disks` e `Nodes`. Essas variáveis representam o número total de blocos lógicos de dados e de nós de armazenamento, respectivamente, para o volume do cluster de banco de dados.

Sintaxe

```
SHOW VOLUME STATUS
```

Exemplo

O exemplo a seguir ilustra um resultado típico de SHOW VOLUME STATUS.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
+---------------+-------+
```

Ajustar o Aurora MySQL

Eventos de espera e estados de thread são ferramentas de ajuste importantes para o Aurora MySQL. Se você puder descobrir por que as sessões estão aguardando recursos e o que elas estão fazendo, poderá reduzir melhor os gargalos. Use as informações nesta seção para encontrar possíveis causas e ações corretivas.

O Amazon DevOps Guru para RDS pode determinar proativamente se seus bancos de dados do Aurora MySQL estão enfrentando condições problemáticas que poderão causar problemas maiores posteriormente. O Amazon DevOps Guru para RDS publica uma explicação e recomendações para ações corretivas em um insight proativo. Esta seção contém insights sobre problemas comuns.

> **Important**
>
> Os eventos de espera e os estados de thread nesta seção são específicos para o Aurora MySQL. Use as informações nesta seção para ajustar somente o Amazon Aurora, e não o Amazon RDS for MySQL.
> Alguns eventos de espera nesta seção não têm análogos nas versões de código aberto desses mecanismos de banco de dados. Outros eventos de espera têm os mesmos nomes que os eventos em mecanismos de código aberto, mas se comportam de maneira diferente. Por exemplo, o armazenamento do Amazon Aurora funciona de maneira diferente do armazenamento de código aberto e, portanto, eventos de espera relacionados a armazenamento indicam condições de recursos diferentes.

Tópicos

- Conceitos essenciais para ajuste do Aurora MySQL
- Ajustar o Aurora MySQL com eventos de espera
- Ajustar o Aurora MySQL com estados de threads
- Ajustar o Aurora MySQL com insights proativos do Amazon DevOps Guru

Conceitos essenciais para ajuste do Aurora MySQL

Antes de ajustar seu banco de dados Aurora MySQL, certifique-se de conhecer quais são os eventos de espera e estados de thread e por que eles estão ocorrendo. Analise também a arquitetura básica de memória e disco do Aurora MySQL ao utilizar o mecanismo de armazenamento InnoDB. Para acessar um diagrama de arquitetura útil, consulte o Manual de referência do MySQL.

Tópicos

- Eventos de espera do Aurora MySQL
- Estados de threads do Aurora MySQL
- Memória do Aurora MySQL
- Processos do Aurora MySQL

Eventos de espera do Aurora MySQL

Um evento de espera indica um recurso pelo qual uma sessão está aguardando. Por exemplo, o evento de espera `io/socket/sql/client_connection` indica que um thread está no processo de lidar com uma nova conexão. Os recursos comuns que uma sessão aguarda incluem:

- Acesso com thread único a um buffer, por exemplo, quando uma sessão está tentando modificar um buffer
- Uma linha que está bloqueada por outra sessão
- Uma leitura de arquivo de dados
- Uma gravação em arquivo de log

Por exemplo, para satisfazer uma consulta, a sessão pode realizar uma varredura de tabela completa. Se esses dados ainda não estiverem na memória, a sessão aguardará a conclusão da E/S do disco. Quando os buffers são lidos na memória, talvez a sessão precise aguardar, pois outras sessões estão acessando os mesmos buffers. O banco de dados registra as esperas utilizando um evento de espera predefinido. Esses eventos estão agrupados em categorias.

Por si só, um evento de espera não mostra um problema de performance. Por exemplo, se os dados solicitados não estão na memória, é necessário ler dados do disco. Se uma sessão bloquear uma linha para uma atualização, outra sessão aguardará que essa linha seja desbloqueada para poder atualizá-la. Uma confirmação exige a conclusão da gravação em um arquivo de log. Esperas são componentes integrais do funcionamento normal de um banco de dados.

Em geral, muitos eventos de espera indicam um problema de performance. Nesses casos, é possível utilizar os dados dos eventos de espera para determinar onde as sessões estão perdendo tempo. Por exemplo, se um relatório que é normalmente executado em minutos passou a demorar várias horas, é possível identificar os eventos de espera que mais contribuem para o tempo de espera total. Se você puder determinar as causas dos principais eventos de espera, às vezes pode aplicar

alterações que melhoram a performance. Por exemplo, se a sua sessão está aguardando uma linha que foi bloqueada por outra sessão, é possível encerrar a sessão responsável pelo bloqueio.

Estados de threads do Aurora MySQL

Um estado geral de thread é um valor `State` associado ao processamento geral de consultas. Por exemplo, o estado de thread `sending data` indica que um thread está lendo e filtrando linhas de uma consulta para determinar o conjunto de resultados correto.

É possível utilizar estados de thread para ajustar o Aurora MySQL de maneira semelhante a como você utiliza eventos de espera. Por exemplo, ocorrências frequentes de `sending data` em geral indicam que uma consulta não está utilizando um índice. Para saber mais sobre estados de thread, consulte o tópico sobre Estados gerais de thread, no Manual de referência do MySQL.

Quando você utiliza o Performance Insights, uma das seguintes condições é verdadeira:

- O Performance Schema está habilitado: o Aurora MySQL mostra eventos de espera em vez do estado de thread.
- O Performance Schema não está habilitado: o Aurora MySQL mostra o estado de thread.

Convém configurar o Performance Schema para gerenciamento automático. O Performance Schema fornece insights adicionais e melhores ferramentas para investigar possíveis problemas de performance. Para obter mais informações, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Memória do Aurora MySQL

No Aurora MySQL, as áreas de memória mais importantes são o grupo de buffer e o buffer de logs.

Tópicos

- Grupo de buffer

Grupo de buffer

O Grupo de buffer é a área de memória compartilhada na qual o Aurora MySQL armazena dados de tabela e índice em cache. As consultas podem acessar dados utilizados com frequência diretamente da memória sem fazer a leitura do disco.

O grupo de buffer está estruturado como uma lista vinculada de páginas. Uma página pode incluir várias linhas. O Aurora MySQL utiliza um algoritmo menos utilizado recentemente (LRU) para deixar páginas fora do grupo.

Para obter mais informações, consulte os tópicos sobre Grupo de buffer no Manual de referência do MySQL.

Processos do Aurora MySQL

O Aurora MySQL utiliza um modelo de processos muito diferente do Aurora PostgreSQL.

Tópicos

- Servidor MySQL (mysqld)
- Threads
- Pool de threads

Servidor MySQL (mysqld)

O servidor MySQL é um único processo de sistema operacional denominado mysqld. Ele não gera processos adicionais. Portanto, um banco de dados Aurora MySQL utiliza mysqld para realizar a maior parte do trabalho.

Quando o servidor MySQL é iniciado, ele escuta conexões de rede de clientes MySQL. Quando um cliente se conecta ao banco de dados, o mysqld abre um thread.

Threads

Os threads do gerenciador de conexões associam cada conexão de cliente a um thread dedicado. Esse thread gerencia a autenticação, executa instruções e retorna os resultados ao cliente. O Gerenciador de conexões cria novos threads quando necessário.

O cache de threads é o conjunto de threads disponíveis. Quando uma conexão é encerrada, o MySQL retorna o thread ao cache de threads quando este não está cheio. A variável de sistema `thread_cache_size` determina o tamanho do cache de threads.

Pool de threads

O pool de threadsconsiste em vários grupos de threads. Cada grupo gerencia um conjunto de conexões de clientes. Quando um cliente se conecta ao banco de dados, o pool de threads atribui

as conexões aos grupos de threads de maneira bidirecional. O pool de threads separa conexões e threads. Não há relação fixa entre conexões e os threads que executam instruções delas recebidas.

Ajustar o Aurora MySQL com eventos de espera

A tabela a seguir resume os eventos de espera do Aurora MySQL que costuma indicar problemas de performance. Os eventos de espera a seguir representam um subconjunto da lista em Eventos de espera do Aurora MySQL.

Eventos de espera	Descrição
cpu	Ocorre quando um thread está ativo na CPU ou está aguardando a CPU.
io/aurora_redo_log_flush	Ocorre quando uma sessão está gravando dados persistentes no armazenamento do Aurora.
io/aurora_respond_to_client	Ocorre quando um thread está aguardando para retornar um conjunto de resultados a um cliente.
io/redo_log_flush	Ocorre quando uma sessão está gravando dados persistentes no armazenamento do Aurora.
io/socket/sql/client_connection	Ocorre quando um segmento está em processo de manipulação de uma nova conexão.
io/table/sql/handler	Ocorre quando o trabalho foi delegado a um mecanismo de armazenamento.
synch/cond/innodb/row_lock_wait	Ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha.

Eventos de espera	Descrição
synch/cond/innodb/row_lock_wait_cond	Ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha.
synch/cond/sql/MDL_context::COND_wait_status	Ocorre quando há threads aguardando em um bloqueio de metadados de tabela.
synch/mutex/innodb/aurora_lock_thread_slot_futex	Ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha.
synch/mutex/innodb/buf_pool_mutex	Ocorre quando um thread adquire um bloqueio no grupo de buffers do InnoDB para acessar uma página na memória.
synch/mutex/innodb/fil_system_mutex	Ocorre quando uma sessão está aguardando para acessar o cache de memória do espaço de tabelas.
synch/mutex/innodb/trx_sys_mutex	Ocorre quando há alta atividade de banco de dados com muitas transações.
synch/sxlock/innodb/hash_table_locks	Ocorre quando as páginas não encontradas no grupo de buffer devem ser lidas de um arquivo.
synch/mutex/innodb/temp_pool_manager_mutex	Esse evento ocorre quando uma sessão está aguardando para adquirir um mutex para o grupo de espaços para tabela temporários de sessão.

cpu

O evento de espera `cpu` ocorre quando um thread está ativo na CPU ou está aguardando a CPU.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

Para cada vCPU, uma conexão pode executar o trabalho nessa CPU. Em algumas situações, o número de conexões ativas que estão prontas para execução é superior ao número de vCPUs. Esse desequilíbrio faz com que conexões aguardem recursos da CPU. Se o número de conexões ativas permanecer consistentemente superior ao número de vCPUs, sua instância enfrentará disputa de CPU. A disputa faz com que o evento de espera `cpu` ocorra.

> **Note**
>
> A métrica do Performance Insights para CPU é `DBLoadCPU`. O valor de `DBLoadCPU` pode diferir do valor da métrica `CPUUtilization` do CloudWatch. A última métrica é coletada do HyperVisor para uma instância de banco de dados.

As métricas do Performance Insights para o sistema operacional fornecem informações detalhadas sobre a utilização da CPU. Por exemplo, é possível exibir as seguintes métricas:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

O Performance Insights relata o uso da CPU pelo mecanismo de banco de dados como `os.cpuUtilization.nice.avg`.

Possíveis causas do maior número de esperas

Quando esse evento ocorre mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

- Consultas analíticas
- Transações altamente simultâneas
- Transações de longa execução
- Um aumento súbito no número de conexões, conhecido como tempestade de logins
- Um aumento na alternância de contexto

Ações

Se o evento de espera `cpu` domina a atividade do banco de dados, isso não indica necessariamente um problema de performance. Responda a esse evento somente quando a performance diminuir.

Dependendo do motivo para o aumento na utilização da CPU, considere as seguintes estratégias:

- Aumente a capacidade de CPU do host. Em geral, essa abordagem apenas fornece alívio temporário.
- Identifique as principais consultas para uma possível otimização.
- Redirecione uma parte da workload somente leitura para nós de leitor, se aplicável.

Tópicos

- Identifique as sessões ou consultas que estão causando o problema
- Analisar e otimizar a workload alta da CPU

Identifique as sessões ou consultas que estão causando o problema

Para encontrar as sessões e consultas, consulte a tabela Top SQL (SQL principal) no Performance Insights em busca das instruções SQL que apresentam a maior carga de CPU. Para obter mais informações, consulte Análise de métricas usando o painel do Performance Insights.

Em geral, uma ou duas instruções SQL consomem a maioria dos ciclos de CPU. Concentre seus esforços nessas instruções. Suponha que a sua instância de banco de dados tenha 2 vCPUs com

carga de banco de dados de 3,1 sessões ativas médias (AAS), todas no estado da CPU. Nesse caso, a instância está vinculada à CPU. Considere as estratégias a seguir:

- Faça upgrade para uma classe de instância maior com mais vCPUs.
- Ajuste as consultas para ter uma carga de CPU mais baixa.

Neste exemplo, as principais consultas SQL têm uma carga de banco de dados de 1,5 AAS, todas no estado da CPU. Outra instrução SQL tem uma carga de 0,1 no estado da CPU. Neste exemplo, se você interromper a instrução SQL de menor carga, não reduzirá significativamente a carga do banco de dados. Porém, se você otimizar as duas consultas de alta carga de forma que elas sejam duas vezes mais eficientes, eliminará o gargalo da CPU. Se você reduzir a carga da CPU de 1,5 AAS em 50%, a AAS para cada instrução diminuirá para 0,75. A carga total do banco de dados gasta na CPU agora é de 1,6 AAS. Esse valor está abaixo da linha máxima de vCPU de 2,0.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights. Consulte também o artigo do AWS Support sobre Como solucionar problemas e resolver a alta utilização da CPU em instâncias do Amazon RDS for MySQL.

Analisar e otimizar a workload alta da CPU

Depois de identificar uma ou mais consultas que aumentam o uso da CPU, é possível otimizá-las ou encerrar a conexão. O exemplo a seguir mostra como encerrar uma conexão.

```
CALL mysql.rds_kill(processID);
```

Para obter mais informações, consulte mysql.rds_kill.

Se você encerrar uma sessão, essa ação poderá disparar uma longa reversão.

Siga as diretrizes para otimizar consultas

Para otimizar consultas, considere estas diretrizes:

- Execute a instrução `EXPLAIN`.

  Esse comando mostra as etapas individuais envolvidas na execução de uma consulta. Para obter mais informações, consulte o tópico sobre como Otimizar operações de bloqueio, na documentação do MySQL.
- Execute a instrução `SHOW PROFILE`.

Use essa instrução para rever detalhes de perfil que podem indicar o uso de recursos para instruções executadas durante a sessão atual. Para obter mais informações, consulte o tópico sobre a Instrução SHOW PROFILE, na documentação do MySQL.

- Execute a instrução `ANALYZE TABLE`.

  Use essa instrução para atualizar as estatísticas de índices das tabelas acessadas pela consulta com alto consumo de CPU. Analisando a instrução, você pode ajudar o otimizador a escolher um plano de execução apropriado. Para obter mais informações, consulte o tópico sobre a Instrução ANALYZE TABLE, na documentação do MySQL.

Siga as diretrizes para melhorar o uso da CPU

Para melhorar o uso da CPU em uma instância de banco de dados, siga estas diretrizes:

- Verifique se todas as consultas estão utilizando índices adequados.
- Descubra se é possível utilizar consultas paralelas do Aurora. Você pode usar essa técnica para reduzir o uso da CPU no nó principal, empurrando para baixo o processamento de funções, a filtragem de linhas e a projeção de colunas para a cláusula `WHERE`.
- Descubra se o número de execuções de SQL por segundo corresponde aos limites esperados.
- Descubra se a manutenção do índice ou a criação de novos índices ocupam os ciclos de CPU necessários para a sua workload de produção. Programe atividades de manutenção fora dos horários de pico de atividades.
- Descubra se é possível utilizar particionamentos para ajudar a reduzir o conjunto de dados da consulta. Para obter mais informações, leia a postagem de blog sobre Como planejar e otimizar o Amazon Aurora com compatibilidade com o MySQL para workloads consolidadas.

Verifique se há tempestades de conexões

Se a métrica `DBLoadCPU` não for muito alta, mas a métrica `CPUUtilization` for muito alta, a causa da alta utilização da CPU é externa ao mecanismo de banco de dados. Um exemplo clássico é uma tempestade de conexões.

Verifique se as seguintes condições são válidas:

- Há um aumento simultâneo na métrica `CPUUtilization` do Performance Insights e na métrica `DatabaseConnections` do Amazon CloudWatch.

- O número de threads na CPU é maior que o número de vCPUs.

Se as condições anteriores forem verdadeiras, diminua o número de conexões de banco de dados. Por exemplo, é possível utilizar um grupo de conexões, como o RDS Proxy. Para conhecer as práticas recomendadas de gerenciamento e escalabilidade eficientes de conexões, consulte o whitepaper Amazon Aurora MySQL DBA Handbook for Connection Management.

io/aurora_redo_log_flush

O evento `io/aurora_redo_log_flush` ocorre quando uma sessão está gravando dados persistentes no armazenamento do Amazon Aurora.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 2

Contexto

O evento `io/aurora_redo_log_flush` refere-se a uma operação de entrada/saída (E/S) no Aurora MySQL.

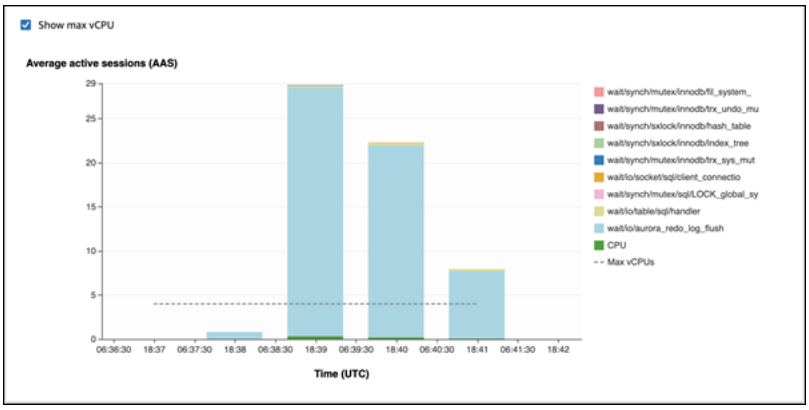
> **Note**
>
> No Aurora MySQL versão 3, esse evento de espera é denominado io/redo_log_flush.

Possíveis causas do maior número de esperas

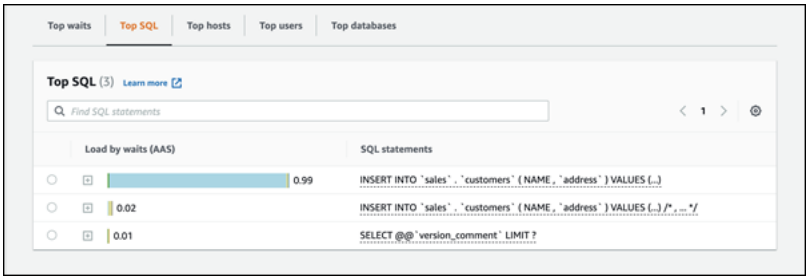
Para persistência de dados, as confirmações exigem uma gravação durável no armazenamento estável. Se o banco de dados estiver fazendo muitas confirmações, significa que há um evento de espera na operação de E/S de gravação, o evento de espera `io/aurora_redo_log_flush`.

Nos exemplos a seguir, 50.000 registros são inseridos em um cluster de banco de dados Aurora MySQL com a classe de instância de banco de dados db.r5.xlarge:

- No primeiro exemplo, cada sessão insere 10.000 registros, linha por linha. Por padrão, se um comando de linguagem de manipulação de dados (DML) não estiver em uma transação, o Aurora MySQL utilizará confirmações implícitas. A confirmação automática está habilitada. Isso significa que há uma confirmação para cada inserção de linha. O Performance Insights mostra que as conexões passam a maior parte do tempo esperando pelo evento de espera `io/aurora_redo_log_flush`.

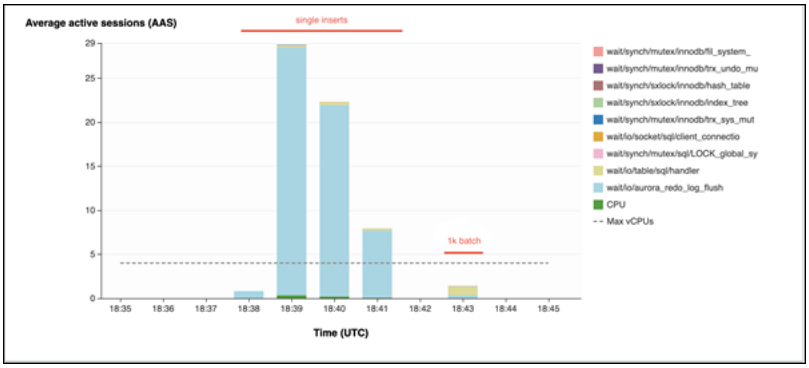


  Isso é causado pelas instruções de inserção simples utilizadas.



  Os 50.000 registros demoram 3,5 minutos para serem inseridos.

- No segundo exemplo, as inserções são feitas em 1.000 lotes, ou seja, cada conexão realiza 10 confirmações em vez de 10.000. O Performance Insights mostra que as conexões não passam a maior parte do tempo no evento de espera `io/aurora_redo_log_flush`.



Os 50.000 registros demoram 4 segundos para serem inseridos.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e consultas problemáticas
- Agrupar suas operações de gravação
- Desativar a confirmação automática
- Utilizar transações
- Utilizar lotes

Identificar as sessões e consultas problemáticas

Se a sua instância de banco de dados estiver enfrentando um gargalo, a primeira tarefa é encontrar as sessões e consultas responsáveis. Leia esta útil postagem no blog de banco dados da AWS sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Para identificar sessões e consultas que estão causam um gargalo

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha a instância de banco de dados.
4. Em Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

As consultas na parte superior da lista estão causando a maior carga no banco de dados.

Agrupar suas operações de gravação

Os exemplos a seguir acionam o evento de espera `io/aurora_redo_log_flush`. (A confirmação automática está habilitada.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Para reduzir o tempo gasto esperando no evento de espera `io/aurora_redo_log_flush`, agrupe suas operações de gravação logicamente em uma única confirmação para reduzir chamadas persistentes ao armazenamento.

Desativar a confirmação automática

Desative a confirmação automática antes de fazer grandes alterações que não estejam em uma transação, conforme mostrado no exemplo a seguir.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
```

```
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utilizar transações

É possível utilizar transações, conforme mostrado no exemplo a seguir.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizar lotes

Você pode fazer alterações em lotes, conforme mostrado no exemplo a seguir. No entanto, o uso de lotes muito grandes pode causar problemas de performance, especialmente em réplicas de leitura ou ao fazer uma recuperação em um ponto anterior no tempo (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/aurora_respond_to_client

O evento `io/aurora_respond_to_client` quando um thread está aguardando para retornar um conjunto de resultados a um cliente.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 2

Contexto

O evento `io/aurora_respond_to_client` indica que um thread está aguardando para retornar um conjunto de resultados a um cliente.

O processamento da consulta está concluído, e os resultados estão sendo retornados ao cliente da aplicação. Porém, como não há largura de banda de rede suficiente no cluster de banco de dados, um thread está aguardando para retornar o conjunto de resultados.

Possíveis causas do maior número de esperas

Quando o evento `io/aurora_respond_to_client` aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Classe de instância de banco de dados insuficiente para a workload

A classe de instância de banco de dados utilizada pelo cluster de banco de dados não tem a largura de banda de rede necessária para processar a workload com eficiência.

Conjuntos de resultados grandes

Houve um aumento no tamanho do conjunto de resultados retornado porque a consulta retorna maiores números de linhas. O conjunto de resultados maior consome mais largura de banda de rede.

Maior carga no cliente

Pode haver pressão da CPU, pressão da memória ou saturação da rede no lado do cliente. Um aumento na carga do cliente atrasa o recebimento de dados do cluster de bancos de dados Aurora MySQL.

Maior latência de rede

Pode haver maior latência de rede entre o cluster de bancos de dados Aurora MySQL e o cliente. A latência de rede mais alta aumenta o tempo necessário para o cliente receber os dados.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Escalar a classe da instância de banco de dados
- Verificar se há resultados inesperados na workload
- Distribuir a workload com instâncias de leitor
- Usar o modificador SQL_BUFFER_RESULT

Identificar as sessões e as consultas que estão causando os eventos

É possível utilizar o Performance Insights para mostrar consultas bloqueadas pelo evento de espera `io/aurora_respond_to_client`. Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta e descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para localizar consultas SQL que são responsáveis pela carga alta

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.

4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a AWSPublicação no blog de banco de dados sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Escalar a classe da instância de banco de dados

Verifique se houve aumentos no valor das métricas do Amazon CloudWatch relacionadas à taxa de transferência da rede, como `NetworkReceiveThroughput` e `NetworkTransmitThroughput`. Se a largura de banda da rede da classe da instância de banco de dados estiver sendo atingida, você poderá escalar a classe da instância de banco de dados utilizada pelo cluster de banco de dados modificando o cluster de banco de dados. Uma classe de instância de banco de dados com largura de banda de rede maior retorna dados aos clientes de maneira mais eficiente.

Para obter informações sobre o monitoramento de métricas do Amazon CloudWatch, consulte Visualizar métricas no console do Amazon RDS. Para obter informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora. Para ter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Verificar se há resultados inesperados na workload

Verifique a workload no cluster de banco de dados e certifique-se de que ela não esteja gerando resultados inesperados. Por exemplo, pode haver consultas retornando um número maior de linhas que o esperado. Nesse caso, é possível utilizar métricas de contadores do Performance Insights, como `Innodb_rows_read`. Para obter mais informações, consulte Métricas de contadores do Performance Insights.

Distribuir a workload com instâncias de leitor

É possível distribuir workloads somente leitura com réplicas do Aurora. É possível escalar horizontalmente adicionando mais réplicas do Aurora. Isso pode resultar em um aumento nos limites de controle de utilização para a largura de banda da rede. Para obter mais informações, consulte Clusters de banco de dados do Amazon Aurora.

Usar o modificador SQL_BUFFER_RESULT

Você pode adicionar o modificador `SQL_BUFFER_RESULT` a instruções `SELECT` para forçar o resultado em uma tabela temporária antes que eles sejam retornados ao cliente. Esse modificador pode ajudar com problemas de performance quando bloqueios do InnoDB não estão sendo liberados porque as consultas estão no estado de espera `io/aurora_respond_to_client`. Para obter mais informações, consulte o tópico sobre a Instrução SELECT, na documentação do MySQL.

io/redo_log_flush

O evento `io/redo_log_flush` ocorre quando uma sessão está gravando dados persistentes no armazenamento do Amazon Aurora.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 3

Contexto

O evento `io/redo_log_flush` refere-se a uma operação de entrada/saída (E/S) no Aurora MySQL.

> **Note**
>
> No Aurora MySQL versão 2, esse evento de espera é denominado io/aurora_redo_log_flush.

Possíveis causas do maior número de esperas

Para persistência de dados, as confirmações exigem uma gravação durável no armazenamento estável. Se o banco de dados estiver fazendo muitas confirmações, significa que há um evento de espera na operação de E/S de gravação, o evento de espera `io/redo_log_flush`.

Para ver exemplos do comportamento desse evento de espera, consulte io/aurora_redo_log_flush.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e consultas problemáticas
- Agrupar suas operações de gravação
- Desativar a confirmação automática
- Utilizar transações
- Utilizar lotes

Identificar as sessões e consultas problemáticas

Se a sua instância de banco de dados estiver enfrentando um gargalo, a primeira tarefa é encontrar as sessões e consultas responsáveis. Leia esta útil postagem no blog de banco dados da AWS sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Para identificar sessões e consultas que estão causam um gargalo

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha a instância de banco de dados.
4. Em Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   As consultas na parte superior da lista estão causando a maior carga no banco de dados.

Agrupar suas operações de gravação

Os exemplos a seguir acionam o evento de espera `io/redo_log_flush`. (A confirmação automática está habilitada.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Para reduzir o tempo gasto esperando no evento de espera `io/redo_log_flush`, agrupe suas operações de gravação logicamente em uma única confirmação para reduzir chamadas persistentes ao armazenamento.

Desativar a confirmação automática

Desative a confirmação automática antes de fazer grandes alterações que não estejam em uma transação, conforme mostrado no exemplo a seguir.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utilizar transações

É possível utilizar transações, conforme mostrado no exemplo a seguir.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizar lotes

Você pode fazer alterações em lotes, conforme mostrado no exemplo a seguir. No entanto, o uso de lotes muito grandes pode causar problemas de performance, especialmente em réplicas de leitura ou ao fazer uma recuperação em um ponto anterior no tempo (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/socket/sql/client_connection

O evento `io/socket/sql/client_connection` ocorre quando um segmento está em processo de manipulação de uma nova conexão.

Tópicos

- Versões compatíveis do mecanismo

- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

O evento `io/socket/sql/client_connection` indica que o mysqld está ocupado criando threads para lidar com novas conexões de cliente recebidas. Nesse cenário, o processamento do atendimento a novas solicitações de conexão de clientes diminui enquanto as conexões aguardam a atribuição do thread. Para obter mais informações, consulte Servidor MySQL (mysqld).

Possíveis causas do maior número de esperas

Quando esse evento aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

- Há um aumento repentino nas novas conexões de usuário da aplicação para a sua instância do Amazon RDS.
- Sua instância de banco de dados não consegue processar novas conexões porque a rede, a CPU ou a memória está sendo limitada.

Ações

Se `io/socket/sql/client_connection` domina a atividade do banco de dados, isso não indica necessariamente um problema de performance. Em um banco de dados não ocioso, um evento de espera está sempre na parte superior. Reaja apenas quando a performance piorar. Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e consultas problemáticas
- Siga as práticas recomendadas para o gerenciamento de conexões

- Aumentar a escala da sua instância na vertical se os recursos estiverem sendo limitados
- Conferir os principais hosts e usuários
- Consultar as tabelas performance_schema
- Verificar os estados de threads das suas consultas
- Auditar suas solicitações e consultas
- Agrupar suas conexões de banco de dados

Identificar as sessões e consultas problemáticas

Se a sua instância de banco de dados estiver enfrentando um gargalo, a primeira tarefa é encontrar as sessões e consultas responsáveis. Leia esta útil postagem de blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Para identificar sessões e consultas que estão causam um gargalo

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha a instância de banco de dados.
4. Em Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   As consultas na parte superior da lista estão causando a maior carga no banco de dados.

Siga as práticas recomendadas para o gerenciamento de conexões

Para gerenciar suas conexões, considere as seguintes estratégias:

- Use o agrupamento de conexões.

  É possível aumentar gradualmente o número de conexões conforme necessário. Para obter mais informações, consulte o whitepaper Amazon Aurora MySQL Database Administrator's Handbook.
- Use um nó de leitor para redistribuir o tráfego somente leitura.

  Para obter mais informações, consulte Réplicas do Aurora e Conexões de endpoints do Amazon Aurora.

Aumentar a escala da sua instância na vertical se os recursos estiverem sendo limitados

Procure exemplos de controle de utilização nos seguintes recursos:

- CPU

  Verifique suas métricas do Amazon CloudWatch com relação ao alto uso da CPU.
- Rede

  Verifique se há um aumento no valor das métricas `network receive throughput` e `network transmit throughput` do CloudWatch. Se sua instância tiver atingido o limite de largura de banda da rede da classe de instância, considere aumentar a escala vertical da sua instância do RDS para um tipo de classe de instância superior. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.
- Memória liberável

  Verifique se há uma queda na métrica do CloudWatch `FreeableMemory`. Além disso, considere habilitar o Monitoramento aprimorado. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.

Conferir os principais hosts e usuários

Use o Performance Insights para verificar os principais hosts e usuários. Para obter mais informações, consulte Análise de métricas usando o painel do Performance Insights.

Consultar as tabelas performance_schema

Para obter uma contagem precisa das conexões atuais e totais, consulte as tabelas `performance_schema`. Com essa técnica, você identifica o usuário ou host de origem responsável por criar um alto número de conexões. Por exemplo, consulte as tabelas `performance_schema` da seguinte maneira.

```
SELECT * FROM performance_schema.accounts;
SELECT * FROM performance_schema.users;
SELECT * FROM performance_schema.hosts;
```

Verificar os estados de threads das suas consultas

Se o problema de performance for contínuo, verifique os estados de thread das suas consultas. No cliente `mysql`, emita o comando a seguir.

```
show processlist;
```

Auditar suas solicitações e consultas

Para verificar a natureza das solicitações e consultas de contas de usuários, use o recurso de Auditoria avançada do Aurora MySQL. Para aprender a habilitar a auditoria, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.

Agrupar suas conexões de banco de dados

Considere utilizar o Amazon RDS Proxy para o gerenciamento das conexões. Com o RDS Proxy, você pode permitir que suas aplicações agrupem e compartilhem conexões de banco de dados para melhorar sua capacidade de escala. O proxy do RDS torna as aplicações mais resilientes a falhas de banco de dados conectando-se automaticamente a uma instância de banco de dados em espera e preservando as conexões de aplicações. Para obter mais informações, consulte Amazon RDS Proxypara Aurora.

io/table/sql/handler

O evento `io/table/sql/handler` ocorre quando o trabalho foi delegado a um mecanismo de armazenamento.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

O evento `io/table` indica uma espera pelo acesso a uma tabela. Esse evento ocorre não importa se os dados estão armazenados em cache no grupo de buffer ou acessados no disco. O evento `io/table/sql/handler` indica um aumento na atividade da workload.

Um manipulador é uma rotina especializada em um determinado tipo de dados ou focada em certas tarefas especiais. Por exemplo, um manipulador de eventos recebe e digere eventos e sinais provenientes do sistema operacional ou de uma interface de usuário. Um manipulador de memória realiza tarefas relacionadas a memória. Um manipulador de entrada de arquivo é uma função que recebe a entrada de arquivos e realiza tarefas especiais nos dados de acordo com o contexto.

Visualizações como `performance_schema.events_waits_current` muitas vezes indicam `io/table/sql/handler` quando a espera real é um evento de espera aninhado, como um bloqueio. Quando a espera real não é `io/table/sql/handler`, o Performance Insights relata o evento de espera aninhado. Quando o Performance Insights relata `io/table/sql/handler`, isso representa o processamento de InnoDB da solicitação de E/S e não um evento de espera aninhado oculto. Para saber mais, consulte o tópico sobre Eventos "átomo" e "molécula" no Performance Schema, no Manual de referência do MySQL.

O evento `io/table/sql/handler` costuma aparecer nos principais eventos de espera com esperas de E/S, como `io/aurora_redo_log_flush`.

Possíveis causas do maior número de esperas

No Performance Insights, picos repentinos no evento `io/table/sql/handler` indicam aumento na atividade da workload. O aumento da atividade significa E/S elevada.

O Performance Insights filtra os IDs de eventos de aninhamento e não informa uma espera `io/table/sql/handler` quando o evento aninhado subjacente é uma espera de bloqueio. Por exemplo, se o evento da causa raiz for synch/mutex/innodb/aurora_lock_thread_slot_futex, o Performance Insights exibirá essa espera nos principais eventos de espera, e não em `io/table/sql/handler`.

Em visualizações como `performance_schema.events_waits_current`, esperas por `io/table/sql/handler` geralmente aparecem quando a espera real é um evento de espera aninhado, como um bloqueio. Quando a espera real é diferente de `io/table/sql/handler`, o Performance Insights procura a espera aninhada e relata a espera real em vez de `io/table/sql/handler`. Quando o Performance Insights relata `io/table/sql/handler`, a espera real é `io/`

`table/sql/handler`, e não um evento de espera aninhado oculto. Para saber mais, consulte o tópico sobre Eventos "átomo" e "molécula" no Performance Schema, no Manual de referência do MySQL 5.7.

Ações

Se esse evento de espera domina a atividade do banco de dados, isso não indica necessariamente um problema de performance. Um evento de espera está sempre na parte superior quando o banco de dados está ativo. Você precisará reagir somente quando a performance piorar.

Recomendamos diferentes ações dependendo dos outros eventos de espera exibidos.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Verifique se existe uma correlação com as métricas de contadores do Performance Insights
- Verificar se existem outros eventos de espera correlacionados

Identificar as sessões e as consultas que estão causando os eventos

Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta e descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para localizar consultas SQL que são responsáveis pela carga alta

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Verifique se existe uma correlação com as métricas de contadores do Performance Insights

Verifique as métricas do contador do Performance Insights, como `Innodb_rows_changed`. Se as métricas de contadores estiverem correlacionadas com `io/table/sql/handler`, siga estas etapas:

1. No Performance Insights, procure as instruções SQL que representam o evento de espera principal `io/table/sql/handler`. Se possível, otimize essa instrução de forma que ela retorne menos linhas.
2. Recupere as principais tabelas das visualizações `schema_table_statistics` e `x $schema_table_statistics`. Essas visualizações mostram o tempo gasto por tabela. Para ter mais informações, consulte As visualizações schema_table_statistics e x$schema_table_statistics, no Manual de referência do MySQL.

   Por padrão, as linhas são classificadas por tempo de espera total em ordem decrescente. Tabelas com mais disputa aparecem primeiro. A saída indica se o tempo é gasto em buscas, inserções, leituras, gravações, atualizações ou exclusões.

```
mysql> select * from sys.schema_table_statistics limit 1\G

*************************** 1. row ***************************
     table_schema: read_only_db
       table_name: sbtest41
    total_latency: 54.11 m
     rows_fetched: 6001557
    fetch_latency: 39.14 m
    rows_inserted: 14833
   insert_latency: 5.78 m
     rows_updated: 30470
   update_latency: 5.39 m
     rows_deleted: 14833
   delete_latency: 3.81 m
 io_read_requests: NULL
          io_read: NULL
  io_read_latency: NULL
io_write_requests: NULL
         io_write: NULL
```

```
 io_write_latency: NULL
 io_misc_requests: NULL
  io_misc_latency: NULL
1 row in set (0.11 sec)
```

Verificar se existem outros eventos de espera correlacionados

Se `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` forem juntos os principais colaboradores para a anomalia de carga de banco de dados, verifique se a variável `innodb_adaptive_hash_index` está habilitada. Se estiver, considere aumentar o valor do parâmetro `innodb_adaptive_hash_index_parts`.

Se o Adaptive Hash Index estiver desabilitado, considere habilitá-lo. Para saber mais sobre o Adaptive Hash Index do MySQL, consulte os seguintes recursos:

- O artigo Is Adaptive Hash Index in InnoDB right for my workload?, no site da Percona
- Adaptive Hash Index, no Manual de referência do MySQL
- O artigo Contention in MySQL InnoDB: Useful Info From the Semaphores Section, no site da Percona

> **Note**
>
> O Adaptive Hash Index não é compatível com instâncias de banco de dados de leitor do Aurora.
> Em alguns casos, a performance pode ser ruim em uma instância de leitor quando `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` são dominantes. Em caso afirmativo, considere redirecionar a workload temporariamente à instância de banco de dados de gravador e habilitar o Adaptive Hash Index.

synch/cond/innodb/row_lock_wait

O evento `synch/cond/innodb/row_lock_wait` ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha. Para obter mais informações, consulte o tópico sobre InnoDB locking na documentação do MySQL.

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 3

Possíveis causas do maior número de esperas

Várias instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas simultaneamente.

Ações

Recomendamos diferentes ações dependendo dos outros eventos de espera exibidos.

Tópicos

- Localizar e responder às instruções SQL responsáveis por esse evento de espera
- Localizar e responder à sessão de bloqueio

Localizar e responder às instruções SQL responsáveis por esse evento de espera

Utilize o Performance Insights para identificar as instruções SQL responsáveis por esse evento de espera. Considere as estratégias a seguir:

- Se bloqueios de linha forem um problema persistente, considere reescrever a aplicação para utilizar o bloqueio otimista.
- Utilize instruções de várias linhas.
- Disperse a workload por objetos de banco de dados diferentes. É possível fazer isso por meio de particionamento.
- Verifique o valor do parâmetro `innodb_lock_wait_timeout`. Ele controla quanto tempo as transações aguardam antes de gerarem um erro de tempo limite.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Localizar e responder à sessão de bloqueio

Determine se a sessão de bloqueio está ociosa ou ativa. Além disso, descubra se a sessão é proveniente de uma aplicação ou de um usuário ativo.

Para identificar a sessão que está mantendo o bloqueio, é possível executar `SHOW ENGINE INNODB STATUS`. O exemplo a seguir mostra uma saída.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 1688153, ACTIVE 82 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 4244, OS thread handle 70369524330224, query id 4020834 172.31.14.179
 reinvent executing
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 24 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 4 n bits 72 index GEN_CLUST_INDEX of table test.t1 trx
 id 1688153 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

Outra alternativa é utilizar a consulta a seguir para extrair detalhes sobre os bloqueios atuais.

```
mysql> SELECT p1.id waiting_thread,
    p1.user waiting_user,
    p1.host waiting_host,
    it1.trx_query waiting_query,
    ilw.requesting_engine_transaction_id waiting_transaction,
    ilw.blocking_engine_lock_id blocking_lock,
    il.lock_mode blocking_mode,
    il.lock_type blocking_type,
    ilw.blocking_engine_transaction_id blocking_transaction,
    CASE it.trx_state
        WHEN 'LOCK WAIT'
        THEN it.trx_state
        ELSE p.state end blocker_state,
    concat(il.object_schema,'.', il.object_name) as locked_table,
    it.trx_mysql_thread_id blocker_thread,
    p.user blocker_user,
    p.host blocker_host
FROM performance_schema.data_lock_waits ilw
JOIN performance_schema.data_locks il
ON ilw.blocking_engine_lock_id = il.engine_lock_id
```

```
AND ilw.blocking_engine_transaction_id = il.engine_transaction_id
JOIN information_schema.innodb_trx it
ON ilw.blocking_engine_transaction_id = it.trx_id join information_schema.processlist p
ON it.trx_mysql_thread_id = p.id join information_schema.innodb_trx it1
ON ilw.requesting_engine_transaction_id = it1.trx_id join
 information_schema.processlist p1
ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
waiting_thread: 4244
waiting_user: reinvent
waiting_host: 123.456.789.012:18158
waiting_query: select id1 from test.t1 where id1=1 for update
waiting_transaction: 1688153
blocking_lock: 70369562074216:11:4:2:70369549808672
blocking_mode: X
blocking_type: RECORD
blocking_transaction: 1688142
blocker_state: User sleep
locked_table: test.t1
blocker_thread: 4243
blocker_user: reinvent
blocker_host: 123.456.789.012:18156
1 row in set (0.00 sec)
```

Ao identificar a sessão, suas opções incluem:

- Entre em contato com o proprietário da aplicação ou o usuário.
- Se a sessão de bloqueio estiver ociosa, considere encerrá-la. Essa ação pode desencadear uma reversão longa. Para saber como encerrar uma sessão, consulte Encerrar uma sessão ou consulta.

Para obter mais informações sobre como identificar transações de bloqueio, consulte Uso de informações de transações e bloqueios do InnoDB, na documentação do MySQL.

synch/cond/innodb/row_lock_wait_cond

O evento `synch/cond/innodb/row_lock_wait_cond` ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha. Para obter mais informações, consulte o tópico sobre InnoDB locking na documentação do MySQL.

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 2

Possíveis causas do maior número de esperas

Várias instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas simultaneamente.

Ações

Recomendamos diferentes ações dependendo dos outros eventos de espera exibidos.

Tópicos

- Localizar e responder às instruções SQL responsáveis por esse evento de espera
- Localizar e responder à sessão de bloqueio

Localizar e responder às instruções SQL responsáveis por esse evento de espera

Utilize o Performance Insights para identificar as instruções SQL responsáveis por esse evento de espera. Considere as estratégias a seguir:

- Se bloqueios de linha forem um problema persistente, considere reescrever a aplicação para utilizar o bloqueio otimista.
- Utilize instruções de várias linhas.
- Disperse a workload por objetos de banco de dados diferentes. É possível fazer isso por meio de particionamento.
- Verifique o valor do parâmetro `innodb_lock_wait_timeout`. Ele controla quanto tempo as transações aguardam antes de gerarem um erro de tempo limite.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Localizar e responder à sessão de bloqueio

Determine se a sessão de bloqueio está ociosa ou ativa. Além disso, descubra se a sessão é proveniente de uma aplicação ou de um usuário ativo.

Para identificar a sessão que está mantendo o bloqueio, é possível executar `SHOW ENGINE INNODB STATUS`. O exemplo a seguir mostra uma saída.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 2771110, ACTIVE 112 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 24, OS thread handle 70369573642160, query id 13271336 172.31.14.179
 reinvent Sending data
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 43 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 3 n bits 0 index GEN_CLUST_INDEX of table test.t1 trx
 id 2771110 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

Outra alternativa é utilizar a consulta a seguir para extrair detalhes sobre os bloqueios atuais.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
```

```
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Ao identificar a sessão, suas opções incluem:

- Entre em contato com o proprietário da aplicação ou o usuário.
- Se a sessão de bloqueio estiver ociosa, considere encerrá-la. Essa ação pode desencadear uma reversão longa. Para saber como encerrar uma sessão, consulte Encerrar uma sessão ou consulta.

Para obter mais informações sobre como identificar transações de bloqueio, consulte Uso de informações de transações e bloqueios do InnoDB, na documentação do MySQL.

synch/cond/sql/MDL_context::COND_wait_status

O evento `synch/cond/sql/MDL_context::COND_wait_status` ocorre quando há threads aguardando em um bloqueio de metadados de tabela.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

O evento `synch/cond/sql/MDL_context::COND_wait_status` indica que há threads aguardando um bloqueio de metadados de tabela. Em alguns casos, uma sessão mantém um bloqueio de metadados em uma tabela, e outra sessão tenta obter o mesmo bloqueio na mesma tabela. Nesse caso, a segunda sessão aguarda no evento de espera `synch/cond/sql/MDL_context::COND_wait_status`.

O MySQL utiliza o bloqueio de metadados para gerenciar o acesso simultâneo a objetos de banco de dados e garantir a consistência dos dados. O bloqueio de metadados é aplicável a tabelas, esquemas, eventos agendados, espaços de tabelas e bloqueios de usuários adquiridos com a função `get_lock` e programas armazenados. Programas armazenados incluem procedimentos, funções e acionadores. Para ter mais informações, consulte o tópico sobre Bloqueio de metadados, na documentação do MySQL.

A lista de processos do MySQL mostra esta sessão no estado `waiting for metadata lock`. No Performance Insights, se `Performance_schema` estiver habilitado, o evento `synch/cond/sql/MDL_context::COND_wait_status` será exibido.

O tempo limite padrão para uma consulta aguardando um bloqueio de metadados baseia-se no valor do parâmetro `lock_wait_timeout`, cujo padrão é 31.536.000 segundos (365 dias).

Para obter mais detalhes sobre os diferentes bloqueios do InnoDB e os tipos de bloqueios que podem causar conflitos, consulte o tópico sobre Bloqueio do InnoDB, na documentação do MySQL.

Possíveis causas do maior número de esperas

Quando o evento `synch/cond/sql/MDL_context::COND_wait_status` aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Transações de longa execução

Uma ou mais transações estão modificando muitos dados e mantendo bloqueios nas tabelas por muito tempo.

Transações ociosas

Uma ou mais transações permanecem abertas por muito tempo, sem serem confirmadas ou revertidas.

Instruções DDL em tabelas grandes

Uma ou mais instruções de definição de dados (DDL), como comandos `ALTER TABLE`, foram executadas em tabelas muito grandes.

Bloqueios de tabela explícitos

Existem bloqueios explícitos em tabelas que não estão sendo liberados em tempo hábil. Por exemplo, uma aplicação pode executar instruções `LOCK TABLE` incorretamente.

Ações

Convém tomar medidas diferentes, dependendo das causas do evento de espera e da versão do cluster de banco de dados Aurora MySQL.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Verifique se há eventos anteriores
- Executar consultas no Aurora MySQL versão 2
- Responder à sessão de bloqueio

Identificar as sessões e as consultas que estão causando os eventos

É possível utilizar o Performance Insights para mostrar consultas bloqueadas pelo evento de espera `synch/cond/sql/MDL_context::COND_wait_status`. Porém, para identificar a sessão de

bloqueio, consulte tabelas de metadados de `performance_schema` e `information_schema` no cluster de banco de dados.

Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta e descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para localizar consultas SQL que são responsáveis pela carga alta

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights dessa instância de banco de dados é exibido.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a AWSPublicação no blog de banco de dados sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Verifique se há eventos anteriores

É possível obter insights sobre esse evento de espera para conferir se há ocorrências passadas dele. Para isso, conclua as seguintes ações:

- Verifique a linguagem de manipulação de dados (DML) e a taxa de transferência e latência de DDL para ver se houve alterações na workload.

  É possível utilizar o Performance Insights para encontrar consultas aguardando esse evento na ocasião do problema. Também é possível visualizar o resumo das consultas executadas próximo à ocorrência do problema.

- Se logs de auditoria ou logs gerais estiverem habilitados para o cluster de banco de dados, será possível verificar todas as consultas executadas nos objetos (schema.table) envolvidos na transação em espera. Você também pode verificar consultas com execução concluída antes da transação.

As informações disponíveis para solucionar problemas com eventos anteriores são limitadas. A realização dessas verificações não mostra qual objeto está aguardando informações. No entanto, é possível identificar tabelas com alta carga na ocasião do evento e o conjunto de linhas operadas com frequência que estão causando conflito na ocasião do problema. Em seguida, você pode utilizar essas informações para reproduzir o problema em um ambiente de teste e fornecer insights sobre a sua causa.

Executar consultas no Aurora MySQL versão 2

No Aurora MySQL versão 2, é possível identificar a sessão bloqueada diretamente, consultando tabelas `performance_schema` ou visualizações de esquema `sys`. Um exemplo é capaz de ilustrar como consultar tabelas para identificar consultas e sessões de bloqueio.

Na saída da lista de processos a seguir, o ID da conexão 89 está aguardando um bloqueio de metadados e está executando um comando `TRUNCATE TABLE`. Em uma consulta nas tabelas `performance_schema` ou visualizações de esquema `sys`, a saída mostra que a sessão de bloqueio é 76.

```
MySQL [(none)]> select @@version, @@aurora_version;
+-----------+------------------+
| @@version | @@aurora_version |
+-----------+------------------+
| 5.7.12    | 2.11.5           |
+-----------+------------------+
1 row in set (0.01 sec)

MySQL [(none)]> show processlist;
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+-------------------------------+
| Id | User            | Host               | db        | Command | Time | State
                    | Info                          |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+-------------------------------+
|  2 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                          |
```

```
|  4 | rdsadmin        | localhost          | NULL      | Sleep   |    2 | NULL
                    | NULL                           |
|  5 | rdsadmin        | localhost          | NULL      | Sleep   |    1 | NULL
                    | NULL                           |
| 20 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
| 21 | rdsadmin        | localhost          | NULL      | Sleep   |  261 | NULL
                    | NULL                           |
| 66 | auroramysql5712 | 172.31.21.51:52154 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 67 | auroramysql5712 | 172.31.21.51:52158 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 68 | auroramysql5712 | 172.31.21.51:52150 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 69 | auroramysql5712 | 172.31.21.51:52162 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 70 | auroramysql5712 | 172.31.21.51:52160 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 71 | auroramysql5712 | 172.31.21.51:52152 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 72 | auroramysql5712 | 172.31.21.51:52156 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 73 | auroramysql5712 | 172.31.21.51:52164 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 74 | auroramysql5712 | 172.31.21.51:52166 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 75 | auroramysql5712 | 172.31.21.51:52168 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 76 | auroramysql5712 | 172.31.21.51:52170 | NULL      | Query   |    0 | starting
                    | show processlist               |
| 88 | auroramysql5712 | 172.31.21.51:52194 | NULL      | Query   |   22 | User sleep
                    | select sleep(10000)            |
| 89 | auroramysql5712 | 172.31.21.51:52196 | NULL      | Query   |    5 | Waiting for
 table metadata lock | truncate table sbtest.sbtest1 |
+----+-----------------+--------------------+-----------+---------+------
+--------------------------------+------------------------------+
18 rows in set (0.00 sec)
```

Em seguida, uma consulta nas tabelas `performance_schema` ou visualizações de esquema `sys` mostra que a sessão de bloqueio é 76.

```
MySQL [(none)]> select * from sys.schema_table_lock_waits;
```

```
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
| object_schema | object_name | waiting_thread_id | waiting_pid | waiting_account
           | waiting_lock_type | waiting_lock_duration | waiting_query
   | waiting_query_secs | waiting_query_rows_affected | waiting_query_rows_examined |
 blocking_thread_id | blocking_pid | blocking_account             | blocking_lock_type
 | blocking_lock_duration | sql_kill_blocking_query | sql_kill_blocking_connection |
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
| sbtest        | sbtest1     |               121 |          89 |
 auroramysql5712@192.0.2.0    | EXCLUSIVE          | TRANSACTION           | truncate
 table sbtest.sbtest1 |                10 |                            0 |
               0 |                108 |           76 | auroramysql5712@192.0.2.0    |
 SHARED_READ         | TRANSACTION            | KILL QUERY 76           | KILL 76
              |
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
1 row in set (0.00 sec)
```

Responder à sessão de bloqueio

Ao identificar a sessão, suas opções incluem:

- Entre em contato com o proprietário da aplicação ou o usuário.
- Se a sessão de bloqueio estiver ociosa, considere encerrá-la. Essa ação pode desencadear uma reversão longa. Para saber como encerrar uma sessão, consulte Encerrar uma sessão ou consulta.

Para ter mais informações sobre como identificar transações de bloqueio, consulte o tópico sobre como Utilizar informações de transações e bloqueios do InnoDB, na documentação do MySQL.

synch/mutex/innodb/aurora_lock_thread_slot_futex

O evento `synch/mutex/innodb/aurora_lock_thread_slot_futex` ocorre quando uma sessão bloqueou uma linha para uma atualização e outra linha tenta atualizar a mesma linha. Para ter mais informações, consulte o tópico sobre Bloqueio InnoDB, na Referência do MySQL.

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 2

Possíveis causas do maior número de esperas

Várias instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas simultaneamente.

Ações

Recomendamos diferentes ações dependendo dos outros eventos de espera exibidos.

Tópicos

- Localizar e responder às instruções SQL responsáveis por esse evento de espera
- Localizar e responder à sessão de bloqueio

Localizar e responder às instruções SQL responsáveis por esse evento de espera

Utilize o Performance Insights para identificar as instruções SQL responsáveis por esse evento de espera. Considere as estratégias a seguir:

- Se bloqueios de linha forem um problema persistente, considere reescrever a aplicação para utilizar o bloqueio otimista.
- Utilize instruções de várias linhas.
- Disperse a workload por objetos de banco de dados diferentes. É possível fazer isso por meio de particionamento.
- Verifique o valor do parâmetro `innodb_lock_wait_timeout`. Ele controla quanto tempo as transações aguardam antes de gerarem um erro de tempo limite.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Localizar e responder à sessão de bloqueio

Determine se a sessão de bloqueio está ociosa ou ativa. Além disso, descubra se a sessão é proveniente de uma aplicação ou de um usuário ativo.

Para identificar a sessão que está mantendo o bloqueio, é possível executar `SHOW ENGINE INNODB STATUS`. O exemplo a seguir mostra uma saída.

```
mysql> SHOW ENGINE INNODB STATUS;

----------------------TRANSACTION 302631452, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
MySQL thread id 80109, OS thread handle 0x2ae915060700, query id 938819 10.0.4.12
 reinvent updating
UPDATE sbtest1 SET k=k+1 WHERE id=503
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 148 page no 11 n bits 30 index `PRIMARY` of table
 `sysbench2`.`sbtest1` trx id 302631452 lock_mode X locks rec but not gap waiting
Record lock, heap no 30 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
```

Outra alternativa é utilizar a consulta a seguir para extrair detalhes sobre os bloqueios atuais.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
```

```
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Ao identificar a sessão, suas opções incluem:

- Entre em contato com o proprietário da aplicação ou o usuário.
- Se a sessão de bloqueio estiver ociosa, considere encerrá-la. Essa ação pode desencadear uma reversão longa. Para saber como encerrar uma sessão, consulte Encerrar uma sessão ou consulta.

Para ter mais informações sobre como identificar transações de bloqueio, consulte o tópico sobre como Utilizar informações de transações e bloqueios do InnoDB, no Manual de referência do MySQL.

synch/mutex/innodb/buf_pool_mutex

O evento `synch/mutex/innodb/buf_pool_mutex` ocorre quando um thread adquire um bloqueio no grupo de buffers do InnoDB para acessar uma página na memória.

Tópicos

- Versões de mecanismos relevantes
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões de mecanismos relevantes

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 2

Contexto

O mutex `buf_pool` é um único mutex que protege as estruturas de dados de controle do grupo de buffer.

Para saber mais, consulte o tópico sobre como Monitorar esperar de mutex do InnoDB utilizando o Performance Schema, na documentação do MySQL.

Possíveis causas do maior número de esperas

Este é um evento de espera específico para workload. Causas comuns do surgimento de `synch/mutex/innodb/buf_pool_mutex` entre os principais eventos de espera incluem:

- O tamanho do grupo de buffer não é suficientemente grande para manter o conjunto de dados de trabalho.
- A workload é mais específica para determinadas páginas de uma certa tabela no banco de dados, resultando na disputa no grupo de buffer.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Utilizar o Performance Insights
- Criar réplicas do Aurora
- Examinar o tamanho do grupo de buffer
- Monitorar o histórico do status global

Identificar as sessões e as consultas que estão causando os eventos

Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta e descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para visualizar o gráfico Top SQL (SQL principal) no Console de Gerenciamento da AWS

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Abaixo do gráfico Database load (Carga de banco de dados), escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Utilizar o Performance Insights

Esse evento está relacionado à workload. Você pode utilizar o Performance Insights para fazer o seguinte:

- Identificar quando eventos de espera são iniciados e se há alguma alteração na workload nesse tempo a partir dos logs da aplicação ou origens relacionadas.
- Identificar as instruções SQL responsáveis por esse evento de espera. Examinar o plano de execução das consultas para garantir que elas sejam otimizadas e estejam utilizando índices apropriados.

  Se as principais consultas responsáveis pelo evento de espera estiverem relacionadas ao mesmo objeto ou tabela de banco de dados, considere particionar esse objeto ou tabela.

Criar réplicas do Aurora

É possível criar réplicas do Aurora para fornecer tráfego somente leitura. Também é possível pode utilizar o Aurora Auto Scaling para lidar com surtos no tráfego de leitura. Certifique-se de executar tarefas agendadas somente leitura e backups lógicos nas réplicas do Aurora.

Para ter mais informações, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.

Examinar o tamanho do grupo de buffer

Verifique se o tamanho do grupo de buffer é suficiente para a workload, observando a métrica `innodb_buffer_pool_wait_free`. Se o valor dessa métrica for alto e estiver aumentando continuamente, isso indica que o tamanho do grupo de buffer não é suficiente para lidar com a workload. Se `innodb_buffer_pool_size` tiver sido definido corretamente, o valor de `innodb_buffer_pool_wait_free` deverá ser pequeno. Para obter mais informações, consulte Innodb_buffer_pool_wait_free, na documentação do MySQL.

Aumente o tamanho do grupo de buffer se a instância de banco de dados tiver memória suficiente para buffers de sessão e tarefas do sistema operacional. Se não tiver, altere a instância de banco de dados para uma classe de instância de banco de dados maior a fim de obter memória adicional que possa ser alocada ao grupo de buffer.

> **Note**
>
> O Aurora MySQL ajusta automaticamente o valor de `innodb_buffer_pool_instances` com base no valor de `innodb_buffer_pool_size` configurado.

Monitorar o histórico do status global

Monitorando as taxas de alteração das variáveis de status, é possível detectar problemas de bloqueio ou memória na sua instância de banco de dados. Habilite o histórico de status global (GoSH) se ele ainda não estiver habilitado. Para obter mais informações sobre o GoSH, consulte o tópico sobre como Gerenciar o histórico de status global.

Você também pode criar métricas personalizadas do Amazon CloudWatch para monitorar variáveis de status. Para obter mais informações, consulte o tópico sobre como Publicar métricas personalizadas.

synch/mutex/innodb/fil_system_mutex

O evento `synch/mutex/innodb/fil_system_mutex` ocorre quando uma sessão está aguardando para acessar o cache de memória do espaço de tabelas.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

O InnoDB utiliza espaços de tabela para gerenciar a área de armazenamento de tabelas e arquivos de log. O cache de memória de espaços de tabela é uma estrutura de memória global que

mantém informações sobre espaços de tabela. O MySQL usa esperas `synch/mutex/innodb/fil_system_mutex` para controlar o acesso simultâneo ao cache de memória de espaços de tabela.

O evento `synch/mutex/innodb/fil_system_mutex` indica que atualmente há mais de uma operação que precisa recuperar e manipular informações no cache de memória de espaços de tabela para o mesmo espaço de tabela.

Possíveis causas do maior número de esperas

Quando o evento `synch/mutex/innodb/fil_system_mutex` aparece mais que o normal, possivelmente indicando um problema de performance, isso geralmente ocorre quando todas as seguintes condições estão presentes:

- Aumento nas operações simultâneas da linguagem de manipulação de dados (DML) que atualizam ou excluem dados na mesma tabela.
- O espaço de tabela desta tabela é muito grande e tem muitas páginas de dados.
- O fator para preenchimento dessas páginas de dados é baixo.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Reorganizar tabelas grandes no horário fora de pico

Identificar as sessões e as consultas que estão causando os eventos

Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta e descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para localizar consultas SQL que são responsáveis pela carga alta

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Outra maneira de descobrir quais consultas estão causando muitas esperas `synch/mutex/innodb/fil_system_mutex` é verificar `performance_schema`, como no exemplo a seguir.

```
mysql> select * from performance_schema.events_waits_current where EVENT_NAME='wait/
synch/mutex/innodb/fil_system_mutex'\G
*************************** 1. row ***************************
            THREAD_ID: 19
             EVENT_ID: 195057
         END_EVENT_ID: 195057
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:6700
          TIMER_START: 1010146190118400
            TIMER_END: 1010146196524000
           TIMER_WAIT: 6405600
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 2. row ***************************
```

```
            THREAD_ID: 23
             EVENT_ID: 5480
         END_EVENT_ID: 5480
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:5906
          TIMER_START: 995269979908800
            TIMER_END: 995269980159200
           TIMER_WAIT: 250400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 3. row ***************************
            THREAD_ID: 55
             EVENT_ID: 23233794
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:449
          TIMER_START: 1010492125341600
            TIMER_END: 1010494304900000
           TIMER_WAIT: 2179558400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: 23233786
   NESTING_EVENT_TYPE: WAIT
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Reorganizar tabelas grandes no horário fora de pico

Reorganize tabelas grandes que você identifica como a fonte de números elevados de eventos de espera `synch/mutex/innodb/fil_system_mutex` durante uma janela de manutenção fora do horário de produção. Isso garante que a limpeza do mapa de espaços de tabela internos não ocorra quando o acesso rápido à tabela for essencial. Para obter informações sobre como reorganizar tabelas, consulte a Instrução OPTIMIZE TABLE, na Referência do MySQL.

synch/mutex/innodb/trx_sys_mutex

O evento `synch/mutex/innodb/trx_sys_mutex` ocorre quando há alta atividade de banco de dados com muitas transações.

Tópicos

- Versões de mecanismos relevantes
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões de mecanismos relevantes

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versões 2 e 3

Contexto

Internamente, o mecanismo de banco de dados InnoDB utiliza o nível de isolamento de leitura repetível com snapshots para fornecer consistência de leitura. Isso fornece uma visão pontual do banco de dados na ocasião em que o snapshot foi criado.

No InnoDB, todas as alterações são aplicadas ao banco de dados logo que chegam, independentemente de terem sido confirmadas ou não. Essa abordagem significa que, sem o controle de simultaneidade de várias versões (MVCC), todos os usuários conectados ao banco de dados visualizam todas as alterações e as linhas mais recentes. Portanto, o InnoDB requer uma maneira de acompanhar as alterações para saber o que deve ser revertido quando necessário.

Para fazer isso, o InnoDB utiliza um sistema de transações (`trx_sys`) para acompanhar snapshots. Esse sistema de transações faz o seguinte:

- Acompanha o ID da transação para cada linha nos logs de operações desfazer.
- Usa uma estrutura interna do InnoDB denominada `ReadView`, que ajuda a identificar quais IDs de transação estão visíveis para um snapshot.

Possíveis causas do maior número de esperas

Qualquer operação de banco de dados que exija manuseio consistente e controlado (criação, leitura, atualização e exclusão) de IDs de transações gera uma chamada de `trx_sys` para o mutex.

Essas chamadas acontecem dentro de três funções:

- `trx_sys_mutex_enter` – Cria o mutex.
- `trx_sys_mutex_exit` – Libera o mutex.
- `trx_sys_mutex_own` – Testa se existe uma propriedade para o mutex.

A instrumentação do Performance Schema do InnoDB rastreia todas as chamadas de mutex `trx_sys`. O acompanhamento inclui, mas não se limita a, o gerenciamento de `trx_sys` na inicialização ou encerramento do banco de dados, operações de reversão, limpezas de operações desfazer, acesso de leitura de linha e cargas de grupo de buffer. A alta atividade do banco de dados com um grande número de transações resulta no surgimento de `synch/mutex/innodb/trx_sys_mutex` entre os principais eventos de espera.

Para saber mais, consulte o tópico sobre como Monitorar esperar de mutex do InnoDB utilizando o Performance Schema, na documentação do MySQL.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar as sessões e as consultas que estão causando os eventos
- Examinar outros eventos de espera

Identificar as sessões e as consultas que estão causando os eventos

Em geral, bancos de dados com carga de moderada a significativa apresentam eventos de espera. Os eventos de espera podem ser aceitáveis quando a performance é ideal. Se a performance não for

ideal, examine onde o banco de dados está passando a maior parte do tempo. Observe os eventos de espera que contribuem para a carga mais alta. Descubra se é possível otimizar o banco de dados e a aplicação para reduzir esses eventos.

Para visualizar o gráfico Top SQL (SQL principal) no Console de gerenciamento da AWS

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Abaixo do gráfico Database load (Carga de banco de dados), escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a Publicação no blog sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Examinar outros eventos de espera

Examine os outros eventos de espera associados ao evento de espera `synch/mutex/innodb/trx_sys_mutex`. Isso pode fornecer mais informações sobre a natureza da workload. Um número elevado de transações pode reduzir a taxa de transferência, mas a workload também pode tornar isso necessário.

Para obter mais informações sobre como otimizar transações, consulte o tópico sobre como Otimizar o gerenciamento de transações do InnoDB, na documentação do MySQL.

synch/sxlock/innodb/hash_table_locks

O evento `synch/sxlock/innodb/hash_table_locks` ocorre quando as páginas não encontradas no grupo de buffer devem ser lidas do armazenamento.

Tópicos

- Versões compatíveis do mecanismo

- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões:

- Aurora MySQL versões 2 e 3

Contexto

O evento `synch/sxlock/innodb/hash_table_locks` indica que uma workload está acessando com frequência dados que não estão armazenados no grupo de buffer. Esse evento de espera está associado a novas adições de página e remoções de dados antigos do grupo de buffer. Os dados armazenados no grupo de buffer antigos e os dados novos devem ser armazenados em cache, para que as páginas antigas sejam despejadas de forma a permitir o armazenamento em cache das novas páginas. O MySQL aplica um algoritmo menos utilizado recentemente (LRU) para expulsar páginas do grupo de buffers. A workload está tentando acessar dados que não foram carregados no grupo de buffer ou dados que foram despejados do grupo de buffer.

Esse evento de espera ocorre quando a workload deve acessar os dados em arquivos no disco ou quando os blocos são liberados ou adicionados à lista de LRUs do grupo de buffer. Essas operações esperam até obter um bloqueio excluído compartilhado (SX-lock). Esse SX-lock é utilizado para a sincronização na tabela de hash, uma tabela na memória projetada para melhorar a performance do acesso ao grupo de buffer.

Para obter mais informações, consulte o tópico sobre o Grupo de buffer, na documentação do MySQL.

Possíveis causas do maior número de esperas

Quando o evento de espera `synch/sxlock/innodb/hash_table_locks` aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Um grupo de buffers subdimensionado

O tamanho do grupo de buffer é muito pequeno para manter na memória todas as páginas frequentemente acessadas.

Workload pesada

A workload está causando despejos frequentes e recarregamentos de páginas de dados no cache de buffer.

Erros de leitura das páginas

Ocorrem erros ao ler páginas no grupo de buffer, o que pode indicar corrupção dos dados.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Aumentar o tamanho do grupo de buffer
- Melhorar padrões de acesso a dados
- Reduzir ou evitar varreduras de tabelas inteiras
- Verificar se há páginas corrompidas nos logs de erros

Aumentar o tamanho do grupo de buffer

Certifique-se de que o grupo de buffer esteja apropriadamente escalado para a workload. Para isso, é possível verificar a taxa de acertos do cache do grupo de buffer. Em geral, se o valor cair abaixo de 95%, considere aumentar o tamanho do grupo de buffer. Um grupo de buffer maior pode manter por mais tempo na memória as páginas acessadas com frequência. Para aumentar o tamanho do grupo de buffer, modifique o valor do parâmetro `innodb_buffer_pool_size`. O valor padrão desse parâmetro baseia-se no tamanho da classe da instância de banco de dados. Para obter mais informações, consulte Práticas recomendadas para a configuração do banco de dados do Amazon Aurora MySQL.

Melhorar padrões de acesso a dados

Confira as consultas afetadas por essa espera e seus planos de execução. Considere melhorar padrões de acesso a dados. Por exemplo, se você estiver utilizando mysqli_result::fetch_array, poderá tentar aumentar o tamanho da busca das matrizes.

É possível utilizar o Performance Insights para mostrar consultas e sessões que possam estar causando o evento de espera `synch/sxlock/innodb/hash_table_locks`.

Para localizar consultas SQL que são responsáveis pela carga alta

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância de banco de dados.
4. No gráfico Database load (Carga de banco de dados), escolha Slice by wait (Segmentar por espera).
5. Na parte inferior da página, escolha Top SQL (SQL principal).

   O gráfico mostra as consultas SQL que são responsáveis pela carga. Os que estão no topo da lista são os mais responsáveis. Para solucionar um gargalo, concentre-se nessas instruções.

Para obter uma visão geral útil da solução de problemas de uso do Performance Insights, consulte a AWSPublicação no blog de banco de dados sobre como Analisar workloads do Amazon Aurora MySQL com o Performance Insights.

Reduzir ou evitar varreduras de tabelas inteiras

Monitore sua workload para ver se ela está executando varreduras de tabelas inteiras e, se estiver, reduza ou evite essas varreduras. Por exemplo, é possível monitorar variáveis de status como `Handler_read_rnd_next`. Para obter mais informações, consulte o tópico sobre Variáveis de status do servidor, na documentação do MySQL.

Verificar se há páginas corrompidas nos logs de erros

É possível consultar mysql-error.log em busca de mensagens relacionadas a corrupções que foram detectadas perto do momento do problema. As mensagens com as quais é possível trabalhar para solucionar o problema estão no log de erros. Talvez seja necessário recriar objetos que foram sinalizados como corrompidos.

synch/mutex/innodb/temp_pool_manager_mutex

O evento de espera `synch/mutex/innodb/temp_pool_manager_mutex` ocorre quando uma sessão está aguardando para adquirir um mutex para gerenciar o grupo de espaços para tabela temporários de sessão.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte nas seguintes versões do mecanismo:

- Aurora MySQL versão 3

Contexto

O Aurora MySQL versão 3.x e posterior usa `temp_pool_manager_mutex` para controlar várias sessões acessando o grupo temporário de espaços para tabela ao mesmo tempo. O Aurora MySQL gerencia o armazenamento por meio de um volume de cluster do Aurora para dados persistentes e armazenamento local para arquivos temporários. Um espaço para tabela temporário é necessário quando uma sessão cria uma tabela temporária no volume de cluster do Aurora.

Quando uma sessão solicita pela primeira vez um espaço para tabela temporário, o MySQL aloca espaços para tabela temporários da sessão do grupo compartilhado. Uma sessão pode conter até dois espaços para tabela temporários por vez para os seguintes tipos de tabela:

- Tabelas temporárias criadas pelo usuário.
- Tabelas temporárias internas geradas pelo otimizador

O mecanismo `TempTable` padrão usa o seguinte mecanismo de transbordamento para lidar com tabelas temporárias:

- Armazena tabelas na RAM até o limite `temptable_max_ram`.
- Muda para arquivos mapeados na memória no armazenamento local quando a RAM está cheia.
- Usa o volume do cluster compartilhado quando os arquivos mapeados na memória atingem o limite `temptable_max_mmap`.

Depois que as tabelas temporárias excedem os limites da RAM e do armazenamento local, o MySQL as gerencia usando o espaço para tabela em disco.

Quando uma sessão exige uma tabela temporária em disco, o MySQL:

- Procura espaços para tabela `INACTIVE` disponíveis no grupo para reutilização.
- Criará dez espaços para tabela se não existirem espaços `INACTIVE`.

Quando uma sessão é desconectada, o MySQL:

- Trunca os espaços para tabela temporários da sessão.
- Marca-os como INACTIVE no grupo para reutilização.
- Mantém o tamanho atual do grupo até a reinicialização do servidor.
- Volta ao tamanho padrão do grupo (dez espaços para tabela) após a reinicialização.

Possíveis causas do maior número de esperas

Situações comuns que causam esse evento de espera:

- Sessões simultâneas que criam tabelas temporárias internas no volume do cluster.
- Sessões simultâneas que criam tabelas temporárias no volume do cluster.
- Encerramento repentino de sessões que usam espaços para tabela ativos.
- Expansão do grupo de espaços para tabela durante workloads com uso intenso de gravação.
- Consultas simultâneas que acessam `INFORMATION_SCHEMA.`

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Monitorar e otimizar o uso de tabelas temporárias
- Analisar as consultas usando INFORMATION_SCHEMA
- Aumentar o parâmetro innodb_sync_array_size
- Implementar um pool de conexões

Monitorar e otimizar o uso de tabelas temporárias

Para monitorar e otimizar o uso de tabelas temporárias, use um destes métodos:

- Confira o contador `Created_tmp_disk_tables` no Insights de Performance para monitorar a criação de tabelas temporárias em disco no cluster do Aurora.
- Execute este comando em seu banco de dados para monitorar diretamente a criação de tabelas temporárias: `mysql> show status like '%created_tmp_disk%'`.

> **Note**
>
> O comportamento da tabela temporária difere entre os nós de leitura e gravação do Aurora MySQL. Para obter mais informações, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3.

Depois de identificar as consultas que criam tabelas temporárias, siga estas etapas de otimização:

- Use `EXPLAIN` para examinar os planos de execução de consultas e identificar onde e por que as tabelas temporárias estão sendo criadas.
- Modifique as consultas para reduzir o uso de tabelas temporárias sempre que possível.

Se a otimização de consultas por si só não resolver os problemas de performance, pense em ajustar estes parâmetros de configuração:

- `temptable_max_ram`: controla o uso máximo de RAM para tabelas temporárias.
- `temptable_max_mmap`: define o limite para armazenamento de arquivos mapeados na memória.
- `tmp_table_size`: aplica-se quando `aurora_tmptable_enable_per_table_limit` está habilitado (desabilitado por padrão).

> **Important**
>
> Observe que determinadas condições de consulta sempre exigirão tabelas temporárias em disco, independentemente das configurações. Para acessar mais informações sobre o mecanismo de armazenamento `TempTable`, consulte Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL.

Analisar as consultas usando INFORMATION_SCHEMA

Quando você consulta tabelas `INFORMATION_SCHEMA`, o MySQL cria tabelas temporárias do InnoDB no volume do cluster. Cada sessão precisa de um espaço para tabela temporário para essas tabelas, o que pode causar problemas de performance durante o alto acesso simultâneo.

Para melhorar a performance:

- Use `PERFORMANCE_SCHEMA` em vez de `INFORMATION_SCHEMA` sempre que possível.
- Se você precisar usar `INFORMATION_SCHEMA`, reduza a frequência com que você executa essas consultas.

Aumentar o parâmetro innodb_sync_array_size

O parâmetro `innodb_sync_array_size` controla o tamanho da matriz de espera mutex/lock no MySQL. O valor padrão de 1 funciona para workloads gerais, mas aumentá-lo pode reduzir a contenção de threads durante a alta simultaneidade.

Quando sua workload mostra um número crescente de threads de espera:

- Monitore o número de threads de espera em sua workload.
- Defina `innodb_sync_array_size` igual ou superior à contagem de vCPUs da sua instância para dividir a estrutura de coordenação de threads e reduzir a contenção.

> **Note**
>
> Para determinar o número de vCPUs disponíveis em sua instância do RDS, consulte as especificações da vCPU em Tipos de instância do Amazon RDS.

Implementar um pool de conexões

O MySQL atribui um espaço para tabela dedicado a cada sessão que cria uma tabela temporária. Esse espaço para tabela permanece ativo até que a conexão com o banco de dados termine. Para gerenciar seus recursos com maior eficiência:

- Implemente o agrupamento de conexões para limitar o número de espaços para tabelas temporários ativos.

- Reutilize as conexões existentes em vez de criar outras para cada operação.

Ajustar o Aurora MySQL com estados de threads

A tabela a seguir resume os estados de threads gerais mais comuns do Aurora MySQL.

Estado geral do thread	Descrição
???	Esse estado de thread indica que um thread está processando uma instrução `SELECT` que requer o uso de uma tabela temporária interna para classificar os dados.
???	Esse estado de thread indica que um thread está lendo e filtrando linhas de uma consulta para determinar o conjunto de resultados correto.

criar índice de classificação

O estado de thread `creating sort index` indica que um thread está processando uma instrução `SELECT` que requer o uso de uma tabela temporária interna para classificar os dados.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações sobre estados de thread têm suporte para as seguintes versões:

- Aurora MySQL versão 2 até 2.09.2

Contexto

O estado `creating sort index` aparece quando uma consulta com uma cláusula `ORDER BY` ou `GROUP BY` não pode utilizar um índice existente para realizar a operação. Nesse caso, o MySQL

precisa realizar uma operação `filesort` mais cara. Em geral, essa operação é executada na memória quando o conjunto de resultados não é muito grande. Caso contrário, ela envolve a criação de um arquivo no disco.

Possíveis causas do maior número de esperas

O surgimento de `creating sort index` por si só não indica um problema. Se a performance for ruim e você vir instâncias frequentes de `creating sort index`, a causa mais provável são consultas lentas com operadores `ORDER BY` ou `GROUP BY`.

Ações

A diretriz geral é encontrar consultas com cláusulas `ORDER BY` ou `GROUP BY` associadas aos aumentos no estado `creating sort index`. Em seguida, verifique se adicionar um índice ou aumentar o tamanho do buffer de classificação resolve o problema.

Tópicos

- Habilite o Performance Schema se ele não estiver habilitado
- Identificar as consultas com problemas
- Examinar os planos de explicação para o uso da classificação de arquivo
- Aumentar o tamanho do buffer de classificação

Habilite o Performance Schema se ele não estiver habilitado

O Performance Insights relata estados de thread somente quando instrumentos do Performance Schema não estão habilitados. Quando os instrumentos do Performance Schema estão habilitados, o Performance Insights relata eventos de espera. Os instrumentos do Performance Schema fornecem insights adicionais e melhores ferramentas quando você investiga possíveis problemas de performance. Portanto, convém habilitar o Performance Schema. Para obter mais informações, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Identificar as consultas com problemas

Para identificar consultas atuais que estão causando aumentos no estado `creating sort index`, execute `show processlist` e veja se alguma das consultas tem `ORDER BY` ou `GROUP BY`. Opcionalmente, execute `explain for connection N`, em que `N` é o ID de lista de processos da consulta com `filesort`.

Para identificar consultas anteriores que estão causando esses aumentos, ative o log de consultas lentas e localize as consultas com `ORDER BY`. Execute `EXPLAIN` nas consultas lentas e procure "usando classificação de arquivo". Para obter mais informações, consulte Examinar os planos de explicação para o uso da classificação de arquivo.

Examinar os planos de explicação para o uso da classificação de arquivo

Identifique as instruções com cláusulas `ORDER BY` ou `GROUP BY` que resultam no estado `creating sort index`.

O exemplo a seguir mostra como executar `explain` em uma consulta. A coluna `Extra` mostra que essa consulta usa `filesort`.

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2064548
     filtered: 100.00
        Extra: Using filesort
1 row in set, 1 warning (0.01 sec)
```

O exemplo a seguir mostra o resultado da execução de `EXPLAIN` na mesma consulta depois que um índice é criado na coluna `c1`.

```
mysql> alter table mytable add index (c1);
```

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: index
possible_keys: NULL
```

```
          key: c1
      key_len: 1023
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using index
1 row in set, 1 warning (0.01 sec)
```

Para obter informações sobre o uso de índices para otimização da ordem de classificação, consulte Otimização ORDER BY, na documentação do MySQL.

Aumentar o tamanho do buffer de classificação

Para ver se uma consulta específica exigia um processo `filesort` que criou um arquivo no disco, verifique o valor da variável `sort_merge_passes` após a execução da consulta. Por exemplo:

```
mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)

--- run query
mysql> select * from mytable order by u limit 10;
--- run status again:

mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)
```

Se o valor de `sort_merge_passes` for alto, considere aumentar o tamanho do buffer de classificação. Aplique o aumento no nível da sessão, porque elevá-lo globalmente pode aumentar significativamente a quantidade de RAM utilizada pelo MySQL. O exemplo a seguir mostra como alterar o tamanho do buffer de classificação antes de executar uma consulta.

```
mysql> set session sort_buffer_size=10*1024*1024;
```

```
Query OK, 0 rows affected (0.00 sec)
-- run query
```

enviar dados

O estado de thread `sending data` indica que um thread está lendo e filtrando linhas de uma consulta para determinar o conjunto de resultados correto. O nome é enganoso porque implica que o estado está transferindo dados, e não coletando e preparando dados para envio posterior.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações sobre estados de thread têm suporte para as seguintes versões:

- Aurora MySQL versão 2 até 2.09.2

Contexto

Muitos estados de thread são de curta duração. Operações ocorrendo durante `sending data` tendem a realizar um grande número de leituras em disco ou cache. Portanto, `sending data` geralmente é o estado de execução mais longa durante a vida útil de uma determinada consulta. Esse estado aparece quando o Aurora MySQL está fazendo o seguinte:

- Lendo e processando linhas para uma instrução `SELECT`
- Executando um grande número de leituras do disco ou da memória
- Concluindo uma leitura completa de todos os dados de uma consulta específica
- Lendo dados de uma tabela, um índice ou o trabalho de um procedimento armazenado
- Classificando, agrupando ou ordenando dados

Depois que o estado `sending data` terminar de preparar os dados, o estado de thread `writing to net` indicará o retorno dos dados para o cliente. Normalmente, `writing to net` é capturado

somente quando o conjunto de resultados é muito grande ou a latência de rede grave está retardando a transferência.

Possíveis causas do maior número de esperas

O surgimento de `sending data` por si só não indica um problema. Se a performance for ruim e você vir instâncias frequentes de `sending data`, as causas mais prováveis são as seguintes.

Tópicos

- Consulta ineficiente
- Configuração do servidor abaixo do ideal

Consulta ineficiente

Na maioria dos casos, o que é responsável por esse estado é uma consulta que não está utilizando um índice apropriado para localizar o conjunto de resultados de uma determinada consulta. Por exemplo, considere uma consulta que está lendo uma tabela de 10 milhões de registros para todos os pedidos feitos na Califórnia e na qual coluna de estado não está indexada ou está mal indexada. No último caso, o índice pode existir, mas o otimizador o ignora devido à baixa cardinalidade.

Configuração do servidor abaixo do ideal

Se várias consultas aparecerem no estado `sending data`, o servidor de banco de dados pode estar indevidamente configurado. Especificamente, o servidor pode apresentar os seguintes problemas:

- O servidor de banco de dados não possui capacidade de computação suficiente: E/S de disco, tipo e velocidade de disco, CPU ou número de CPUs.
- O servidor está sem recursos alocados, como o grupo de buffers do InnoDB para tabelas do InnoDB ou o buffer de chaves para tabelas MyIsam.
- Configurações de memória por thread, como `sort_buffer`, `read_buffer` e `join_buffer` consomem mais RAM do que o necessário, deixando o servidor físico sem recursos de memória.

Ações

A diretriz geral é localizar consultas que retornem muitas linhas, verificando o Performance Schema. Se o registro em log de consultas que não usam índices estiver habilitado, você também poderá examinar os resultados dos logs lentos.

Tópicos

- Habilite o Performance Schema se ele não estiver habilitado
- Examinar configurações de memória
- Examinar os planos de explicação para o uso de índices
- Verificar o volume dos dados retornados
- Verificar se há problemas de simultaneidade
- Verificar a estrutura das suas consultas

Habilite o Performance Schema se ele não estiver habilitado

O Performance Insights relata estados de thread somente quando instrumentos do Performance Schema não estão habilitados. Quando os instrumentos do Performance Schema estão habilitados, o Performance Insights relata eventos de espera. Os instrumentos do Performance Schema fornecem insights adicionais e melhores ferramentas quando você investiga possíveis problemas de performance. Portanto, convém habilitar o Performance Schema. Para obter mais informações, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Examinar configurações de memória

Examine as configurações de memória para os grupos de buffer primários. Certifique-se de que esses grupos sejam dimensionados adequadamente para a workload. Se o banco de dados utilizar várias instâncias de grupos de pool, certifique-se de que elas não estejam divididas em muitos grupos de buffer pequenos. Threads só podem utilizar um grupo de buffer de cada vez.

Certifique-se de que as seguintes configurações de memória utilizadas para cada thread sejam dimensionadas corretamente:

- read_buffer
- read_rnd_buffer
- sort_buffer
- join_buffer
- binlog_cache

A menos que você tenha um motivo específico para modificar as configurações, use os valores padrão.

Examinar os planos de explicação para o uso de índices

Para consultas no estado de thread `sending data`, examine o plano para determinar se os índices apropriados são utilizados. Se uma consulta não estiver utilizando um índice útil, considere adicionar dicas como `USE INDEX` ou `FORCE INDEX`. Dicas podem aumentar ou diminuir muito o tempo necessário para executar uma consulta. Portanto, tenha cautela antes de adicioná-las.

Verificar o volume dos dados retornados

Verifique as tabelas que estão sendo consultadas e a quantidade de dados que elas contêm. Alguns desses dados podem ser arquivados? Em muitos casos, a causa de tempos de execução de consultas ruins não é o resultado do plano de consulta, mas sim o volume de dados a serem processados. Muitos desenvolvedores são eficientes ao adicionar dados a um banco de dados, mas raramente consideram o ciclo de vida do conjunto de dados nas fases de design e de desenvolvimento.

Procure consultas que tenham boa performance em bancos de dados de baixo volume, mas que tenham performance ruim no seu sistema atual. Às vezes, os desenvolvedores que projetam consultas específicas podem não perceber que essas consultas estão retornando 350.000 linhas. Eles podem ter desenvolvido essas consultas em um ambiente de menor volume com conjuntos de dados menores do que os dos ambientes de produção.

Verificar se há problemas de simultaneidade

Verifique se várias consultas do mesmo tipo estão sendo executadas ao mesmo tempo. Algumas formas de consultas são executadas de maneira eficiente quando são executadas sozinhas. No entanto, se formas semelhantes de consultas forem executadas juntas ou em alto volume, elas poderão causar problemas de simultaneidade. Frequentemente, esses problemas são causados quando o banco de dados utiliza tabelas temporárias para renderizar resultados. Um nível restritivo de isolamento de transações também pode causar problemas de simultaneidade.

Se tabelas forem lidas e gravadas simultaneamente, o banco de dados pode estar utilizando bloqueios. Para ajudar a identificar períodos de baixa performance, examine o uso de bancos de dados por meio de processos em lote em larga escala. Para ver bloqueios e reversões recentes, examine a saída do comando `SHOW ENGINE INNODB STATUS`.

Verificar a estrutura das suas consultas

Verifique se as consultas capturadas desses estados usam subconsultas. Esse tipo de consulta costuma resultar em uma performance ruim, pois o banco de dados compila os resultados

internamente e os substitui de volta na consulta para renderizar os dados. Esse processo é uma etapa adicional para o banco de dados. Em muitos casos, essa etapa pode resultar em uma performance ruim em uma condição de carregamento altamente simultânea.

Verifique também se as suas consultas usam um grande número de cláusulas `ORDER BY` e `GROUP BY`. Nessas operações, muitas vezes o banco de dados deve primeiro formar todo o conjunto de dados na memória. Em seguida, deve ordená-lo ou agrupá-lo de uma maneira específica antes de o retornar ao cliente.

Ajustar o Aurora MySQL com insights proativos do Amazon DevOps Guru

Os insights proativos do DevOps Guru detectam condições problemáticas conhecidas em clusters de banco de dados do Aurora MySQL antes que elas ocorram. O DevOps Guru pode fazer o seguinte:

- Evitar muitos problemas comuns de bancos de dados ao comparar a configuração do seu banco de dados com as configurações comuns recomendadas.
- Alertar sobre problemas críticos em sua frota que, se não forem controlados, poderão causar problemas maiores no futuro.
- Alertar sobre problemas recém-descobertos.

Cada insight proativo contém uma análise da causa do problema e recomendações de ações corretivas.

Tópicos

- O tamanho da lista de histórico do InnoDB aumentou significativamente
- O banco de dados está criando tabelas temporárias no disco

O tamanho da lista de histórico do InnoDB aumentou significativamente

A lista de histórico de alterações de linha aumentou significativamente desde *`data`* até *`tamanho`* em *`db-instance`*. Esse aumento afeta a performance de consulta e do desligamento do banco de dados.

Tópicos

- Versões compatíveis do mecanismo
- Contexto

- Causas prováveis desse problema
- Ações
- Métricas relevantes

Versões compatíveis do mecanismo

Essas informações de insights são compatíveis com todas as versões do Aurora MySQL.

Contexto

O sistema de transações do InnoDB mantém o controle de simultaneidade de várias versões (MVCC). Quando uma linha é modificada, a versão pré-modificação dos dados que estão sendo modificados é armazenada como um registro de anulação em um log de anulação. Cada registro de anulação tem uma referência ao seu registro de redefinição anterior, formando uma lista vinculada.

A lista de histórico do InnoDB é uma lista global dos registros de anulação de transações confirmadas. O MySQL usa a lista de histórico para remover os registros e as páginas de log quando as transações não precisarem mais do histórico. O tamanho da lista de histórico é o número total de registros de anulação que contêm modificações na lista do histórico. Cada log contém uma ou mais modificações. Se a lista de histórico do InnoDB ficar muito grande, indicando um grande número de versões de linhas antigas, as consultas e os desligamentos de bancos de dados ficarão mais lentos.

Causas prováveis desse problema

As causas comuns para uma longa lista de histórico incluem o seguinte:

- Transações de longa duração, seja de leitura ou gravação
- Uma carga de gravação pesada

Ações

Recomendamos ações distintas dependendo dos motivos do insight.

Tópicos

- Não inicie nenhuma operação que envolve um desligamento de banco de dados até que a lista de histórico do InnoDB diminua
- Identificar e encerrar transações de longa duração

- Identificar os principais hosts e usuários usando o recurso Insights de Performance

Não inicie nenhuma operação que envolve um desligamento de banco de dados até que a lista de histórico do InnoDB diminua

Como uma lista de histórico do InnoDB longa retarda desligamentos de banco de dados, reduza o tamanho da lista antes de iniciar operações que envolvam o desligamento de um banco de dados. Essas operações incluem upgrades da versão primária de bancos de dados.

Identificar e encerrar transações de longa duração

Você pode encontrar transações de longa duração consultando `information_schema.innodb_trx`.

> **Note**
>
> Certifique-se também de procurar transações de longa duração em réplicas de leitura.

Como identificar e encerrar transações de longa duração

1. No cliente SQL, execute a seguinte consulta:

```
SELECT a.trx_id,
      a.trx_state,
      a.trx_started,
      TIMESTAMPDIFF(SECOND,a.trx_started, now()) as "Seconds Transaction Has Been
 Open",
      a.trx_rows_modified,
      b.USER,
      b.host,
      b.db,
      b.command,
      b.time,
      b.state
FROM  information_schema.innodb_trx a,
      information_schema.processlist b
WHERE a.trx_mysql_thread_id=b.id
  AND TIMESTAMPDIFF(SECOND,a.trx_started, now()) > 10
ORDER BY trx_started
```

2. Encerre cada transação de longa duração com o procedimento armazenado mysql.rds_kill.

Identificar os principais hosts e usuários usando o recurso Insights de Performance

Otimize as transações para que grandes quantidades de linhas modificadas sejam imediatamente confirmadas.

Métricas relevantes

As métricas a seguir estão relacionadas a esse insight:

- `trx_rseg_history_len`: essa métrica do contador pode ser visualizada no Insights de Performance, bem como na tabela `INFORMATION_SCHEMA.INNODB_METRICS`. Para obter mais informações, consulte a Tabela de métricas INFORMATION_SCHEMA do InnoDB na documentação do MySQL.
- `RollbackSegmentHistoryListLength`: essa métrica do Amazon CloudWatch mede os logs de ações desfeitas que registram transações confirmadas com registros marcados para exclusão. Esses registros estão programados para serem processados pela operação de limpeza do InnoDB. A métrica `trx_rseg_history_len` tem o mesmo valor que `RollbackSegmentHistoryListLength`.
- `PurgeBoundary`: o número da transação até o qual a limpeza do InnoDB é permitida. Se essa métrica do CloudWatch não avançar por longos períodos, é uma boa indicação de que a limpeza do InnoDB está bloqueada por transações de longa duração. Para investigar, confira as transações ativas no cluster de banco de dados do Aurora MySQL. Essa métrica está disponível apenas para o Aurora MySQL versão 2.11 e posteriores e versões 3.08 e posteriores.
- `PurgeFinishedPoint`: o número da transação até o qual a limpeza do InnoDB é realizada. Essa métrica do CloudWatch pode ajudar a examinar a rapidez com que a limpeza do InnoDB está progredindo. Essa métrica está disponível apenas para o Aurora MySQL versão 2.11 e posteriores e versões 3.08 e posteriores.
- `TransactionAgeMaximum`: a idade da transação em execução ativa mais antiga. Essa métrica do CloudWatch está disponível para o Aurora MySQL versão 3.08 e posteriores.
- `TruncateFinishedPoint`: o número da transação até o qual o truncamento é desfeito. Essa métrica do CloudWatch está disponível apenas para o Aurora MySQL versão 2.11 e posteriores e versões 3.08 e posteriores.

Para obter mais informações sobre as métricas do CloudWatch, consulte Métricas no nível da instância do Amazon Aurora.

O banco de dados está criando tabelas temporárias no disco

Seu uso recente de tabelas temporárias em disco aumentou significativamente, até *porcentagem*. O banco de dados está criando cerca de *número* tabelas temporárias por segundo. Isso pode afetar a performance e aumentar as operações no disco em *instância de banco de dados*.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Causas prováveis desse problema
- Ações
- Métricas relevantes

Versões compatíveis do mecanismo

Essas informações de insights são compatíveis com todas as versões do Aurora MySQL.

Contexto

Às vezes, o servidor MySQL precisa criar uma tabela temporária interna ao processar uma consulta. O Aurora MySQL pode conter uma tabela temporária interna na memória, onde ela pode ser processada pelo mecanismo de armazenamento TempTable ou MEMORY, ou armazenada em disco pelo InnoDB. Para obter mais informações, consulte Uso de tabela temporária interna no MySQL no Manual de referência do MySQL.

Causas prováveis desse problema

Um aumento nas tabelas temporárias em disco indica o uso de consultas complexas. Se a memória configurada for insuficiente para armazenar tabelas temporárias na memória, o Aurora MySQL criará as tabelas no disco. Isso pode afetar a performance e aumentar as operações no disco.

Ações

Recomendamos ações distintas dependendo dos motivos do insight.

- Para o Aurora MySQL versão 3, recomendamos que você use o mecanismo de armazenamento TempTable.
- Otimize suas consultas para retornar menos dados selecionando somente as colunas necessárias.

Se você ativar o esquema de performance com todos os instrumentos `statement` habilitados e cronometrados, poderá consultar `SYS.statements_with_temp_tables` para recuperar a lista de consultas que usam tabelas temporárias. Para obter mais informações, consulte Pré-requisitos para usar o esquema sys na documentação do MySQL.

- Considere indexar colunas envolvidas em operações de classificação e agrupamento.
- Reescreva suas consultas para evitar as colunas `BLOB` e `TEXT`. Essas colunas sempre usam o disco.
- Ajuste os seguintes parâmetros de banco de dados: `tmp_table_size` e `max_heap_table_size`.

  O valor padrão para esses parâmetros é 16 MiB. Ao usar o mecanismo de armazenamento MEMORY para tabelas temporárias na memória, o tamanho máximo delas é definido pelo valor `tmp_table_size` ou `max_heap_table_size`, o que for menor. Quando esse tamanho máximo é atingido, o MySQL converte automaticamente a tabela temporária interna na memória em uma tabela temporária interna em disco do InnoDB. Para obter mais informações, consulte Usar o mecanismo de armazenamento TempTable no Amazon RDS para MySQL e Amazon Aurora MySQL.

  > **Note**
  >
  > Ao criar explicitamente tabelas MEMORY com CREATE TABLE, somente a variável `max_heap_table_size` determina o tamanho máximo até o qual uma tabela pode crescer. Também não há conversão para um formato em disco.

Métricas relevantes

As seguintes métricas do recurso Insights de Performance estão relacionadas a esse insight:

- Created_tmp_disk_tables
- Created_tmp_tables

Para obter mais informações, consulte Created_tmp_disk_tables na documentação do MySQL.

Consulta paralela do Amazon Aurora MySQL

Este tópico descreve a performance de consultas paralelas do Amazon Aurora edição compatível com MySQL. Este recurso usa um caminho de processamento especial para determinadas consultas com muitos dados, aproveitando a arquitetura de armazenamento compartilhada do Aurora. As consultas paralelas funcionam melhor com clusters de bancos de dados Aurora MySQL que têm tabelas com milhões de linhas e consultas analíticas que levam minutos ou horas para concluir.

Tópicos

- Visão geral de consultas paralelas do Aurora MySQL
- Criar um cluster de banco de dados de consulta paralela no Aurora MySQL
- Ativar e desativar a consulta paralela no Aurora MySQL
- Otimizar a consulta paralela no Aurora MySQL
- Verificar quais declarações usam consulta paralela para Aurora MySQL
- Monitorar a consulta paralela do Aurora MySQL
- Constructos do SQL para consulta paralela no Aurora MySQL

Visão geral de consultas paralelas do Aurora MySQL

A consulta paralela do Aurora MySQL é uma otimização que paraleliza uma parte da E/S e da computação envolvidas nas consultas de processamento com muitos dados. O trabalho que é paralelizado inclui linhas de recuperação de armazenamento, extração de valores de colunas e determinação de quais linhas correspondem às condições na cláusula `WHERE` e em cláusulas de junção. O trabalho com muitos dados é delegado (em termos de otimização de banco de dados, empurrado) para vários nós na camada de armazenamento distribuído do Aurora. Sem as consultas paralelas, cada consulta traz todos os dados de varredura para um nó único dentro do cluster do Aurora MySQL (o nó de cabeçalho) e realiza todo o processamento de consultas lá.

> **Tip**
>
> O mecanismo de banco de dados PostgreSQL também tem um recurso denominado “consulta paralela”. Esse recurso não está relacionado à consulta paralela do Aurora.

Quando o recurso de consulta paralela está habilitado, o mecanismo do Aurora MySQL determina automaticamente quando as consultas podem se beneficiar, sem exigir alterações de SQL como

avisos ou atributos de tabela. Nas seções a seguir, você encontrará uma explicação sobre quando uma consulta paralela é aplicada a uma consulta. Você também descobrirá como garantir que a consulta paralela seja aplicada onde ela pode oferecer o maior benefício.

> **Note**
>
> A otimização de consultas paralelas oferece o maior benefício para consultas de longa duração que levam minutos ou horas para serem concluídas. Geralmente, o Aurora MySQL não executa a otimização de consulta paralela para consultas pouco dispendiosas. Normalmente, ele também não executa otimização de consultas paralelas se outra técnica de otimização faz mais sentido, como o armazenamento em cache de consultas, armazenamento em cache de grupos de buffer ou pesquisas de índice. Se o recurso de consulta paralela não estiver sendo usado quando você espera, consulte Verificar quais declarações usam consulta paralela para Aurora MySQL.

Tópicos

- Benefícios
- Arquitetura
- Pré-requisitos
- Limitações
- Os custos de E/S com a consulta paralela

Benefícios

Com a consulta paralela, você pode executar consultas analíticas com uso intensivo de dados em tabelas do Aurora MySQL. Em muitos casos, você pode obter uma melhoria de performance de ordem de grandeza em relação à divisão tradicional de trabalho para processamento de consultas.

Os benefícios da consulta paralela incluem o seguinte:

- Melhor performance de E/S devido ao paralelismo de solicitações de leitura físicas entre vários nós de armazenamento.
- Menor tráfego de rede. O Aurora não transmite páginas inteiras de dados dos nós de armazenamento até o nó de cabeçalho para depois filtrar as linhas e colunas desnecessárias. Em vez disso, o Aurora transmite tuplas compactas que contêm somente os valores das colunas necessários para o conjunto de resultados.

- Uso reduzido de CPU no nó de cabeçalho, pois o processamento da função, a filtragem de linhas e a projeção de colunas para a cláusula `WHERE` são delegadas (ou “empurradas”).
- Pressão reduzida da memória no grupo de buffers. As páginas processadas pela consulta paralela não são adicionadas ao grupo de buffer. Essa abordagem reduz a chance de uma verificação intensiva de dados despejar dados usados com frequência do grupo de buffer.
- Possível redução da duplicação de dados em seu pipeline de extração, transformação e carga (ETL), ao tornar mais prática a realização de consultas analíticas de longa execução em dados existentes.

Arquitetura

O recurso de consulta paralela usa os princípios de arquitetura mais importantes do Aurora MySQL: desacoplamento do mecanismo de banco de dados do subsistema de armazenamento e redução do tráfego de rede com a simplificação dos protocolos de comunicação. O Aurora MySQL usa essas técnicas para acelerar operações de gravação intensiva, como processamento de log de refazimento. A consulta paralela aplica os mesmos princípios nas operações de leitura.

> **Note**
>
> A arquitetura da consulta paralela do Aurora MySQL é diferente da arquitetura de outros recursos semelhantes em outros sistemas de banco de dados. A consulta paralela do Aurora MySQL não envolve o multiprocessamento simétrico (SMP) e, portanto, não depende da capacidade da CPU do servidor de banco de dados. O processamento paralelo ocorre na camada de armazenamento, de forma independente do servidor do Aurora MySQL que funciona como o coordenador de consultas.

Por padrão, sem a consulta paralela, o processamento de uma consulta do Aurora envolve a transmissão de dados brutos para um nó único dentro do cluster do Aurora (o nó do cabeçalho). Depois, o Aurora executa todo o processamento adicional para essa consulta em um único thread desse nó único. Com a consulta paralela, a maior parte do trabalho intensivo de E/S e da CPU é delegado aos nós na camada de armazenamento. Somente as linhas compactas do conjunto de resultados são transmitidas de volta ao nó de cabeçalho, com as linhas já filtradas e os valores das colunas já extraídos e transformados. O benefício com relação à performance vem com a redução no tráfego da rede, a redução do uso de CPU no nó de cabeçalho e o paralelismo da E/S entre os nós de armazenamento. A quantidade de E/S, filtragem e projeção que são paralelizados independe do número de instâncias de banco de dados no cluster do Aurora que executa a consulta.

Pré-requisitos

O uso de todos os recursos da consulta paralela exige um cluster de bancos de dados do Aurora MySQL que esteja executando a versão 2.09 ou posterior. Se você já tiver um cluster que deseja utilizar com consultas paralelas, será possível atualizá-lo para uma versão compatível e habilitar consultas paralelas posteriormente. Nesse caso, siga o procedimento de atualização em Considerações sobre atualização para consultas paralelas porque os nomes de definição da configuração e os valores padrão são diferentes nessas versões mais recentes.

As instâncias de banco de dados no cluster devem usar as classes de instância `db.r*`.

A otimização da junção de hash deve estar ativada para o cluster. Para saber como, consulte Habilitar a junção de hash para clusters de consulta paralela.

Para personalizar parâmetros como `aurora_parallel_query` e `aurora_disable_hash_join`, é necessário ter um grupo de parâmetros personalizado usado com o cluster. É possível especificar esses parâmetros individualmente para cada instância de banco de dados usando um grupo de parâmetros de banco de dados. No entanto, recomendamos que você os especifique em um grupo de parâmetros do cluster de banco de dados. Dessa forma, todas as instâncias de banco de dados no cluster herdam as mesmas configurações para esses parâmetros.

Limitações

Os seguintes limitações se aplicam ao recurso de consulta paralela:

- A consulta paralela não é compatível com a configuração de armazenamento do cluster de banco de dados do Aurora I/O-Optimized.
- Não é possível usar a consulta paralela com as classes de instância db.t2 ou db.t3. Essa limitação se aplica mesmo se você solicitar a consulta paralela usando a variável de sessão `aurora_pq_force`.
- A consulta paralela não se aplica a tabelas que usam os formatos de linha `COMPRESSED` ou `REDUNDANT`. Use os formatos de linha `COMPACT` ou `DYNAMIC` para tabelas que você planeja usar com a consulta paralela.
- O Aurora usa um algoritmo baseado em custo para determinar se deve usar o mecanismo de consulta paralela para cada instrução SQL. Usar certas construções de SQL em uma instrução pode impedir a consulta paralela ou torná-la improvável para essa instrução. Para obter informações sobre a compatibilidade de construções de SQL com a consulta paralela, consulte Constructos do SQL para consulta paralela no Aurora MySQL.

- Cada instância de bancos de dados Aurora só pode executar um determinado número de sessões de consultas paralelas de cada vez. Se uma consulta tiver várias partes que usam a consulta paralela, como subconsultas, junções ou operadores `UNION`, essas fases serão executadas em sequência. A instrução só conta com uma sessão de consulta paralela única de cada vez. Você pode monitorar o número de sessões ativas usando as variáveis de status de consulta paralela. Você pode verificar o limite de sessões simultâneas para uma determinada instância de banco de dados consultando a variável de status `Aurora_pq_max_concurrent_requests`.
- A consulta paralela está disponível em todas as Regiões da AWS compatíveis com o Aurora. Para a maioria das Regiões da AWS, a versão mínima necessária do Aurora MySQL para usar a consulta paralela é a 2.09.
- A consulta paralela foi projetada para melhorar a performance de consultas com uso intenso de dados. Ela não foi projetada para consultas leves.
- Recomendamos que você use nós de leitor para instruções SELECT, especialmente para as que fazem uso intenso de dados.

Os custos de E/S com a consulta paralela

Se seu cluster do Aurora MySQL usar consulta paralela, você poderá ver um aumento nos valores de `VolumeReadIOPS`. As consultas paralelas não usam o grupo de buffers. Assim, embora as consultas sejam rápidas, esse processamento otimizado pode resultar em aumento nas operações de leitura e nas cobranças associadas.

Os custos de E/S de consulta paralela para uma consulta são medidos na camada de armazenamento e serão iguais ou maiores com a consulta paralela ativada. Seu benefício é a melhoria na performance da consulta. Os custos de E/S de consultas paralelas podem aumentar por dois motivos:

- Mesmo que alguns dos dados em uma tabela estejam no grupo de buffers, a consulta paralela exige que todos os dados sejam escaneados na camada de armazenamento, gerando custos de E/S.
- A execução de uma consulta paralela não aquece o grupo de buffers. Por isso, execuções consecutivas da mesma consulta paralela contribuem para o custo total de E/S.

Criar um cluster de banco de dados de consulta paralela no Aurora MySQL

Para criar um cluster do Aurora MySQL com consulta paralela, adicionar novas instâncias a ele ou executar outras operações administrativas, use as mesmas técnicas do Console de gerenciamento da AWS e da AWS CLI usadas com outros clusters do Aurora MySQL. Você pode criar um cluster novo para trabalhar com a consulta paralela. Também é possível criar um cluster de banco de dados para trabalhar com a consulta paralela fazendo a restauração de um snapshot de um cluster de bancos de dados Aurora compatível com o MySQL. Se você não tem familiaridade com o processo de criação de um cluster novo do Aurora MySQL, encontrará as informações básicas e os pré-requisitos em Criar um cluster de bancos de dados do Amazon Aurora.

Ao escolher uma versão do mecanismo Aurora MySQL, recomendamos que escolha a versão mais recente disponível. No momento, todas as versões disponíveis do Aurora MySQL oferecem suporte a consultas paralelas. Se você utilizar as versões mais recentes, terá maior flexibilidade para ativar e desativar a consulta paralela ou usá-la em clusters existentes.

Ao criar um cluster novo ou restaurar um snapshot, use as mesmas técnicas que você usa com outros clusters do Aurora MySQL para adicionar novas instâncias de banco de dados.

É possível criar um cluster de consulta paralela usando o console do Amazon RDS ou a AWS CLI.

Sumário

- Criar um cluster de consulta paralela com o console
- Criar um cluster de consulta paralela com a CLI

Criar um cluster de consulta paralela com o console

Você pode criar um novo cluster de consulta paralela com o console, conforme descrito a seguir.

Para criar um cluster de consulta paralela com o Console de gerenciamento da AWS

1. Siga o procedimento geral do Console de gerenciamento da AWS em Criar um cluster de bancos de dados do Amazon Aurora.
2. Em Tipo de mecanismo, escolha Aurora MySQL.
3. Em Configuração adicional, escolha um grupo de parâmetros criado para o Grupo de parâmetros do cluster de banco de dados. O uso de um grupo de parâmetros personalizado é exigido para o Aurora MySQL 2.09 ou posteriores. No grupo de parâmetros do cluster de banco de dados, especifique as configurações de parâmetro `aurora_parallel_query=ON`

e `aurora_disable_hash_join=OFF`. Isso habilita a consulta paralela para o cluster e possibilita a otimização de junções de hash que funciona em combinação com a consulta paralela.

Para verificar se um cluster novo pode usar a consulta paralela

1. Crie um cluster usando a técnica anterior.
2. (Para o Aurora MySQL versões 2 ou 3) Verifique se a definição da configuração `aurora_parallel_query` é true.

```
mysql> select @@aurora_parallel_query;
+-------------------------+
| @@aurora_parallel_query |
+-------------------------+
|                       1 |
+-------------------------+
```

3. (Para o Aurora MySQL versão 2) Verifique se a configuração `aurora_disable_hash_join` está definia como false.

```
mysql> select @@aurora_disable_hash_join;
+----------------------------+
| @@aurora_disable_hash_join |
+----------------------------+
|                          0 |
+----------------------------+
```

4. Com algumas tabelas grandes e consultas com uso intenso de dados, confirme nos planos de consulta se algumas de suas consultas estão usando a otimização de consulta paralela. Para fazer isso, siga o procedimento em Verificar quais declarações usam consulta paralela para Aurora MySQL.

Criar um cluster de consulta paralela com a CLI

Você pode criar um novo cluster de consulta paralela com a CLI, conforme descrito a seguir.

Para criar um cluster de consulta paralela com o AWS CLI

1. (Opcional) Verifique quais versões do Aurora MySQL são compatíveis com clusters de consulta paralela. Para fazer isso, use o comando `describe-db-engine-versions` e verifique o valor

do campo `SupportsParallelQuery`. Para ver um exemplo, consulte Verificar se a versão do Aurora MySQL é compatível com a consulta paralela.

2. (Opcional) Crie um grupo de parâmetros do cluster de banco de dados personalizado com as configurações `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF`. Use comandos como os seguintes.

```
aws rds create-db-cluster-parameter-group --db-parameter-group-family aurora-
mysql8.0 --db-cluster-parameter-group-name pq-enabled-80-compatible
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_disable_hash_join,ParameterValue=OFF,ApplyMethod=pending-
reboot
```

   Se você executar essa etapa, especifique a opção `--db-cluster-parameter-group-name` *`my_cluster_parameter_group`* na instrução `create-db-cluster` subsequente. Substitua o nome do seu próprio grupo de parâmetros. Se você omitir essa etapa, crie o grupo de parâmetros e associe-o ao cluster posteriormente, conforme descrito em Ativar e desativar a consulta paralela no Aurora MySQL.

3. Siga o procedimento geral do AWS CLI em Criar um cluster de bancos de dados do Amazon Aurora.

4. Especifique o seguinte conjunto de opções:

   - Para a opção `--engine`, use `aurora-mysql`. Esses valores produzem clusters de consulta paralela compatíveis com o MySQL 5.7 ou 8.0.
   - Para a opção `--db-cluster-parameter-group-name`, especifique o nome de um grupo de parâmetros do cluster de banco de dados que você criou e para o qual especificou o valor do parâmetro `aurora_parallel_query=ON`. Se você omitir essa opção, poderá criar o cluster com um grupo de parâmetros padrão e modificá-lo posteriormente para usar esse grupo de parâmetros personalizado.
   - Para a opção `--engine-version`, use uma versão do Aurora MySQL compatível com a consulta paralela. Use o procedimento de Otimizar a consulta paralela no Aurora MySQL para obter uma lista de versões, se necessário.

O exemplo de código a seguir mostra como fazer isso. Substitua seu próprio valor para cada uma das variáveis de ambiente, como *$CLUSTER_ID*. Este exemplo também especifica a opção `--manage-master-user-password` para gerar a senha mestra do usuário e gerenciá-la no Secrets Manager. Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

```
aws rds create-db-cluster --db-cluster-identifier $CLUSTER_ID \
  --engine aurora-mysql --engine-version 8.0.mysql_aurora.3.04.1 \
  --master-username $MASTER_USER_ID --manage-master-user-password \
  --db-cluster-parameter-group-name $CUSTOM_CLUSTER_PARAM_GROUP

aws rds create-db-instance --db-instance-identifier ${INSTANCE_ID}-1 \
  --engine same_value_as_in_create_cluster_command \
  --db-cluster-identifier $CLUSTER_ID --db-instance-class $INSTANCE_CLASS
```

5. Verifique se um cluster que você criou ou restaurou tem o recurso de consulta paralela disponível.

   Verifique se a configuração `aurora_parallel_query` existe. Se essa configuração tiver o valor 1, a consulta paralela estará pronta para ser usada. Se essa configuração tiver o valor 0, defina-a como 1 para poder usar a consulta paralela. De qualquer forma, o cluster é capaz de executar consultas paralelas.

   ```
   mysql> select @@aurora_parallel_query;
   +------------------------+
   | @@aurora_parallel_query|
   +------------------------+
   |                      1 |
   +------------------------+
   ```

Para restaurar um snapshot em um cluster de consulta paralela com o AWS CLI

1. Verifique quais versões do Aurora MySQL são compatíveis com clusters de consulta paralela. Para fazer isso, use o comando `describe-db-engine-versions` e verifique o valor do campo `SupportsParallelQuery`. Para ver um exemplo, consulte Verificar se a versão do Aurora MySQL é compatível com a consulta paralela. Decida qual versão usar para o cluster restaurado.

2. Encontre um snapshot de cluster compatível com o Aurora MySQL.
3. Siga o procedimento geral do AWS CLI em Restauração de um snapshot de um cluster de banco de dados.

```
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier mynewdbcluster \
  --snapshot-identifier mydbclustersnapshot \
  --engine aurora-mysql
```

4. Verifique se um cluster que você criou ou restaurou tem o recurso de consulta paralela disponível. Use o mesmo procedimento de verificação descrito em Criar um cluster de consulta paralela com a CLI.

Ativar e desativar a consulta paralela no Aurora MySQL

Quando a consulta paralela está habilitada, o Aurora MySQL determina se deve utilizá-la em tempo de execução para cada consulta. No caso de junções, uniões, subconsultas etc., o Aurora MySQL determinará se usará a consulta paralela em tempo de execução para cada bloco de consulta. Para obter mais detalhes, consulte Verificar quais declarações usam consulta paralela para Aurora MySQL e Constructos do SQL para consulta paralela no Aurora MySQL.

É possível habilitar e desabilitar a consulta paralela dinamicamente no nível global e da sessão para uma instância de banco de dados utilizando a opção aurora_parallel_query. Você pode alterar a configuração `aurora_parallel_query` no grupo de clusters de banco de dados para ativar ou desativar a consulta paralela por padrão.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Para ativar ou desativar o parâmetro `aurora_parallel_query` no nível da sessão, use os métodos padrão para alterar a definição da configuração do cliente. Por exemplo, você pode fazer isso pela linha de comando `mysql` ou em uma aplicação JDBC ou ODBC. O comando no cliente MySQL padrão é `set session aurora_parallel_query = {'ON'/'OFF'}`. Você também pode adicionar o parâmetro de nível de sessão à configuração JDBC ou dentro do código da sua aplicação para habilitar ou desabilitar a consulta paralela dinamicamente.

É possível alterar permanentemente a configuração do parâmetro `aurora_parallel_query`, seja para uma instância de banco de dados específica ou para todo o cluster. Se você especificar o valor do parâmetro em um grupo de parâmetros de banco de dados, esse valor só será aplicado a uma instância de banco de dados específica no cluster. Se você especificar o valor do parâmetro em um grupo de parâmetros do cluster de banco de dados, todas as instâncias de banco de dados no cluster herdarão a mesma configuração. Para ativar ou desativar o parâmetro `aurora_parallel_query`, utilize as técnicas usadas com grupos de parâmetros, conforme descrito em Grupos de parâmetros para Amazon Aurora. Siga estas etapas:

1. Crie um grupo de parâmetros de cluster personalizado (recomendado) ou um grupo de parâmetros de banco de dados personalizado.
2. Nesse grupo de parâmetros, atualize `parallel_query` para o valor desejado.
3. Dependendo se você criou um grupo de parâmetros do cluster de banco de dados ou um grupo de parâmetros de banco de dados, anexe o grupo de parâmetros ao cluster do Aurora ou às instâncias de banco de dados específicas nas quais você planeja usar o recurso de consulta paralela.

> **Tip**
>
> Como `aurora_parallel_query` é um parâmetro dinâmico, não é necessário reiniciar o cluster depois de alterar essa configuração. No entanto, todas as conexões que estiverem usando consulta paralela antes de alternar a opção continuarão a fazê-lo até que a conexão seja fechada ou a instância seja reinicializada.

Você pode modificar o parâmetro da consulta paralela usando a operação da API ModifyDBClusterParameterGroup ou ModifyDBParameterGroup no Console de gerenciamento da AWS.

É possível ativar a junção de hash para clusters de consulta paralela, ativar e desativar a consulta paralela usando o console do Amazon RDS ou a AWS CLI e substituir o otimizador de consulta paralela.

Sumário

- Habilitar a junção de hash para clusters de consulta paralela
- Habilitar e desabilitar a consulta paralela com o uso do console
- Habilitar e desabilitar a consulta paralela com o uso da CLI

- Substituir o otimizador de consulta paralela

Habilitar a junção de hash para clusters de consulta paralela

A consulta paralela é normalmente usada para tipos de consultas que consomem muitos recursos e que se beneficiam da otimização de junções de hash. Portanto, é útil garantir que as junções de hash estejam habilitadas para os clusters nos quais você pretende utilizar a consulta paralela. Para obter informações sobre como habilitar junções de hash e usá-las de maneira eficaz, consulte Otimizando grandes consultas de junção do Aurora MySQL com junções hash.

Habilitar e desabilitar a consulta paralela com o uso do console

É possível habilitar ou desabilitar a consulta paralela no nível da instância de banco de dados ou do cluster de banco de dados trabalhando com grupos de parâmetros.

Para habilitar ou desabilitar a consulta paralela para um cluster de banco de dados com o Console de gerenciamento da AWS

1. Crie um parameter group personalizado conforme descrito em Grupos de parâmetros para Amazon Aurora.
2. Atualize aurora_parallel_query para 1 (ativada) ou 0 (desativada). Em clusters nos quais o recurso de consulta paralela está disponível, aurora_parallel_query está desabilitado por padrão.
3. Se você usar um grupo de parâmetros de cluster personalizado, anexe-o ao cluster de bancos de dados Aurora onde planeja usar o recurso de consulta paralela. Se você usar um grupo de parâmetros de banco de dados personalizado, anexe-o a uma ou mais instâncias de banco de dados no cluster. Recomendamos o uso de um grupo de parâmetros de cluster. Isso garante que todas as instâncias de banco de dados no cluster tenham as mesmas configurações para consulta paralela e recursos associados, como a junção de hash.

Habilitar e desabilitar a consulta paralela com o uso da CLI

Você pode modificar o parâmetro de consulta paralela usando o comando `modify-db-cluster-parameter-group` ou `modify-db-parameter-group`. Escolha o comando apropriado dependendo se você especificar o valor de `aurora_parallel_query` por meio de um grupo de parâmetros de cluster de banco de dados ou de um grupo de parâmetros de banco de dados.

Para habilitar ou desabilitar a consulta paralela para um cluster de banco de dados com a CLI

- Modifique o parâmetro de consulta paralela usando o comando `modify-db-cluster-parameter-group`. Use um comando como o seguinte. Substitua o nome apropriado para o seu próprio grupo de parâmetros personalizado. Substitua `ON` ou `OFF` da parte `ParameterValue` da opção `--parameters`.

```
$ aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "cluster_param_group_name"
}

aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters ParameterName=aurora_pq,ParameterValue=ON,ApplyMethod=pending-reboot
```

Você também pode habilitar ou desabilitar a consulta no nível de sessão, por exemplo, por meio da linha de comando `mysql` ou dentro de uma aplicação JDBC ou ODBC. Para fazer isso, use os métodos padrão para alterar uma configuração de cliente. Por exemplo, o comando no cliente MySQL padrão é `set session aurora_parallel_query = {'ON'/'OFF'}` para o Aurora MySQL.

Você também pode adicionar o parâmetro de nível de sessão à configuração JDBC ou dentro do código da sua aplicação para habilitar ou desabilitar a consulta paralela dinamicamente.

Substituir o otimizador de consulta paralela

Você pode usar a variável de sessão `aurora_pq_force` para substituir o otimizador de consulta paralela e solicitar uma consulta paralela para cada consulta. Recomendamos que você faça isso apenas para fins de teste. O exemplo a seguir mostra como usar `aurora_pq_force` em uma sessão.

```
set SESSION aurora_parallel_query = ON;
set SESSION aurora_pq_force = ON;
```

Para desativar a substituição, faça o seguinte:

```
set SESSION aurora_pq_force = OFF;
```

Otimizar a consulta paralela no Aurora MySQL

Para otimizar o cluster de banco de dados para consulta paralela, pense em quais clusters de banco de dados se beneficiariam da consulta paralela e se você deveria fazer a atualização para a consulta paralela. Depois, ajuste a workload e crie objetos de esquema para consulta paralela.

Sumário

- Planejar um cluster de consulta paralela
  - Verificar se a versão do Aurora MySQL é compatível com a consulta paralela
- Considerações sobre atualização para consultas paralelas
  - Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3
  - Fazer upgrade para o Aurora MySQL 2.09 e versões posteriores
- Ajuste de performance da consulta paralela
- Criar objetos de esquema que aproveitem a consulta paralela

Planejar um cluster de consulta paralela

A ação de planejar um cluster de banco de dados com consulta paralela habilitada requer algumas escolhas. Isso inclui a realização de etapas de configuração (criar ou restaurar um cluster do Aurora MySQL completo) e a decisão da extensão da habilitação da consulta paralela no cluster de banco de dados.

Considere o seguinte como parte do planejamento:

- Se você usar o Aurora MySQL que é compatível com o MySQL 5.7, deverá criar um cluster provisionado. Em seguida, habilite a consulta paralela utilizando o parâmetro `aurora_parallel_query`.

  Se você tiver um cluster do Aurora MySQL, não precisará criar outro cluster para usar a consulta paralela. É possível associar o cluster, ou instâncias de banco de dados específicas no cluster, a um grupo de parâmetros que tenha o parâmetro `aurora_parallel_query` habilitado. Ao fazer isso, é possível reduzir o tempo e o esforço para configurar os dados relevantes a serem usados com a consulta paralela.

- Planeje todas as tabelas grandes que você precisa reorganizar para que possa usar a consulta paralela ao acessá-las. Talvez seja necessário criar versões de algumas tabelas grandes nas quais a consulta paralela é útil. Por exemplo, talvez seja necessário remover índices de pesquisa de texto completo. Para obter mais detalhes, consulte Criar objetos de esquema que aproveitem a consulta paralela.

Verificar se a versão do Aurora MySQL é compatível com a consulta paralela

Para verificar quais versões do Aurora MySQL são compatíveis com clusters de consulta paralela, use o comando `describe-db-engine-versions` da AWS CLI e verifique o valor do campo `SupportsParallelQuery`. O exemplo de código a seguir mostra como verificar quais combinações estão disponíveis para clusters de consulta paralela em uma região da AWS especificada. Especifique a string completa do parâmetro `--query` em uma única linha.

```
aws rds describe-db-engine-versions --region us-east-1 --engine aurora-mysql \
--query '*[]|[?SupportsParallelQuery == `true`].[EngineVersion]' --output text
```

Os comandos anteriores produzem uma saída semelhante à seguinte. A saída pode variar, dependendo de quais versões do Aurora MySQL estão disponíveis na região da AWS especificada.

```
5.7.mysql_aurora.2.11.1
5.7.mysql_aurora.2.11.2
5.7.mysql_aurora.2.11.3
5.7.mysql_aurora.2.11.4
5.7.mysql_aurora.2.11.5
5.7.mysql_aurora.2.11.6
5.7.mysql_aurora.2.12.0
5.7.mysql_aurora.2.12.1
5.7.mysql_aurora.2.12.2
5.7.mysql_aurora.2.12.3
5.7.mysql_aurora.2.12.4
8.0.mysql_aurora.3.04.0
8.0.mysql_aurora.3.04.1
8.0.mysql_aurora.3.04.2
8.0.mysql_aurora.3.04.3
8.0.mysql_aurora.3.05.2
8.0.mysql_aurora.3.06.0
8.0.mysql_aurora.3.06.1
8.0.mysql_aurora.3.07.0
8.0.mysql_aurora.3.07.1
```

Após começar a usar a consulta paralela com um cluster, você pode monitorar a performance e remover obstáculos ao uso da consulta paralela. Para obter essas instruções, consulte Ajuste de performance da consulta paralela.

Considerações sobre atualização para consultas paralelas

Dependendo das versões original e de destino ao fazer o upgrade de um cluster de consulta paralela, você pode perceber melhorias nos tipos de consultas que a consulta paralela é capaz de otimizar. Você também pode descobrir que não é necessário especificar um parâmetro de modo de mecanismo especial para a consulta paralela. As seções a seguir explicam as considerações ao fazer upgrade de um cluster com a consulta paralela habilitada.

Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3

Várias instruções, cláusulas e tipos de dados SQL incluem suporte novo ou avançado para consulta paralela começando com o Aurora MySQL versão 3. Ao fazer upgrade de uma versão anterior à versão 3, verifique se consultas adicionais podem se beneficiar com otimizações de consultas paralelas. Para obter informações sobre essas melhorias de consulta paralela, consulte Tipos de dados de coluna,Tabelas particionadas e Funções agregadas, cláusulas GROUP BY e cláusulas HAVING..

Se estiver fazendo upgrade de um cluster de consulta paralela do Aurora MySQL 2.08 ou versão inferior, informe-se também sobre as mudanças em como habilitar a consulta paralela. Para isso, leia Fazer upgrade para o Aurora MySQL 2.09 e versões posteriores.

No Aurora MySQL versão 3, a otimização de junções de hash está habilitada por padrão. A opção de configuração `aurora_disable_hash_join` de versões anteriores não é utilizada.

Fazer upgrade para o Aurora MySQL 2.09 e versões posteriores

No Aurora MySQL versão 2.09 e posteriores, a consulta paralela funciona para clusters provisionados e não requer o parâmetro `parallelquery` do modo de mecanismo. Assim, você não precisa criar um cluster nem restaurar de um snapshot existente para usar a consulta paralela com essas versões. Você pode usar os procedimentos de atualização descritos em Atualizando a versão secundária ou o nível de patch de um cluster de banco de dados de Aurora MySQL a fim de atualizar o cluster para essa versão. Você pode atualizar um cluster mais antigo, independentemente de se tratar de um cluster de consulta paralela ou de um cluster provisionado. Para reduzir o número de opções no menu Engine version (Versão do mecanismo) você pode selecionar Show versions that support the parallel query feature (Mostrar versões que oferecem suporte ao recurso de consulta paralela) para filtrar as entradas nesse menu. Depois, escolha o Aurora MySQL 2.09 ou posteriores.

Depois de fazer upgrade de um cluster de consulta paralela anterior para o Aurora MySQL 2.09 ou versão posterior, habilite a consulta paralela no cluster atualizado. A consulta paralela é desativada por padrão nessas versões e o procedimento para habilitá-la é diferente. A otimização de junções de hash também está desabilitada por padrão e deve ser habilitada separadamente. Por isso, certifique-se de habilitar essas configurações novamente após o upgrade. Para obter instruções sobre como fazer isso, consulte Ativar e desativar a consulta paralela no Aurora MySQL e Habilitar a junção de hash para clusters de consulta paralela.

Em particular, você habilita consulta paralela utilizando os parâmetros de configuração `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF` em vez de `aurora_pq_supported` e `aurora_pq`. Os parâmetros `aurora_pq_supported` e `aurora_pq` é compatível com atualizações nas versões mais recentes do Aurora MySQL.

No cluster atualizado, o atributo `EngineMode` tem o valor `provisioned` em vez de `parallelquery`. Para verificar se a consulta paralela está disponível para uma versão de mecanismo especificada, agora você confere o valor do campo `SupportsParallelQuery` na saída do comando `describe-db-engine-versions` da AWS CLI. Em versões anteriores do Aurora MySQL, você verificou a presença de `parallelquery` na lista `SupportedEngineModes`.

Depois de fazer upgrade para o Aurora MySQL 2.09 ou versão posterior, você poderá aproveitar os recursos a seguir. Esses recursos não estão disponíveis para clusters de consulta paralela que executam versões mais antigas do Aurora MySQL.

- Performance Insights. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.
- Retroceder. Para ter mais informações, consulte Retroceder um cluster de banco de dados Aurora.
- Interromper e iniciar o cluster. Para ter mais informações, consulte Interromper e iniciar um cluster de banco de dados do Amazon Aurora.

Ajuste de performance da consulta paralela

Para gerenciar a performance de uma workload com consulta paralela, verifique se a consulta paralela é usada em consultas em que a otimização é mais útil.

Para isso, você pode fazer o seguinte:

- As tabelas maiores devem ser compatíveis com a consulta paralela. Você pode alterar as propriedades da tabela ou recriar algumas tabelas para que as consultas delas possam aproveitar

a otimização da consulta paralela. Para saber como, consulte Criar objetos de esquema que aproveitem a consulta paralela.

- Monitore quais são as consultas que usam a consulta paralela. Para saber como, consulte Monitorar a consulta paralela do Aurora MySQL.
- Verifique se a consulta paralela está sendo usada para as consultas mais intensas e de longa execução e com o nível certo de simultaneidade para a workload. Para saber como, consulte Verificar quais declarações usam consulta paralela para Aurora MySQL.
- Adapte seu código SQL para permitir que a consulta paralela seja aplicada às consultas esperadas. Para saber como, consulte Constructos do SQL para consulta paralela no Aurora MySQL.

Criar objetos de esquema que aproveitem a consulta paralela

Antes de criar ou modificar tabelas que você planeja usar para consulta paralela, conheça os requisitos descritos em Pré-requisitos e Limitações.

Como a consulta paralela exige que as tabelas usem a configuração `ROW_FORMAT=Compact` ou `ROW_FORMAT=Dynamic`, verifique as definições de configurações do Aurora para saber se houve alguma alteração na opção de configuração `INNODB_FILE_FORMAT`. Execute a instrução `SHOW TABLE STATUS` para confirmar o formato da linha de todas as tabelas do banco de dados.

Antes de alterar seu esquema para permitir que a consulta paralela funcione com mais tabelas, não deixe de fazer testes. Os testes devem confirmar se a consulta paralela resulta em um aumento líquido na performance dessas tabelas. Além disso, certifique-se de que os requisitos do esquema para a consulta paralela são compatíveis com as suas metas.

Por exemplo, antes de mudar de `ROW_FORMAT=Compressed` para `ROW_FORMAT=Compact` ou `ROW_FORMAT=Dynamic`, teste a performance das cargas de trabalho das tabelas originais e novas. Considere também outros efeitos possíveis, como o aumento do volume de dados.

Verificar quais declarações usam consulta paralela para Aurora MySQL

Em operações normais, você não precisa executar nenhuma ação especial para tirar vantagem da consulta paralela. Quando uma consulta atende aos requisitos essenciais de uma consulta paralela, o otimizador de consultas decide automaticamente se a consulta paralela deve ser usada para cada consulta específica.

Se você realizar experimentos em um ambiente de desenvolvimento ou teste, poderá descobrir que a consulta paralela não está sendo usada porque as tabelas são muito pequenas em termos de número de linhas ou de volume de dados geral. Os dados da tabela podem estar inteiramente em um grupo de buffers, especialmente nas tabelas que você criou recentemente para realizar os experimentos.

Ao monitorar ou ajustar a performance do cluster, decida se a consulta paralela está sendo usada nos contextos apropriados. Você pode ajustar o esquema de banco de dados, as configurações, as consultas SQL ou mesmo a topologia do cluster e as configurações de conexão do aplicativo para aproveitar esse recurso.

Para confirmar se uma consulta está usando a consulta paralela, verifique o plano da consulta (também conhecido como "plano de explicação") executando a instrução EXPLAIN. Para ver exemplos de como as instruções, cláusulas e expressões SQL afetam a saída de `EXPLAIN` para a consulta paralela, consulte Constructos do SQL para consulta paralela no Aurora MySQL.

O exemplo a seguir demonstra a diferença entre um plano de consulta tradicional e um plano de consulta paralela. Este plano de explicação é da Consulta 3 do teste comparativo da TPC-H. Muitos dos exemplos de consulta em toda esta seção usam as tabelas do conjunto de dados da TPC-H. Você pode obter as definições da tabela, as consultas e o programa `dbgen` que gera dados de amostra do site da TPC-H.

```
EXPLAIN SELECT l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) AS revenue,
  o_orderdate,
  o_shippriority
FROM customer,
  orders,
  lineitem
WHERE c_mktsegment = 'AUTOMOBILE'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < date '1995-03-13'
AND l_shipdate > date '1995-03-13'
GROUP BY l_orderkey,
  o_orderdate,
  o_shippriority
ORDER BY revenue DESC,
  o_orderdate LIMIT 10;
```

Por padrão, a consulta pode ter um plano como o seguinte. Se você não vir a junção de hash utilizada no plano de consulta, verifique primeiro se a otimização está habilitada.

```
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra                                              |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |  1480234 |    10.00 | Using where; Using temporary; Using filesort       |
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 14875240 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 59270573 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
```

Para o Aurora MySQL versão 3, é possível habilitar a junção de hash no nível da sessão ao emitir a instrução a seguir.

```
SET optimizer_switch='block_nested_loop=on';
```

Para o Aurora MySQL 2.09 e versões posteriores, defina o parâmetro de banco de dados ou parâmetro de cluster de banco de dados `aurora_disable_hash_join` como `0` (desativado). Desativar `aurora_disable_hash_join` define o valor de `optimizer_switch` para `hash_join=on`.

Depois de ativar a junção de hash, tente executar a instrução `EXPLAIN` novamente. Para obter informações sobre como habilitar junções de hash e usá-las de maneira eficaz, consulte Otimizando grandes consultas de junção do Aurora MySQL com junções hash.

Com a junção de hash habilitada, mas a consulta paralela desabilitada, a consulta pode ter um plano semelhante ao seguinte, que utiliza a junção de hash, mas não utiliza a consulta paralela.

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
| id | select_type | table    |...| rows      | Extra
                          |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

```
|  1 | SIMPLE      | customer |...|   5798330 | Using where; Using index; Using
 temporary; Using filesort       |
|  1 | SIMPLE      | orders   |...| 154545408 | Using where; Using join buffer (Hash
 Join Outer table orders)   |
|  1 | SIMPLE      | lineitem |...| 606119300 | Using where; Using join buffer (Hash
 Join Outer table lineitem) |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

Com a consulta paralela habilitada, duas etapas nessa consulta podem utilizar a otimização de consulta paralela, conforme mostrado na coluna `Extra` na saída `EXPLAIN`. O processamento intensivo de E/S e da CPU nessas etapas é empurrado para a camada de armazenamento.

```
+----+...
+-------------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                                    |
+----+...
+-------------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using index; Using temporary; Using filesort
                                                    |
|  1 |...| Using where; Using join buffer (Hash Join Outer table orders); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra)   |
|  1 |...| Using where; Using join buffer (Hash Join Outer table lineitem); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------------
+
```

Para obter informações sobre como interpretar a saída de `EXPLAIN` para uma consulta paralela e as partes das instruções SQL nas quais a consulta paralela pode ser aplicada, consulte Constructos do SQL para consulta paralela no Aurora MySQL.

O exemplo de saída a seguir mostra os resultados da execução da consulta anterior em uma instância db.r4.2xlarge com um grupo de buffers ociosos. A execução da consulta é substancialmente mais rápida quando a consulta paralela é usada.

> **Note**
>
> Como as cronometragens podem depender de vários fatores relativos ao ambiente, seus resultados podem ser diferentes. Sempre faça testes de performance para confirmar suas descobertas com seu próprio ambiente, sua workload, e assim por diante.

```
-- Without parallel query
+------------+------------+------------+----------------+
| l_orderkey | revenue    | o_orderdate | o_shippriority |
+------------+------------+------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+------------+------------+----------------+
10 rows in set (24 min 49.99 sec)
```

```
-- With parallel query
+------------+------------+------------+----------------+
| l_orderkey | revenue    | o_orderdate | o_shippriority |
+------------+------------+------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+------------+------------+----------------+
10 rows in set (1 min 49.91 sec)
```

Muitas das consultas de exemplo em toda esta seção usam as tabelas do conjunto de dados TPC-H, particularmente a tabela PART, que tem 20 milhões de linhas e a definição a seguir.

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| p_partkey     | int(11)       | NO   | PRI | NULL    |       |
| p_name        | varchar(55)   | NO   |     | NULL    |       |
| p_mfgr        | char(25)      | NO   |     | NULL    |       |
| p_brand       | char(10)      | NO   |     | NULL    |       |
| p_type        | varchar(25)   | NO   |     | NULL    |       |
| p_size        | int(11)       | NO   |     | NULL    |       |
```

```
| p_container   | char(10)      | NO   |     | NULL    |       |
| p_retailprice | decimal(15,2) | NO   |     | NULL    |       |
| p_comment     | varchar(23)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Faça experiências com sua workload para ter uma noção de se as instruções SQL individuais poderão aproveitar a consulta paralela. Depois, use as técnicas de monitoramento a seguir para ajudar a verificar com que frequência a consulta paralela é usada em cargas de trabalho reais ao longo do tempo. Para cargas de trabalho reais, fatores extras como os limites de simultaneidade são aplicáveis.

Monitorar a consulta paralela do Aurora MySQL

Se seu cluster do Aurora MySQL usar consulta paralela, você poderá ver um aumento nos valores de `VolumeReadIOPS`. As consultas paralelas não usam o grupo de buffers. Assim, embora as consultas sejam rápidas, esse processamento otimizado pode resultar em aumento nas operações de leitura e nas cobranças associadas.

Além das métricas do Amazon CloudWatch descritas em Visualizar métricas no console do Amazon RDS, o Aurora fornece outras variáveis de status globais. Você pode usar essas variáveis de status global para ajudar a monitorar a execução de consultas paralelas. Elas podem fornecer informações sobre por que o otimizador pode usar ou não a consulta paralela em determinada situação. Para acessar essas variáveis, use o comando `SHOW GLOBAL STATUS`. Veja também essas variáveis listadas a seguir.

Uma sessão de consulta paralela não é necessariamente um mapeamento de um para um com as consultas executadas pelo banco de dados. Por exemplo, suponha que o plano de consulta tem duas etapas que usam consulta paralela. Neste caso, a consulta envolve duas sessões paralelas e os contadores das tentativas de solicitações, e as solicitações bem-sucedidas são acrescidas de dois.

Quando você experimenta a consulta paralela executando as instruções `EXPLAIN`, espere ver aumentos nos contadores designados como "não escolhidos" mesmo que, na realidade, as consultas não estejam sendo executadas. Quando você trabalhar com uma consulta paralela na produção, poderá verificar se os contadores "não escolhidos" são acrescidos com maior rapidez do que o esperado. Nesse caso, você pode ajustar para que a paralela seja executada para as consultas que deseja. Para fazer isso, é possível alterar as configurações do cluster, a combinação de consultas, as instâncias de banco de dados nas quais a consulta paralela está habilitada e assim por diante.

Esse contadores são rastreados no nível da instância de banco de dados. Quando você se conectar a um endpoint diferente, poderá ver métricas diferentes porque cada instância de banco de dados executa seu conjunto próprio de consultas paralelas. Você também poderá ver métricas diferentes quando o endpoint de leitura se conectar a uma instância de banco de dados diferente para cada sessão.

Nome	Descrição
`Aurora_pq_bytes_returned`	O número de bytes de estruturas de dados de tupla transmitidos para o nó de cabeçalho durante as consultas paralelas. Divida por 16.384 para comparar com `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	O número máximo de sessões de consulta paralela que podem ser executadas simultaneamente nesta instância de bancos de dados Aurora. Esse é um número fixo, que depende da classe da instância de banco de dados da AWS.
`Aurora_pq_pages_pushed_down`	O número de páginas de dados (cada uma com um tamanho fixo de 16 KiB) em que a consulta paralela evitou uma transmissão de rede para o nó de cabeçalho.
`Aurora_pq_request_attempted`	O número de sessões de consultas paralelas solicitadas. Esse valor pode representar mais de uma sessão por consulta, dependendo dos elementos SQL, como subconsultas e junções.
`Aurora_pq_request_executed`	O número de sessões de consultas paralelas executadas com êxito.
`Aurora_pq_request_failed`	O número de sessões de consultas paralelas que retornaram um erro para o cliente. Em alguns casos, uma solicitação por uma consulta paralela pode falhar, por exemplo,

Nome	Descrição
	devido a um problema na camada de armazenamento. Nesses casos, a parte da consulta que falhou é reprocessada usando um mecanismo de consulta não paralelo. Se a consulta reprocessada também falhar, será retornado um erro para o cliente e o contador será incrementado.
`Aurora_pq_request_in_progress`	O número de sessões de consultas paralelas em andamento no momento. Esse número se aplica à instância de bancos de dados Aurora em particular à qual você está conectado, e não a todo o cluster de bancos de dados Aurora. Para saber se uma instância de banco de dados está próxima do limite de simultaneidade, compare este valor a `Aurora_pq_max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	O número de vezes em que a consulta paralela não foi escolhida para atender uma consulta. Este valor é a soma de vários outros contadores mais granulares. Uma instrução `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_below_min_rows`	O número de vezes em que a consulta paralela não foi escolhida devido ao número de linhas na tabela. Uma declaração `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_column_bit`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela devido a um tipo de dados sem suporte na lista de colunas projetadas.

Nome	Descrição
`Aurora_pq_request_not_chosen_column_geometry`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem colunas com o tipo de dados `GEOMETRY`. Para saber mais sobre as versões do Aurora MySQL que removem essa limitação, consulte Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3.
`Aurora_pq_request_not_chosen_column_lob`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela inclui colunas com um tipo de dados `LOB` ou colunas `VARCHAR` que são armazenadas externamente devido ao comprimento declarado. Para saber mais sobre as versões do Aurora MySQL que removem essa limitação, consulte Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3.
`Aurora_pq_request_not_chosen_column_virtual`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela contém uma coluna virtual.
`Aurora_pq_request_not_chosen_custom_charset`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem colunas com um conjunto de caracteres personalizado.

Nome	Descrição
`Aurora_pq_request_not_chosen_fast_ddl`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela está sendo alterada atualmente por uma instrução `ALTER` DDL rápida.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	O número de vezes em que a consulta paralela não foi escolhida, ainda que menos de 95 por cento dos dados da tabela já estivessem no grupo de buffers, porque não havia uma quantidade suficiente de dados da tabela fora do buffer que fizesse a consulta paralela valer a pena.
`Aurora_pq_request_not_chosen_full_text_index`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem índices de texto completo.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	O número de vezes em que a consulta paralela não foi escolhida porque uma alta porcentag em de dados da tabela (no momento, maior do que 95 por cento) já se encontrava no grupo de buffers. Nesses casos, o otimizador determina que a leitura dos dados a partir do grupo de buffers é mais eficiente. Uma declaraçã o `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_index_hint`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta inclui uma dica de índice.

Nome	Descrição
`Aurora_pq_request_not_chosen_innodb_table_format`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela usa um formato de linha InnoDB que não é compatível. A consulta paralela do Aurora só se aplica aos formatos de linha `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	O número de solicitações de consultas paralelas que usaram o caminho de processamento de consultas não paralelas, devido à execução da consulta ter sido iniciada dentro de uma transação de execução demorada. Uma declaração `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_no_where_clause`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta não inclui nenhuma cláusula `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta usa uma verificação de intervalo em um índice.
`Aurora_pq_request_not_chosen_row_length_too_long`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque o comprimento total combinado de todas as colunas é muito longo.

Nome	Descrição
`Aurora_pq_request_not_chosen_small_table`	O número de vezes em que a consulta paralela não foi escolhida devido ao tamanho total da tabela, o que é determinado pelo número de linhas e pelo comprimento médio da linha. Uma declaração `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_temporary_table`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta se refere a tabelas temporárias que usam os tipos de tabela `MyISAM` ou `memory` que não têm suporte.
`Aurora_pq_request_not_chosen_tx_isolation`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta usa um nível de isolamento de transação sem suporte. Em instâncias de banco de dados do leitor, a consulta paralela se aplica somente aos níveis de isolamento `REPEATABLE READ` e `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta faz parte de uma instrução `UPDATE` ou `DELETE`.

Nome	Descrição
`Aurora_pq_request_not_chosen_unsupported_access`	O número de solicitações de consultas paralelas que usam o caminho de processamento de consultas não paralelas porque a cláusula `WHERE` não atende aos critérios de consulta paralela. Esse resultado pode ocorrer se a consulta não exigir uma varredura de dados intensiva ou se a consulta for uma instrução `DELETE` ou `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	O número de solicitações de consulta paralelas que usam o caminho de processamento de consulta não paralelo porque o cluster de banco de dados do Aurora MySQL não está usando uma configuração de armazenamento de cluster compatível do Aurora. Esse parâmetro está disponível no Aurora MySQL versão 3.04 e posterior. Para ter mais informações, consulte Limitações.
`Aurora_pq_request_throttled`	O número de vezes em que a consulta paralela não foi escolhida devido ao número máximo de consultas paralelas simultâneas em execução em uma determinada instância de bancos de dados Aurora.

Constructos do SQL para consulta paralela no Aurora MySQL

Na seção a seguir, você pode encontrar mais detalhes sobre por que determinadas instruções SQL usam ou não usam a consulta paralela. Esta seção também detalha como os recursos do Aurora MySQL interagem com a consulta paralela. Essas informações podem ajudar a diagnosticar problemas de performance de um cluster que usa a consulta paralela ou a entender como a consulta paralela pode ser aplicada em sua workload específica.

A decisão de usar a consulta paralela depende de muitos fatores que ocorrem no momento em que a instrução está em execução. Assim, a consulta paralela pode ser usada para algumas consultas sempre, nunca ou somente sob determinadas condições.

> **Tip**
>
> Ao visualizar esses exemplos em HTML, você pode usar o widget Copy (Copiar) no canto superior direito de cada listagem de código para copiar o código SQL e testar por conta própria. O uso do widget Copy (Copiar) evita copiar os caracteres adicionais ao redor das linhas de prompt `mysql>` e de continuação `->`.

Tópicos

- Instrução EXPLAIN
- Cláusula WHERE
- Linguagem de definição de dados (DDL)
- Tipos de dados de coluna
- Tabelas particionadas
- Funções agregadas, cláusulas GROUP BY e cláusulas HAVING.
- Chamadas de funções na cláusula WHERE
- Cláusula LIMIT
- Operadores de comparação
- Junções
- Subconsultas
- UNION
- Visões
- Instruções na linguagem de manipulação de dados (DML)
- Transações e bloqueio
- Índices de árvore B
- Índices de pesquisa de texto completo (FTS)
- Virtual columns
- Mecanismos de armazenamento em cache integrados
- Dicas do Optimizer

- Tabelas temporárias MyISAM

Instrução EXPLAIN

Conforme mostrado em exemplos ao longo desta seção, a instrução `EXPLAIN` indica se cada estágio de uma consulta se qualifica ou não à consulta paralela. Ela também indica quais aspectos de uma consulta são empurrados para a camada de armazenamento. Os itens mais importantes no plano de consulta são os seguintes:

- Um valor diferente de `NULL` na coluna `key` sugere que a consulta pode ser realizada de maneira eficiente usando buscas com índices, e que a consulta paralela é improvável.
- Um valor pequeno na coluna `rows` (um valor que não esteja na casa dos milhões) sugere que a consulta não acessa uma quantidade de dados suficiente para fazer a consulta paralela valer a pena. Isso significa que consulta paralela é improvável.
- A coluna `Extra` mostra se uma consulta paralela é esperada. Essa saída é semelhante ao exemplo a seguir.

  ```
  Using parallel query (A columns, B filters, C exprs; D extra)
  ```

  O número `columns` representa quantas colunas são referenciadas no bloco de consultas.

  O número `filters` representa o número de predicados de `WHERE` que representam uma comparação simples de um valor de coluna com uma constante. A comparação pode ser de igualdade, desigualdade ou intervalo. O Aurora pode paralelizar esses tipos de predicados com mais eficiência.

  O número `exprs` representa o número de expressões, como chamadas de funções, operadores ou outras expressões que também podem ser paralelizadas, embora não com a mesma eficiência que uma condição de filtro.

  O número `extra` representa quantas expressões não podem ser empurradas e são executadas pelo nó de cabeçalho.

Por exemplo, considere a seguinte saída da instrução `EXPLAIN`.

```
mysql> explain select p_name, p_mfgr from part
    -> where p_brand is not null
    -> and upper(p_type) is not null
```

```
    -> and round(p_retailprice) is not null;
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table |...| rows     | Extra
                                |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part  |...| 20427936 | Using where; Using parallel query (5
 columns, 1 filters, 2 exprs; 0 extra) |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
```

A informação da coluna `Extra` mostra que cinco colunas são extraídas de cada linha para avaliar as condições da consulta e construir o conjunto de resultados. Um predicado de `WHERE` envolve um filtro, isto é, uma coluna que é diretamente testada na cláusula `WHERE`. Duas cláusulas `WHERE` exigem a avaliação das expressões mais complicadas, nesse caso envolvendo chamadas de funções. O campo `0 extra` confirma que todas as operações na cláusula `WHERE` são empurradas para a camada de armazenamento como parte do processamento de consulta paralela.

Nos casos em que a consulta paralela não é escolhida, normalmente é possível deduzir a razão a partir das outras colunas da saída de `EXPLAIN`. Por exemplo, o valor de `rows` pode ser muito pequeno ou a coluna `possible_keys` pode indicar que a consulta pode usar uma busca com índices, em vez de uma varredura com muitos dados. O exemplo a seguir mostra uma consulta em que o otimizador pode estimar que a consulta verificará apenas um pequeno número de linhas. Ele faz isso com base nas características da chave primária. Nesse caso, a consulta paralela não é necessária.

```
mysql> explain select count(*) from part where p_partkey between 1 and 100;
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows |
 Extra                    |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
|  1 | SIMPLE      | part  | range | PRIMARY       | PRIMARY | 4       | NULL |   99 |
 Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
```

A saída que mostra se a consulta paralela será usada leva em conta todos os fatores disponíveis no momento em que a instrução `EXPLAIN` é executada. O otimizador pode fazer uma escolha

diferente quando a consulta é de fato executada, caso a situação tenha sido alterada nesse meio-tempo. Por exemplo, `EXPLAIN` pode relatar que uma instrução usará a consulta paralela. Porém, quando a consulta é executada mais tarde, ela poderá não usar a consulta paralela devido às condições atuais. Essas condições podem incluir várias outras consultas paralelas em execução simultaneamente. Elas também podem incluir linhas sendo excluídas da tabela, um novo índice sendo criado, tempo excessivo em uma transação aberta, e assim por diante.

Cláusula WHERE

Para que uma consulta use a otimização de consulta paralela, ela deve incluir uma cláusula `WHERE`.

A otimização de consulta paralela acelera vários tipos de expressões usadas na cláusula `WHERE`:

- Comparações simples do valor de uma coluna com uma constante, conhecidas como filtros. Essas comparações são as mais beneficiadas pela transferência para a camada de armazenamento. O número de expressões de filtro em uma consulta é relatado na saída de `EXPLAIN`.
- Outros tipos de expressões na cláusula `WHERE` também são empurradas para a camada de armazenamento, onde for possível. O número de expressões como essas em uma consulta é relatado na saída de `EXPLAIN`. Essas expressões podem ser chamadas de funções, operadores `LIKE`, expressões `CASE`, e assim por diante.
- No momento, determinadas funções e operadores não são empurrados pela consulta paralela. O número dessas expressões em uma consulta é relatado como o contador `extra` na saída de `EXPLAIN`. O restante da consulta ainda pode usar a consulta paralela.
- Embora as expressões na lista selecionada não sejam empurradas, as consultas que contêm essas funções ainda podem se beneficiar com a redução do tráfego de rede dos resultados intermediários de consultas paralelas. Por exemplo, as consultas que chamam funções de agregação na lista selecionada podem se beneficiar da consulta paralela, ainda que as funções de agregação não sejam empurradas.

Por exemplo, a consulta a seguir faz uma varredura completa na tabela e processa todos os valores da coluna `P_BRAND`. No entanto, ela não usa a consulta paralela porque não inclui nenhuma cláusula `WHERE`.

```
mysql> explain select count(*), p_brand from part group by p_brand;
+----+-------------+-------+------+---------------+------+---------+------+----------
+---------------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows     |
 Extra                           |
```

```
+----+-------------+-------+------+---------------+------+---------+------+----------
+---------------------------------+
|  1 | SIMPLE      | part  | ALL  | NULL          | NULL | NULL    | NULL | 20427936 |
 Using temporary; Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+----------
+---------------------------------+
```

Por outro lado, a seguinte consulta inclui predicados de `WHERE` que filtram os resultados e, portanto, a consulta paralela pode ser aplicada:

```
mysql> explain select count(*), p_brand from part where p_name is not null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
    -> group by p_brand;
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
| id |...| rows     | Extra
                                          |
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
|  1 |...| 20427936 | Using where; Using temporary; Using filesort; Using parallel
 query (5 columns, 1 filters, 2 exprs; 0 extra) |
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
```

Se o otimizador estimar que o número de linhas retornadas para um bloco de consultas for pequeno, a consulta paralela não será usada para esse bloco. O exemplo a seguir mostra um caso em que um operador "maior que" na coluna da chave primária é aplicado em milhões de linhas, o que determina o uso da consulta paralela. O teste inverso com "menor que" estima a aplicação em algumas linhas apenas e não usa a consulta paralela.

```
mysql> explain select count(*) from part where p_partkey > 10;
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
```

```
+----+...+----------
+-------------------------------------------------------------------------+

mysql> explain select count(*) from part where p_partkey < 10;
+----+...+------+-------------------------+
| id |...| rows | Extra                   |
+----+...+------+-------------------------+
|  1 |...|    9 | Using where; Using index |
+----+...+------+-------------------------+
```

Linguagem de definição de dados (DDL)

No Aurora MySQL versão 2, a consulta paralela só está disponível para tabelas cujas operações de linguagem de definição de dados (DDL) rápida estão pendentes. No Aurora MySQL versão 3, é possível utilizar a consulta paralela em uma tabela ao mesmo tempo que uma operação de DDL instantânea.

A DDL instantânea no Aurora MySQL versão 3 substitui o recurso de DDL rápida no Aurora MySQL versão 2. Para saber mais sobre a DDL instantânea, consulte DDL instantânea (Aurora MySQL versão 3).

Tipos de dados de coluna

No Aurora MySQL versão 3, a consulta paralela pode funcionar com tabelas que contêm colunas com os tipos de dados `TEXT`,`BLOB`,`JSON` e`GEOMETRY`. Ela também pode funcionar com colunas `VARCHAR` e `CHAR` com um comprimento máximo declarado superior a 768 bytes. Se a consulta fizer referência a colunas que contenham tipos de objetos grandes, o trabalho adicional para recuperá-las adicionará uma certa sobrecarga ao processamento das consultas. Nesse caso, verifique se a consulta é capaz de omitir as referências a essas colunas. Caso contrário, execute benchmarks para confirmar se essas consultas são mais rápidas com a consulta paralela habilitada ou desabilitada.

No Aurora MySQL versão 2, a consulta paralela apresenta estas limitações para tipos de objetos grandes:

- Os tipos de dados `TEXT`, `BLOB`, `JSON` e `GEOMETRY` não são compatíveis com a consulta paralela. Uma consulta que faça referência a qualquer coluna desses tipos não pode usar a consulta paralela.
- Colunas de tamanho variável (tipos de dados `VARCHAR` e `CHAR`) são compatíveis com a consulta paralela até o tamanho máximo declarado de 768 bytes. Uma consulta que faça referência a qualquer coluna de um tipo declarado com um tamanho máximo maior não poderá usar a consulta

paralela. Para colunas que usam conjuntos de caracteres multibyte, o limite de bytes leva em conta o número máximo de bytes no conjunto de caracteres. Por exemplo, para o conjunto de caracteres `utf8mb4` (que tem um tamanho máximo de caracteres de 4 bytes), uma coluna `VARCHAR(192)` é compatível com a consulta paralela, mas uma coluna `VARCHAR(193)` não é.

Tabelas particionadas

É possível utilizar tabelas particionadas com consulta paralela no Aurora MySQL versão 3. Como essas tabelas particionadas são representadas internamente como várias tabelas menores, uma consulta que utiliza a consulta paralela em uma tabela não particionada pode não utilizar a consulta paralela em uma tabela particionada idêntica. O Aurora MySQL considera se cada partição é grande o suficiente a ponto de se qualificar para a otimização de consulta paralela, em vez de avaliar o tamanho da tabela inteira. Verifique se a variável de status `Aurora_pq_request_not_chosen_small_table` é incrementada quando uma consulta em uma tabela particionada não utiliza a consulta paralela na ocasião esperada.

Por exemplo, considere uma tabela particionada com `PARTITION BY HASH (`*`column`*`) PARTITIONS 2` e outra tabela particionada com `PARTITION BY HASH (`*`column`*`) PARTITIONS 10`. Na tabela com duas partições, estas são cinco vezes maiores que a tabela com dez partições. Portanto, é mais provável que a consulta paralela seja utilizada para consultas na tabela contendo menos partições. No exemplo a seguir, a tabela `PART_BIG_PARTITIONS` apresenta duas partições e `PART_SMALL_PARTITIONS` apresenta dez partições. Com dados idênticos, são maiores as chances de que a consulta paralela seja utilizada para a tabela com menos partições grandes.

```
mysql> explain select count(*), p_brand from part_big_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
| id | select_type | table               | partitions | Extra
                                                                                          |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
|  1 | SIMPLE      | part_big_partitions | p0,p1      | Using where; Using temporary;
 Using parallel query (4 columns, 1 filters, 1 exprs; 0 extra; 1 group-bys, 1 aggrs) |
```

```
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+

mysql> explain select count(*), p_brand from part_small_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+----------------------+------------------------------
+------------------------------+
| id | select_type | table                | partitions                   | Extra
                   |
+----+-------------+----------------------+------------------------------
+------------------------------+
|  1 | SIMPLE      | part_small_partitions | p0,p1,p2,p3,p4,p5,p6,p7,p8,p9 | Using
 where; Using temporary |
+----+-------------+----------------------+------------------------------
+------------------------------+
```

Funções agregadas, cláusulas GROUP BY e cláusulas HAVING.

As consultas que envolvem as funções agregadas frequentemente são boas candidatas ao uso da consulta paralela, pois envolvem a varredura de grandes números de linhas em tabelas grandes.

No Aurora MySQL 3, a consulta paralela pode otimizar chamadas de função agregadas na lista de seleção e na cláusula `HAVING`.

Antes do Aurora MySQL 3, chamadas de funções agregadas na lista selecionada ou na cláusula `HAVING` não são enviadas para a camada de armazenamento. No entanto, a consulta paralela ainda pode melhorar a performance dessas consultas com funções agregadas. Para isso, ela primeiro extrai os valores das colunas das páginas de dados brutos em paralelo na camada de armazenamento. Em seguida, ela transmite esses valores de volta para o nó de cabeçalho em um formato de tupla compacta, em vez de páginas inteiras de dados. Como sempre, a consulta precisa ter pelo menos um predicado `WHERE` para que a consulta paralela seja ativada.

Os exemplos simples a seguir ilustram os tipos de consultas agregadas que podem se beneficiar da consulta paralela. Eles fazem isso retornando resultados intermediários de forma compacta para o nó de cabeçalho ou filtrando as linhas que não encontram correspondência dos resultados intermediários, ou ambos.

```
mysql> explain select sql_no_cache count(distinct p_brand) from part where p_mfgr =
 'Manufacturer#5';
```

```
+----+...+-------------------------------------------------------------------------------+
| id |...| Extra                                                                         |
+----+...+-------------------------------------------------------------------------------+
|  1 |...| Using where; Using parallel query (2 columns, 1 filters, 0 exprs; 0 extra) |
+----+...+-------------------------------------------------------------------------------+

mysql> explain select sql_no_cache p_mfgr from part where p_retailprice > 1000 group by
 p_mfgr having count(*) > 100;
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (3
 columns, 0 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Chamadas de funções na cláusula WHERE

O Aurora pode aplicar a otimização de consulta paralela em chamadas para a maioria das funções incorporadas na cláusula `WHERE`. A paralelização dessas chamadas de funções alivia alguma workload da CPU a partir do nó de cabeçalho. A avaliação das funções de predicado em paralelo durante o estágio inicial da consulta ajuda o Aurora a minimizar a quantidade de dados transmitidos e processados nos estágios posteriores.

No momento, a paralelização não se aplica às chamadas de funções na lista selecionada. Essas funções são avaliadas pelo nó de cabeçalho, mesmo que as chamadas de funções idênticas apareçam na cláusula `WHERE`. Os valores originais das colunas relevantes são incluídos nas tuplas transmitidas dos nós de armazenamento de volta até o nó de cabeçalho. O nó de cabeçalho realiza qualquer transformação, como `UPPER`, `CONCATENATE` e assim por diante, para produzir os valores finais do conjunto de resultados.

No exemplo a seguir, a consulta paralela paraleliza a chamada `LOWER` porque ela aparece na cláusula `WHERE`. A consulta paralela não afeta as chamadas `SUBSTR` e `UPPER` porque elas aparecem na lista selecionada.

```
mysql> explain select sql_no_cache distinct substr(upper(p_name),1,5) from part
    -> where lower(p_name) like '%cornflower%' or lower(p_name) like '%goldenrod%';
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
               |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using parallel query (2 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

As mesmas considerações se aplicam a outras expressões, como as expressões `CASE` ou os operadores `LIKE`. O exemplo a seguir mostra que a consulta paralela avalia a expressão `CASE` e os operadores `LIKE` na cláusula `WHERE`.

```
mysql> explain select p_mfgr, p_retailprice from part
    -> where p_retailprice > case p_mfgr
    ->   when 'Manufacturer#1' then 1000
    ->   when 'Manufacturer#2' then 1200
    ->   else 950
    -> end
    -> and p_name like '%vanilla%'
    -> group by p_retailprice;
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (4
 columns, 0 filters, 2 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

Cláusula LIMIT

No momento, a consulta paralela não é usada para blocos de consulta que incluam uma cláusula `LIMIT`. A consulta paralela ainda pode ser usada em fases anteriores da consulta com `GROUP by`, `ORDER BY` ou junções.

Operadores de comparação

O otimizador estima quantas linhas serão examinadas para avaliar os operadores de comparação e determina se vai usar a consulta paralela com base nessa estimativa.

O primeiro exemplo, a seguir, mostra que uma comparação de igualdade com uma coluna de chave primária pode ser realizada de maneira eficiente sem a consulta paralela. O segundo exemplo, a seguir, mostra que uma comparação semelhante com uma coluna não indexada exige a varredura de milhões de linhas e, portanto, pode se beneficiar da consulta paralela.

```
mysql> explain select * from part where p_partkey = 10;
+----+...+------+-------+
| id |...| rows | Extra |
+----+...+------+-------+
|  1 |...|    1 | NULL  |
+----+...+------+-------+

mysql> explain select * from part where p_type = 'LARGE BRUSHED BRASS';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

As mesmas considerações se aplicam aos testes de desigualdade e a comparações com intervalos, como menor que, maior que, igual a ou `BETWEEN`. O otimizador estima o número de linhas que serão examinadas e determina se vale a pena usar a consulta paralela com base no volume total de E/S.

Junções

As consultas que usam junções com grandes tabelas normalmente envolvem operações com muitos dados que se beneficiam da otimização de consulta paralela. As comparações de valores de colunas entre várias tabelas (ou seja, os próprios predicados das junções) não são paralelizadas no momento. No entanto, a consulta paralela pode empurrar uma parte do processamento interno de outras fases da junção, como a construção do filtro Bloom durante uma junção hash. A consulta paralela pode ser aplicada a consultas de junções, mesmo sem uma cláusula `WHERE`. Portanto, uma consulta de junção é uma exceção à regra de que uma cláusula `WHERE` é necessária para o uso da consulta paralela.

Cada fase do processamento da junção é avaliado para verificar se ela se qualifica para a consulta paralela. Se mais de uma fase puder usar a consulta paralela, essas fases serão executadas em sequência. Assim, cada consulta de junção conta como uma única sessão de consulta paralela em termos de limites de simultaneidade.

Por exemplo, quando uma consulta de junção incluir predicados `WHERE` para filtrar as linhas de uma das tabelas associadas, essa opção de filtragem poderá usar a consulta paralela. Em um outro exemplo, vamos supor que uma consulta de junção use o mecanismo de junção hash para unir uma tabela grande a uma tabela pequena. Nesse caso, a varredura da tabela para produzir a estrutura de dados do filtro Bloom pode conseguir usar a consulta paralela.

> **Note**
>
> A consulta paralela é normalmente usada para tipos de consultas que consomem muitos recursos e que se beneficiam da otimização de junções de hash. O método de ativação da otimização de junções de hash depende da versão do Aurora MySQL. Para obter detalhes sobre cada versão, consulte Habilitar a junção de hash para clusters de consulta paralela. Para obter informações sobre como habilitar junções de hash e usá-las de maneira eficaz, consulte Otimizando grandes consultas de junção do Aurora MySQL com junções hash.

```
mysql> explain select count(*) from orders join customer where o_custkey = c_custkey;
+----+...+----------+-------+---------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
| id |...| table    | type  | possible_keys | key         |...| rows      | Extra

              |
```

```
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
|  1 |...| customer | index | PRIMARY       | c_nationkey |...|  15051972 | Using index

              |
|  1 |...| orders   | ALL   | o_custkey     | NULL        |...| 154545408 | Using join
 buffer (Hash Join Outer table orders); Using parallel query (1 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
```

Para uma consulta de junção que usa o mecanismo de loop aninhado, o bloco de loop aninhado mais externo poderá usar a consulta paralela. O uso da consulta paralela depende dos mesmos fatores de sempre, como a presença de condições de filtragem adicionais na cláusula `WHERE`.

```
mysql> -- Nested loop join with extra filter conditions can use parallel query.
mysql> explain select count(*) from part, partsupp where p_partkey != ps_partkey and
 p_name is not null and ps_availqty > 0;
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table    |...| rows     | Extra
                                  |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part     |...| 20427936 | Using where; Using parallel query (2
 columns, 1 filters, 0 exprs; 0 extra) |
|  1 | SIMPLE      | partsupp |...| 78164450 | Using where; Using join buffer (Block
 Nested Loop)                          |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
```

Subconsultas

O bloco de consulta externa e o bloco de subconsulta interna podem usar a consulta paralela ou não. Isso é determinado com base nas características habituais da tabela, na cláusula `WHERE` e assim por diante, para cada bloco. Por exemplo, a consulta a seguir usa a consulta paralela para o bloco de subconsultas mas não para o bloco mais externo.

```
mysql> explain select count(*) from part where
```

```
   --> p_partkey < (select max(p_partkey) from part where p_name like '%vanilla%');
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type |...| rows     | Extra
                       |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY     |...|     NULL | Impossible WHERE noticed after reading const tables
                       |
|  2 | SUBQUERY    |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

No momento, as subconsultas correlacionadas não podem usar a otimização de consulta paralela.

UNION

Cada bloco de consultas em uma consulta `UNION` pode usar a consulta paralela ou não, dependendo das características da tabela, da cláusula `WHERE` etc., para cada parte da `UNION`.

```
mysql> explain select p_partkey from part where p_name like '%choco_ate%'
    -> union select p_partkey from part where p_name like '%vanil_a%';
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type    |...| rows     | Extra
                          |
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY        |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
|  2 | UNION          |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
| NULL | UNION RESULT | <union1,2> |...|     NULL | Using temporary
                        |
+----+--------------+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Cada cláusula `UNION` dentro da consulta é executada sequencialmente. Mesmo que a consulta inclua vários estágios que utilizem a consulta paralela, ela executa apenas uma

única consulta paralela de cada vez. Portanto, mesmo uma consulta complexa com vários estágios conta apenas como sendo uma em relação ao limite de consultas paralelas simultâneas.

Visões

O otimizador reescreve as consultas que usam visualizações como consultas mais longas que usam tabelas subjacentes. Assim, a consulta paralela funciona da mesma forma, sejam as referências de tabelas visualizações ou tabelas reais. Todas as mesmas considerações que se aplicam ao uso de consultas paralelas para uma consulta, e a quais partes são empurradas, são aplicáveis à consulta final que é reescrita.

Por exemplo, o plano de consulta a seguir mostra uma definição de visualização que normalmente não usa a consulta paralela. Quando a visualização é consultada com cláusulas `WHERE` adicionais, o Aurora MySQL usa a consulta paralela.

```
mysql> create view part_view as select * from part;
mysql> explain select count(*) from part_view where p_partkey is not null;
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 0 filters, 0 exprs;
 1 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

Instruções na linguagem de manipulação de dados (DML)

A instrução `INSERT` pode usar a consulta paralela na fase de processamento `SELECT`, caso a parte `SELECT` atenda às outras condições para usar a consulta paralela.

```
mysql> create table part_subset like part;
mysql> explain insert into part_subset select * from part where p_mfgr =
 'Manufacturer#1';
+----+...+----------
+-------------------------------------------------------------------------+
```

```
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Normalmente, após uma instrução `INSERT`, os dados das linhas recentemente inseridas ficam no grupo de buffers. Portanto, uma tabela pode não se qualificar para a consulta paralela imediatamente após a inserção de um grande número de linhas. Mais tarde, depois que os dados forem removidos do grupo de buffers durante uma operação normal, as consultas que acessam essa tabela poderão começar a usar a consulta paralela novamente.

A instrução `CREATE TABLE AS SELECT` não usa a consulta paralela, mesmo que a porção `SELECT` da instrução seja qualificada para usar a consulta paralela. O aspecto de DDL dessa instrução torna ela incompatível com o processamento de consultas paralelas. Por outro lado, na instrução `INSERT ... SELECT`, a porção `SELECT` pode usar a consulta paralela.

A consulta paralela nunca é usada em instruções `DELETE` ou `UPDATE`, independentemente do tamanho da tabela e dos predicados na cláusula `WHERE`.

```
mysql> explain delete from part where p_name is not null;
+----+-------------+...+----------+-------------+
| id | select_type |...| rows     | Extra       |
+----+-------------+...+----------+-------------+
|  1 | SIMPLE      |...| 20427936 | Using where |
+----+-------------+...+----------+-------------+
```

Transações e bloqueio

É possível usar todos os níveis de isolamento na instância primária do Aurora.

Em instâncias de banco de dados do leitor do Aurora, a consulta paralela se aplica a instruções executadas sob o nível de isolamento `REPEATABLE READ`. O Aurora MySQL versão 2.09 e

versões posteriores também podem usar o nível de isolamento `READ COMMITTED` em instâncias de banco de dados do leitor. O nível de isolamento padrão para instâncias de banco de dados do leitor do Aurora é `REPEATABLE READ`. Para usar o nível de isolamento `READ COMMITTED` em instâncias de banco de dados de leitor, é necessário definir a opção de configuração `aurora_read_replica_read_committed` no nível da sessão. O nível de isolamento `READ COMMITTED` para instâncias de leitor está em conformidade com o comportamento padrão SQL. No entanto, o isolamento é menos rigoroso em instâncias de leitor do que quando as consultas usam o nível de isolamento `READ COMMITTED` na instância de gravador.

Para saber mais sobre níveis de isolamento do Aurora, especialmente as diferenças em `READ COMMITTED` entre instâncias de gravador e de leitor, consulte Níveis de isolamento do Aurora MySQL.

Após a conclusão de uma grande transação, as estatísticas da tabela podem ficar obsoletas. Tais estatísticas obsoletas podem exigir o uso de uma instrução `ANALYZE TABLE` para que o Aurora possa estimar o número de linhas com precisão. Uma instrução em DML de grande escala também pode trazer uma porção substancial dos dados da tabela para o grupo de buffers. A presença desses dados no grupo de buffers pode levar à redução da frequência de uso da consulta paralela para essa tabela até que os dados sejam removidos do grupo.

Quando sua sessão está em uma transação demorada (por padrão, 10 minutos), as consultas adicionais dentro dessa sessão não usam a consulta paralela. Também é possível que o tempo limite seja esgotado durante uma consulta única de longa duração. Esse tipo de tempo limite esgotado pode ocorrer se a execução da consulta se estender por um tempo maior do que o intervalo máximo (atualmente, 10 minutos) antes de o processamento da consulta paralela começar.

Você pode reduzir a probabilidade de iniciar uma transação de longa duração acidentalmente definindo `autocommit=1` nas sessões `mysql`, onde você executa as consultas ad hoc (de execução única). Até mesmo uma instrução `SELECT` em uma tabela inicia uma transação criando uma visualização de leitura. Uma visualização de leitura é um conjunto de dados consistente para as consultas posteriores que permanecem até que a transação seja confirmada. Esteja ciente dessa restrição também ao usar aplicativos JDBC ou ODBC com o Aurora, pois eles podem estar sendo executados com a configuração de `autocommit` desativada.

O exemplo a seguir mostra como a execução de uma consulta em uma tabela com a configuração de `autocommit` desativada cria uma visualização de leitura que inicia implicitamente uma transação. As consultas executadas logo em seguida ainda podem usar a consulta paralela. No entanto, após uma pausa de vários minutos, as consultas não se qualificam mais para a consulta paralela. O

encerramento da transação com `COMMIT` ou `ROLLBACK` restaura a possibilidade de qualificação para a consulta paralela.

```
mysql> set autocommit=0;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+

mysql> select sleep(720); explain select sql_no_cache count(*) from part where
 p_retailprice > 10.0;
+------------+
| sleep(720) |
+------------+
|          0 |
+------------+
1 row in set (12 min 0.00 sec)

+----+...+---------+-------------+
| id |...| rows    | Extra       |
+----+...+---------+-------------+
|  1 |...| 2976129 | Using where |
+----+...+---------+-------------+

mysql> commit;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
```

```
+----+...+---------
+-------------------------------------------------------------------------------+
```

Para ver quantas vezes as consultas não foram qualificadas para a consulta paralela porque estavam dentro de transações de longa duração, verifique a variável de status `Aurora_pq_request_not_chosen_long_trx`.

```
mysql> show global status like '%pq%trx%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Aurora_pq_request_not_chosen_long_trx | 4     |
+--------------------------------------+-------+
```

As instruções `SELECT` que obtêm bloqueios, como as sintaxes `SELECT FOR UPDATE` ou `SELECT LOCK IN SHARE MODE`, não podem usar a consulta paralela.

A consulta paralela pode funcionar para uma tabela que seja bloqueada por uma instrução `LOCK TABLES`.

```
mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055';
+----+...+-----------
+-------------------------------------------------------------------------------+
| id |...| rows      | Extra
          |
+----+...+-----------
+-------------------------------------------------------------------------------+
|  1 |...| 154545408 | Using where; Using parallel query (3 columns, 1 filters, 0
 exprs; 0 extra) |
+----+...+-----------
+-------------------------------------------------------------------------------+

mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055' for update;
+----+...+-----------+-------------+
| id |...| rows      | Extra       |
+----+...+-----------+-------------+
|  1 |...| 154545408 | Using where |
+----+...+-----------+-------------+
```

Índices de árvore B

As estatísticas coletadas pela instrução `ANALYZE TABLE` ajudam o otimizador a decidir quando usar a consulta paralela ou as buscas com índice, com base nas características dos dados de cada coluna. Mantenha as estatísticas atualizadas com a execução de `ANALYZE TABLE` após as operações DML que fazem alterações substanciais nos dados dentro de uma tabela.

Se as buscas com índices podem executar uma consulta de maneira eficiente sem uma varredura com muitos dados, o Aurora pode usar buscas com índices. Isso evita a sobrecarga do processamento de consultas paralelas. Também existem limites de simultaneidade para o número de consultas paralelas que podem ser executadas simultaneamente em um cluster de bancos de dados Aurora. Certifique-se de usar as práticas recomendadas na indexação de suas tabelas, de modo que suas consultas mais frequentes e aquelas que são mais usadas de forma simultânea utilizem buscas com índices.

Índices de pesquisa de texto completo (FTS)

No momento, a consulta paralela não é usada em tabelas que contêm um índice de pesquisa de texto completo, independentemente de a consulta fazer referência ou não a colunas indexadas dessa forma ou usar o operador `MATCH`.

Virtual columns

No momento, a consulta paralela não é usada para tabelas que contêm uma coluna virtual, independentemente de a consulta se referir a qualquer coluna virtual.

Mecanismos de armazenamento em cache integrados

O Aurora inclui mecanismos de armazenamento em cache incorporados: o grupo de buffers e o cache de consultas. O otimizador do Aurora escolhe um entre esses mecanismos de armazenamento em cache e a consulta paralela de acordo com a eficiência oferecida por cada um deles para cada consulta específica.

Quando uma consulta paralela filtra as linhas, e transforma e extrai valores das colunas, os dados são transmitidos de volta para o nó de cabeçalho como tuplas em vez de páginas de dados. Portanto, a execução de uma consulta paralela não adiciona páginas ao grupo de buffers ou remove páginas que já estão no grupo de buffers.

O Aurora verifica o número de páginas de dados da tabela que estão presentes no grupo de buffers e a proporção dos dados da tabela que esse número representa. O Aurora usa essas informações

para determinar se é mais eficiente usar a consulta paralela (e ignorar os dados presentes no grupo de buffers). Como alternativa, o Aurora pode usar o caminho de processamento de consulta não paralela, que usa os dados armazenados em cache no grupo de buffers. Quais páginas são armazenadas em cache e como as consultas com muitos dados afetam o armazenamento em cache e a remoção vai depender das configurações relacionadas ao grupo de buffers. Portanto, pode ser difícil prever se uma consulta específica vai usar a consulta paralela, pois a escolha depende de dados que estão em constante mudança dentro do grupo de buffers.

Além disso, o Aurora impõe limites de simultaneidade para as consultas paralelas. Como nem toda consulta usa a consulta paralela, em geral as tabelas que são acessadas por várias consultas simultaneamente têm uma porção substancial de seus dados no grupo de buffers. Portanto, o Aurora muitas vezes não escolhe essas tabelas para a consulta paralela.

Quando você executa uma sequência de consultas não paralelas na mesma tabela, a primeira consulta pode ficar lenta devido à ausência de dados do grupo de buffers. Em seguida, a segunda consulta, bem como as subsequentes, são muito mais rápidas porque o grupo de buffers agora está "aquecido". As consultas paralelas normalmente apresentam uma performance consistente desde a primeira consulta em uma tabela. Ao conduzir testes de performance, compare as consultas não paralelas tanto com um grupo de buffers ocioso quanto com um aquecido. Em alguns casos, os resultados com um grupo de buffers aquecido pode ser comparável aos tempos obtidos com a consulta paralela. Nesses casos, considere fatores como a frequência de consultas em relação a essa tabela. Considere também se vale a pena manter os dados dessa tabela no grupo de buffer.

O cache de consultas evita a que a consulta seja executada novamente quando uma consulta idêntica é enviada e a tabela subjacente não foi alterada. As consultas otimizadas pelo recurso de consulta paralela podem acessar o cache de consulta, tornando-as efetivamente instantâneas quando executadas novamente.

> **Note**
>
> Ao conduzir comparações de performance, o cache de consulta pode apresentar tempos artificialmente baixos. Portanto, em situações de comparação você pode usar o aviso `sql_no_cache`. Esse aviso evita que o resultado seja obtido do cache da consulta, ainda que essa mesma consulta tenha sido executada anteriormente. O aviso vem imediatamente após a instrução `SELECT` em uma consulta. Muitos exemplos de consultas paralelas neste tópico incluem essa dica, a fim de tornar os tempos de consulta comparáveis entre as versões da consulta para as quais a consulta paralela está habilitada ou desabilitada.

Lembre-se de remover esse aviso de seu código-fonte ao implantar o uso de consulta paralela na produção.

Dicas do Optimizer

Outra forma de controlar o Optimizer é usar dicas do Optimizer, que podem ser especificadas em instruções individuais. Por exemplo, é possível ativar a otimização para uma tabela em uma instrução e, depois, desativá-la para outra tabela. Para ter mais informações sobre essas dicas, consulte Dicas do Optimizer no Guia de referência do MySQL.

É possível usar dicas SQL com consultas do Aurora MySQL para ajustar a performance. Também é possível usar dicas para impedir que planos de execução para consultas importantes sejam alterados devido a condições imprevisíveis.

Estendemos o recurso de dicas de SQL para ajudar você a controlar as opções do Optimizer para seus planos de consulta. Essas dicas se aplicam a consultas que usam a otimização de consultas paralelas. Para ter mais informações, consulte Dicas do Aurora MySQL.

Tabelas temporárias MyISAM

A otimização de consulta paralela se aplica somente a tabelas InnoDB. Como o Aurora MySQL usa o MyISAM nos bastidores para tabelas temporárias, as fases de consulta internas que envolvem tabelas temporárias nunca usam a consulta paralela. Essas fases da consulta são indicadas por `Using temporary` na saída de `EXPLAIN`.

Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL

Você pode usar o recurso de Auditoria avançada de alta performance no Amazon Aurora MySQL para realizar a auditoria da atividade do banco de dados. Para isso, habilite a coleta de logs de auditoria definindo vários parâmetros do cluster de banco de dados. Com a Auditoria avançada habilitada, você pode usá-la para registrar qualquer combinação de eventos compatíveis.

Você pode visualizar ou baixar os logs de auditoria para revisar as informações de auditoria de uma instância de banco de dados de cada vez. Para fazer isso, você pode usar os procedimentos no Monitorar arquivos de log do Amazon Aurora.

> **Tip**
>
> Para um cluster de banco de dados do Aurora contendo várias instâncias de banco de dados, talvez seja mais conveniente examinar os logs de auditoria de todas as instâncias no cluster. Para fazer isso, você pode usar o CloudWatch Logs. Você pode ativar uma configuração no nível do cluster para publicar os dados do log de auditoria do Aurora MySQL em um grupo de logs no CloudWatch. Em seguida, você pode visualizar, filtrar e pesquisar os logs de auditoria por meio da interface do CloudWatch. Para obter mais informações, consultePublicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs

Como habilitar a Auditoria avançada

Use os parâmetros descritos nesta seção para habilitar e configurar a Auditoria avançada para seu cluster de banco de dados.

Use o parâmetro `server_audit_logging` para habilitar ou desabilitar auditoria avançada.

Usar o parâmetro `server_audit_events` para especificar quais eventos devem ser registrados em log.

Use os parâmetros `server_audit_incl_users` e `server_audit_excl_users` para especificar quem será auditado. Por padrão, todos os usuários são auditados. Para obter detalhes sobre como esses parâmetros funcionam quando um ou ambos são deixados em branco, ou os mesmos nomes de usuário são especificados em ambos, consulte server_audit_incl_users e server_audit_excl_users.

Configure a Auditoria avançada, definindo esses parâmetros no parameter group usado por seu cluster de banco de dados. Você pode usar o procedimento mostrado em Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora para modificar os parâmetros de cluster de banco de dados usando o Console de gerenciamento da AWS. Você pode usar o comando da AWS CLI modify-db-cluster-parameter-group ou o comando da API do Amazon RDS ModifyDBClusterParameterGroup para modificar parâmetros de cluster de banco de dados de forma programática.

A modificação desses parâmetros não requer que o cluster do banco de dados seja reiniciado quando o grupo de parâmetros já está associado ao cluster. Quando você associa o grupo de parâmetros ao cluster pela primeira vez, é necessária uma reinicialização do cluster.

Tópicos

- server_audit_logging
- server_audit_events
- server_audit_incl_users
- server_audit_excl_users

server_audit_logging

Ativa ou desativa a Auditoria avançada. Este parâmetro é OFF (Desligado) por padrão. Ajuste-o para ON (Ligado) para ativar a auditoria avançada.

Nenhum dado de auditoria aparece nos logs, a menos que você também defina um ou mais tipos de eventos para auditoria usando o parâmetro `server_audit_events`.

Para confirmar se os dados de auditoria são registrados em log para uma instância de banco de dados, verifique se alguns arquivos de log dessa instância têm nomes do formulário `audit/audit.log.`*`other_identifying_information`*. Para ver os nomes dos arquivos de logs, siga o procedimento no Como visualizar e listar arquivos de log do banco de dados.

server_audit_events

Contém a lista de eventos delimitada por vírgulas a registrar. Os eventos devem ser especificados em letras maiúsculas, e não pode haver espaço em branco entre os elementos da lista, por exemplo: `CONNECT,QUERY_DDL`. O parâmetro padrão é uma string vazia.

Você pode registrar qualquer combinação dos seguintes eventos:

- CONNECT – Registra conexões bem-sucedidas e falhas, além de desconexões. Este evento inclui informações do usuário.
- QUERY – Registra todas as consultas em texto simples, incluindo aquelas que falham devido a sintaxe ou erros de permissão.

> **Tip**
>
> Com esse tipo de evento ativado, os dados de auditoria incluem informações sobre o monitoramento contínuo e as informações de verificação de integridade que o Aurora faz automaticamente. Se você estiver interessado apenas em determinados tipos de operações, poderá usar os tipos mais específicos de eventos. Você também pode usar a interface do CloudWatch para pesquisar nos logs eventos relacionados a bancos de dados, tabelas ou usuários específicos.

- QUERY_DCL – Semelhante ao evento de QUERY, mas retorna apenas as consultas de linguagem de controle de dados (DCL) (GRANT, REVOKE, etc.).
- QUERY_DDL – Semelhante ao evento de QUERY, mas retorna apenas as consultas de linguagem de definição de dados (DDL) (CREATE, ALTER, etc.).
- QUERY_DML – Semelhante ao evento de QUERY, mas retorna apenas as consultas de linguagem de manipulação de dados (DML) (INSERT, UPDATE, etc. e também SELECT).
- TABLE – Registra as tabelas que foram afetadas pela execução da consulta.

> **Note**
>
> Não há nenhum filtro no Aurora que exclua determinadas consultas dos logs de auditoria. Para excluir consultas `SELECT`, exclua todas as instruções DML.
> Se determinado usuário estiver relatando essas consultas `SELECT` internas nos logs de auditoria, você poderá excluí-lo definindo o parâmetro do cluster de banco de dados server_audit_excl_users. Entretanto, se esse usuário também for usado em outras atividades e não puder ser omitido, não haverá outra opção para excluir consultas `SELECT`.

server_audit_incl_users

Contém a lista de nomes de usuário delimitada por vírgulas para os usuários cuja atividade foi registrada. Não pode haver espaços em branco entre os elementos da lista, por exemplo:

`user_3,user_4`. O parâmetro padrão é uma string vazia. O tamanho máximo é de 1024 caracteres. Os nomes de usuários especificados devem corresponder aos valores correspondentes na coluna `User` da tabela `mysql.user`. Para obter mais informações sobre nomes de usuário, consulte Nomes de usuário e senhas de contas na documentação do MySQL.

Se `server_audit_incl_users` e `server_audit_excl_users` estiverem vazios (por padrão), todos os usuários serão auditados.

Se você adicionar usuários a `server_audit_incl_users` e deixar `server_audit_excl_users` vazio, somente esses usuários serão auditados.

Se você adicionar usuários a `server_audit_excl_users` e deixar `server_audit_incl_users` vazio, somente esses usuários serão auditados, exceto os listados em `server_audit_excl_users`.

Se você adicionar os mesmos usuários a `server_audit_excl_users` e `server_audit_incl_users`, esses usuários serão auditados. Quando o mesmo usuário estiver listado em ambas as configurações, é dada prioridade mais alta a `server_audit_incl_users`.

Os eventos de conexão e de desconexão não são afetados por esta variável; eles serão sempre registrados, se assim for especificado. O usuário é registrado em log, mesmo que ele já esteja especificado no parâmetro `server_audit_excl_users`, pois `server_audit_incl_users` tem uma prioridade maior.

server_audit_excl_users

Contém a lista de nomes de usuário delimitada por vírgulas para os usuários cuja atividade não foi registrada. Não pode haver espaços em branco entre os elementos da lista, por exemplo: `rdsadmin,user_1,user_2`. O parâmetro padrão é uma string vazia. O tamanho máximo é de 1024 caracteres. Os nomes de usuários especificados devem corresponder aos valores correspondentes na coluna `User` da tabela `mysql.user`. Para obter mais informações sobre nomes de usuário, consulte Nomes de usuário e senhas de contas na documentação do MySQL.

Se `server_audit_incl_users` e `server_audit_excl_users` estiverem vazios (por padrão), todos os usuários serão auditados.

Se você adicionar usuários a `server_audit_excl_users` e deixar `server_audit_incl_users` vazio, somente os usuários listados em `server_audit_excl_users` não serão auditados, enquanto todos os outros usuários serão.

Se você adicionar os mesmos usuários a `server_audit_excl_users` e `server_audit_incl_users`, esses usuários serão auditados. Quando o mesmo usuário estiver listado em ambas as configurações, é dada prioridade mais alta a `server_audit_incl_users`.

Os eventos de conexão e de desconexão não são afetados por esta variável; eles serão sempre registrados, se assim for especificado. Um usuário é registrado caso também esteja especificado no parâmetro `server_audit_incl_users`, porque essa configuração tem prioridade maior do que `server_audit_excl_users`.

Visualizar logs de auditoria

Você pode visualizar e baixar os logs de auditoria usando o console. Na página Databases (Bancos de dados), escolha a instância de banco de dados para exibir seus detalhes e role até a seção Logs. Os logs de auditoria produzidos pelo recurso de auditoria avançada têm nomes do formulário `audit/audit.log.`*`other_identifying_information`*.

Para baixar um arquivo de log, localize o arquivo na seção Logs e escolha Download (Fazer download).

Você também pode obter uma lista dos arquivos de log usando o comando describe-db-log-files da AWS CLI. Você pode baixar o conteúdo de um arquivo de log usando o comando download-db-log-file-portion da AWS CLI. Para obter mais informações, consulte Como visualizar e listar arquivos de log do banco de dados e Como baixar um arquivo de log de banco de dados.

Detalhes do log de auditoria

Os arquivos de log são representados como arquivos de variáveis separadas por vírgula (CSV) no formato UTF-8. As consultas também são colocadas entre aspas simples (').

O log de auditoria é armazenado separadamente no armazenamento local de cada instância do Aurora MySQL. Cada instância distribui gravações em quatro arquivos de log por vez. O tamanho máximo de um arquivo de log é de 100 MB. Quando esse limite não configurável é atingido, o Aurora troca os arquivos e gera um novo arquivo.

> **Tip**
>
> As entradas do log não estão em ordem sequencial. Para ordenar as entradas, use o valor do carimbo de data/hora. Para ver os eventos mais recentes, talvez seja preciso analisar todos os arquivos de log. Para obter mais flexibilidade na classificação e pesquisa dos dados

de log, ative a configuração para carregar os logs de auditoria no CloudWatch e visualizá-los usando a interface do CloudWatch.
Para visualizar dados de auditoria com mais tipos de campos e com saída no formato JSON, você também pode usar o recurso Database Activity Streams. Para obter mais informações, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams

Os arquivos de log de auditoria incluem as seguintes informações delimitadas por vírgulas nas linhas, na ordem especificada:

Campo	Descrição
timestamp	O time stamp do Unix para o evento registrado com precisão de microssegundo.
serverhost	O nome da instância em que o evento foi registrado.
username	O nome de usuário conectado.
host	O host no qual o usuário está conectado.
connectionid	O número de ID da conexão para a operação registrada.
queryid	O número do ID de consulta, que pode ser usado para encontrar os eventos relacionais da tabela e consultas relacionadas. Para eventos `TABLE`, são adicionadas várias linhas.
operação	O tipo de ação gravado. Os valores possíveis são: `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME` e `DROP`.
banco de dados	O banco de dados ativo, conforme definido pelo comando `USE`.
objeto	Para eventos de `QUERY`, esse valor indica a consulta executada pelo banco de dados. Para eventos de `TABLE`, indica o nome da tabela.
retcode	O código de retorno da operação registrada.

Replicação com o Amazon Aurora MySQL

Os recursos de replicação do Aurora MySQL são fundamentais para a alta disponibilidade e a performance do cluster. O Aurora facilita criar ou redimensionar clusters com até 15 réplicas do Aurora.

Todas as réplicas funcionam pelos mesmos dados subjacentes. Se algumas instâncias de banco de dados ficarem offline, outras permanecerão disponíveis para continuar o processamento de consultas ou para assumir a gravação, caso necessário. O Aurora distribui automaticamente as conexões somente leitura entre várias instâncias de banco de dados, ajudando o cluster do Aurora a dar suporte a workloads que exigem muitas consultas.

Nos tópicos a seguir, é possível encontrar informações sobre como a replicação do Aurora MySQL funciona e como ajustar as configurações de replicação tendo em vista disponibilidade e performance melhores.

Tópicos

- Usar réplicas do Aurora
- Opções de replicação para Amazon Aurora MySQL
- Considerações sobre performance da replicação do Amazon Aurora MySQL
- Zero-downtime restart (ZDR – Reinício com tempo de inatividade zero) para Amazon Aurora MySQL
- Configurar filtros de replicação com o Aurora MySQL
- Monitorar a replicação do Amazon Aurora MySQL
- Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS
- Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário)
- Usar a replicação baseada em GTID

Usar réplicas do Aurora

As réplicas do Aurora são endpoints independentes em um cluster de banco de dados do Aurora, cuja melhor utilidade é dimensionar operações de leitura e aumentar a disponibilidade. É possível distribuir até 15 réplicas do Aurora entre as zonas de disponibilidade que um cluster de banco de

dados abrange em uma Região da AWS. Embora o volume do cluster de banco de dados seja composto de várias cópias dos dados para o cluster de banco de dados, os dados no volume do cluster são representados como um único volume lógico para a instância primária e para as réplicas do Aurora no cluster de banco de dados. Para obter mais informações sobre réplicas do Aurora, consulte Réplicas do Aurora.

As réplicas do Aurora funcionam bem para a escalabilidade de leitura porque são totalmente dedicadas a operações de leitura no seu volume de cluster. As operações de gravação são gerenciadas pela instância principal. Como o volume do cluster é compartilhado entre todas as instâncias no seu cluster de banco de dados do Aurora MySQL, nenhum trabalho adicional é necessário para replicar uma cópia dos dados para cada réplica do Aurora. Por outro lado, as réplicas de leitura do MySQL devem reproduzir, em um único thread, todas as operações de gravação da instância de banco de dados de origem no armazenamento de dados local. Essa limitação pode afetar a possibilidade de réplicas de leitura do MySQL de dar suporte a grandes volumes de tráfego de leitura.

Com o Aurora MySQL, quando a réplica do Aurora é excluída, o endpoint da instância é removido imediatamente, e a réplica do Aurora é removida do endpoint leitor. Se houver instruções em execução na réplica do Aurora que está sendo excluída, haverá um período de carência de três minutos. As instruções existentes podem ser concluídas durante o período de carência. Quando o período de carência termina, a réplica do Aurora é fechada e excluída.

Important

As réplicas do Aurora para o Aurora MySQL usam sempre o nível de isolamento de transação padrão `REPEATABLE READ` para operações nas tabelas do InnoDB. Você pode usar o comando `SET TRANSACTION ISOLATION LEVEL` para alterar o nível de transação somente para a instância primária de um cluster de banco de dados do Aurora MySQL. Essa restrição evita bloqueios no nível do usuário nas réplicas do Aurora e permite que elas escalem para oferecer suporte a milhares de conexões de usuários ativos ainda que com um mínimo de atraso na réplica.

Note

As instruções DDL executadas na instância principal podem interromper conexões de banco de dados nas réplicas do Aurora associadas. Se uma conexão de réplica do Aurora estiver usando um objeto de banco de dados ativamente (por exemplo, uma tabela) e esse objeto for

modificado na instância principal usando uma instrução DDL, a conexão de réplica do Aurora será interrompida.

> **Note**
>
> A região China (Ningxia) não oferece suporte a réplicas de leitura entre regiões.

Opções de replicação para Amazon Aurora MySQL

Você pode configurar a replicação entre qualquer uma das seguintes opções:

- Dois clusters de banco de dados do Aurora MySQL em diferentes Regiões da AWS, criando uma réplica de leitura entre regiões de um cluster de banco de dados do Aurora MySQL.

  Para obter mais informações, consulte Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS.
- Dois clusters de banco de dados do Aurora MySQL na mesma Região da AWS usando a replicação do log binário (binlog) do MySQL.

  Para obter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).
- Uma instância de banco de dados RDS para MySQL como a fonte de um cluster de banco de dados do Aurora MySQL, criando uma réplica de leitura do Aurora de uma instância de banco de dados RDS para MySQL.

  Você pode usar essa abordagem para trazer mudanças de dados existentes e contínuas Aurora MySQL durante a migração para Aurora. Para ter mais informações, consulte Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora.

  Você também pode usar essa abordagem para aumentar a escalabilidade das consultas de leitura para seus dados. É possível consultar os dados usando uma ou mais instâncias de banco de dados dentro de um cluster de Aurora MySQL somente leitura. Para obter mais informações, consulte Escalar leituras para o banco de dados MySQL com o Amazon Aurora.

- Um cluster de banco de dados do Aurora MySQL em uma Região da AWS e até cinco clusters de banco de dados do Aurora somente leitura Aurora MySQL em diferentes regiões, criando um banco de dados global do Aurora.

  Você pode usar um banco de dados global do Aurora para comportar aplicações presentes ao redor do mundo. O cluster de banco de dados do Aurora MySQL primário tem uma instância gravador e até 15 réplicas do Aurora. Os clusters de banco de dados secundários do Aurora MySQL somente leitura podem ser compostos de até 16 réplicas de Aurora. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.

> **Note**
>
> Reinicializar a instância primária de um cluster de banco de dados do Amazon Aurora também reinicia automaticamente as réplicas do Aurora desse cluster de banco de dados para restabelecer um ponto de entrada que garante a consistência de leitura/gravação em todo o cluster de banco de dados.

Considerações sobre performance da replicação do Amazon Aurora MySQL

Os recursos a seguir ajudam você a ajustar a performance da replicação do Aurora MySQL.

O recurso de compactação do log de réplicas reduz a largura de banda da rede para mensagens de replicação. Como cada mensagem é transmitida para todas as réplicas do Aurora, os benefícios são maiores para cluster maiores. Esse recurso envolve certa sobrecarga da CPU no nó gravador para realizar a compactação. Ele está sempre habilitado no Aurora MySQL versão 2 e versão 3.

O recurso de filtragem de logs binários reduz a largura de banda da rede para mensagens de replicação. Como as réplicas do Aurora não usam as informações de log binário incluídas nas mensagens de replicação, esses dados são omitidos das mensagens enviadas para esses nós.

No Aurora MySQL versão 2, você pode controlar esse recurso alterando o valor do parâmetro `aurora_enable_repl_bin_log_filtering`. Por padrão, esse parâmetro está ativado. Como essa otimização destina-se a ser transparente, você pode desativar essa configuração somente durante operações de diagnóstico ou solução de problemas relacionadas a replicação. Por exemplo, você pode fazer isso para corresponder o comportamento de um cluster do Aurora MySQL mais antigo no qual esse recurso não estava disponível.

A filtragem do log binário está sempre habilitada no Aurora MySQL versão 3.

Zero-downtime restart (ZDR – Reinício com tempo de inatividade zero) para Amazon Aurora MySQL

O recurso Zero-downtime restart (ZDR – Reinício com tempo de inatividade zero) pode preservar algumas ou todas as conexões ativas com instâncias de banco de dados durante determinados tipos de reinicializações. O ZDR se aplica a reinicializações que o Aurora executa automaticamente para resolver condições de erro, por exemplo, quando uma réplica começa a ficar muito atrasada em relação à origem.

> **Important**
>
> O mecanismo ZDR opera com base no melhor esforço. As versões, as classes de instância, as condições de erro, operações SQL compatíveis e outros fatores do Aurora MySQL que determinam onde o ZDR se aplica estão sujeitos a alterações a qualquer momento.

O ZDR para Aurora MySQL 2.x requer a versão 2.10 e posterior. O ZDR está disponível em todas as versões secundárias do Aurora MySQL 3.x. No Aurora MySQL versões 2 e 3, o mecanismo ZDR está habilitado por padrão, e o Aurora não usa o parâmetro `aurora_enable_zdr`.

Na página Events (Eventos), o Aurora informa atividades relacionadas ao reinício do tempo de inatividade zero. O Aurora registra um evento quando tenta reiniciar usando o mecanismo ZDR. Esse evento indica por que o Aurora executa a reinicialização. Em seguida, o Aurora registra outro evento quando a reinicialização for concluída. Esse evento final relata quanto tempo o processo demorou e quantas conexões foram preservadas ou descartadas durante a reinicialização. Você pode consultar o log de erros do banco de dados para ver mais detalhes sobre o que aconteceu durante a reinicialização.

Embora as conexões permaneçam intactas após uma operação ZDR bem-sucedida, algumas variáveis e recursos são reinicializados. Os seguintes tipos de informações não são preservados por meio de uma reinicialização causada pela reinicialização do tempo de inatividade zero:

- Variáveis globais. O Aurora restaura variáveis de sessão, mas não restaura variáveis globais após a reinicialização.
- Variáveis de status. Em particular, o valor do tempo de atividade informado pelo status do mecanismo é redefinido.

- `LAST_INSERT_ID`.
- Estado de `auto_increment` na memória para tabelas. O estado de incremento automático na memória é reinicializado. Para ter mais informações sobre valores de incremento automático, consulte o Guia de referência do MySQL.
- Informações diagnósticas das tabelas `INFORMATION_SCHEMA` e `PERFORMANCE_SCHEMA`. Essas informações de diagnóstico também aparecem na saída de comandos como `SHOW PROFILE` e `SHOW PROFILES`.

A tabela a seguir mostra as versões, os perfis de instância e outras circunstâncias que determinam se o Aurora pode ou não usar o mecanismo ZDR ao reiniciar instâncias de banco de dados no cluster.

Aurora MySQL versão	O ZDR se aplica ao gravador?	O ZDR se aplica aos leitores?	O ZDR está sempre habilitado?	Observações
2.x, anterior a 2.10.0	Não	Não	N/D	O ZDR não está disponível para essas versões.
2.10.0 a 2.11.0	Sim	Sim	Sim	O Aurora reverte todas as transações que estão em andamento em conexões ativas. Sua aplicação deve tentar executar as transações novamente. O Aurora cancela todas as conexões que usam TLS/SSL, tabelas temporárias, bloqueios de tabela ou bloqueios de usuário.
2.11.1 e posterior	Sim	Sim	Sim	O Aurora reverte todas as transações que estão em andamento em conexões ativas. Sua aplicação deve tentar executar as transações novamente.

Aurora MySQL versão	O ZDR se aplica ao gravador?	O ZDR se aplica aos leitores?	O ZDR está sempre habilitado?	Observações
				O Aurora cancela todas as conexões que usam tabelas temporárias, bloqueios de tabela ou bloqueios de usuário.
3.01 a 3.03	Sim	Sim	Sim	O Aurora reverte todas as transações que estão em andamento em conexões ativas. Sua aplicação deve tentar executar as transações novamente. O Aurora cancela todas as conexões que usam TLS/SSL, tabelas temporárias, bloqueios de tabela ou bloqueios de usuário.
3.04 e posterior	Sim	Sim	Sim	O Aurora reverte todas as transações que estão em andamento em conexões ativas. Sua aplicação deve tentar executar as transações novamente. O Aurora cancela todas as conexões que usam tabelas temporárias, bloqueios de tabela ou bloqueios de usuário.

Configurar filtros de replicação com o Aurora MySQL

Você pode usar filtros de replicação para especificar quais bancos de dados e tabelas são replicados com uma réplica de leitura. Os filtros de replicação podem incluir bancos de dados e tabelas na replicação ou excluí-los da replicação.

Veja a seguir alguns casos de uso para filtros de replicação:

- Para reduzir o tamanho de uma réplica de leitura. Com a filtragem de replicação, você pode excluir os bancos de dados e tabelas que não são necessários na réplica de leitura.

- Para excluir bancos de dados e tabelas de réplicas de leitura por motivos de segurança.
- Para replicar diferentes bancos de dados e tabelas para casos de uso específicos em diferentes réplicas de leitura. Por exemplo, você pode usar réplicas de leitura específicas para análise ou fragmentação.
- Para um cluster de banco de dados que tenha réplicas de leitura em diferentes Regiões da AWS, para replicar diferentes bancos de dados ou tabelas em diferentes Regiões da AWS.
- Para especificar quais bancos de dados e tabelas serão replicados com um cluster de banco de dados do Aurora MySQL configurado como uma réplica em uma topologia de replicação de entrada. Para obter mais informações sobre essa configuração, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Tópicos

- Configurar parâmetros de filtragem de replicação para o Aurora MySQL
- Limitações de filtragem de replicação para o Aurora MySQL
- Exemplos de filtragem de replicação para o Aurora MySQL
- Visualizar os filtros de replicação para uma réplica de leitura

Configurar parâmetros de filtragem de replicação para o Aurora MySQL

Para configurar filtros de replicação, defina os seguintes parâmetros:

- `binlog-do-db`: replicar alterações para os logs binários especificados. Ao definir esse parâmetro para um cluster de origem do log binário, somente os logs binários especificados no parâmetro são replicados.
- `binlog-ignore-db`: não replicar alterações para os logs binários especificados. Quando o parâmetro `binlog-do-db` é definido para um cluster de origem do log binário, esse parâmetro não é avaliado.
- `replicate-do-db` – Replicar alterações nos bancos de dados especificados. Ao definir esse parâmetro para um cluster de réplica do log binário, somente os bancos de dados especificados no parâmetro são replicados.
- `replicate-ignore-db` – Não replique as alterações nos bancos de dados especificados. Quando o parâmetro `replicate-do-db` é definido para um cluster de réplica do log binário, esse parâmetro não é avaliado.

- `replicate-do-table` – Replicar alterações nas tabelas especificadas. Quando você define esse parâmetro para uma réplica de leitura, somente as tabelas especificadas no parâmetro são replicadas. Além disso, quando o parâmetro `replicate-do-db` ou `replicate-ignore-db` estiver definido, inclua o banco de dados que contém as tabelas especificadas na replicação com o cluster de réplica do log binário.
- `replicate-ignore-table` – Não replique as alterações nas tabelas especificadas. Quando o parâmetro `replicate-do-table` é definido para um cluster de réplica do log binário, esse parâmetro não é avaliado.
- `replicate-wild-do-table` – Replicar tabelas com base nos padrões de nome de banco de dados e tabela especificados. Os caracteres curinga `%` e `_` são compatíveis. Quando o parâmetro `replicate-do-db` ou `replicate-ignore-db` estiver definido, inclua o banco de dados que contém as tabelas especificadas na replicação com o cluster de réplica do log binário.
- `replicate-wild-ignore-table` – Não replique tabelas com base nos padrões de nome de banco de dados e tabela especificados. Os caracteres curinga `%` e `_` são compatíveis. Quando o parâmetro `replicate-do-table` ou `replicate-wild-do-table` é definido para um cluster de réplica do log binário, esse parâmetro não é avaliado.

Os parâmetros são avaliados na ordem em que estão listados. Para obter mais informações sobre como esses parâmetros funcionam, consulte a documentação do MySQL:

- Para obter informações gerais, consulte Opções e variáveis do servidor de réplica.
- Para obter informações sobre como os parâmetros de filtragem de replicação de banco de dados são avaliados, consulte Avaliação de opções de replicação em nível de banco de dados e log binário.
- Para obter informações sobre como os parâmetros de filtragem de replicação de tabela são avaliados, consulte Avaliação de opções de replicação em nível de tabela.

Por padrão, cada um desses parâmetros tem um valor vazio. Em cada cluster de log binário, é possível usar esses parâmetros para definir, alterar e excluir filtros de replicação. Quando você define um desses parâmetros, separe cada filtro dos outros com uma vírgula.

Você pode usar `%` os caracteres curinga `_` e nos parâmetros `replicate-wild-do-table` e `replicate-wild-ignore-table`. O curinga `%` corresponde a qualquer número de caracteres e o caractere curinga `_` corresponde apenas a um caractere.

O formato de log binário da instância de banco de dados de origem é importante para replicação porque determina o registro de alterações de dados. A configuração do parâmetro `binlog_format` determina se a replicação é baseada em linha ou baseada em declaração. Para obter mais informações, consulte Configurar o registro em log binário do Aurora MySQL para bancos de dados single-AZ.

> **Note**
>
> Todas as instruções DDL (Data Definition Language, linguagem de definição de dados) são replicadas como instruções, independentemente da `binlog_format` configuração na instância de banco de dados de origem.

Limitações de filtragem de replicação para o Aurora MySQL

As seguintes limitações se aplicam à filtragem de replicação para o Aurora MySQL:

- Os filtros de replicação são compatíveis somente com o Aurora MySQL versão 3.
- Cada parâmetro de filtragem de replicação tem um limite de 2.000 caracteres.
- As vírgulas não são compatíveis em filtros de replicação.
- A filtragem de replicação não suporta transações XA.

  Para obter mais informações, consulte Restrictions on XA Transactions na documentação do MySQL.

Exemplos de filtragem de replicação para o Aurora MySQL

Para configurar a filtragem de replicação para uma réplica de leitura, modifique os parâmetros de filtragem de replicação no grupo de parâmetros do cluster de banco de dados associado à réplica de leitura.

> **Note**
>
> Não é possível modificar um grupo de parâmetros de cluster de banco de dados padrão. Se a réplica de leitura estiver usando um grupo de parâmetros padrão, crie um novo grupo de parâmetros e o associe à instância de banco de dados. Para obter mais informações sobre

os grupos de parâmetros de cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Você pode definir parâmetros em um grupo de parâmetros de cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Para obter informações sobre como configurar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora. Quando você define parâmetros em um grupo de parâmetros de cluster de banco de dados, todos os clusters de banco de dados associados ao grupo de parâmetros usam as configurações de parâmetros. Se você definir os parâmetros de filtragem de replicação em um grupo de parâmetros de cluster de banco de dados, verifique se o grupo de parâmetros está associado apenas a clusters de réplica de leitura. Deixe os parâmetros de filtragem de replicação vazios para instâncias de banco de dados de origem.

Os exemplos a seguir definem os parâmetros usando o AWS CLI. Estes exemplos definem `ApplyMethod` para `immediate` de modo que as mudanças do parâmetro ocorram imediatamente depois que o comando CLI termina. Se você quiser que uma alteração pendente seja aplicada depois que a réplica de leitura for reinicializada, defina como `ApplyMethod` `pending-reboot`.

Os exemplos a seguir definem filtros de replicação:

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example Incluir bancos de dados em replicação

O exemplo a seguir inclui os bancos de dados `mydb1` e `mydb2` na replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
```

```
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example Incluir tabelas na replicação

O exemplo a seguir inclui as tabelas `table1` e `table2` no banco de dados `mydb1` na replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example Incluir tabelas na replicação usando caracteres curinga

O exemplo a seguir inclui tabelas com nomes que começam com `order` e `return` no banco de dados `mydb` na replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example Excluir bancos de dados da replicação

O exemplo a seguir exclui os bancos de dados `mydb5` e `mydb6` da replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6,ApplyMethod=immediate"
```

Example Excluir tabelas da replicação

O exemplo a seguir exclui a tabela `table1` no banco de dados `mydb5` e a tabela `table2` no banco de dados `mydb6` da replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
```

```
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example Excluir tabelas da replicação usando caracteres curinga

O exemplo a seguir exclui tabelas com nomes que começam com `order` e `return` no banco de dados `mydb7` da replicação.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Visualizar os filtros de replicação para uma réplica de leitura

Você pode visualizar os filtros de replicação de uma réplica de leitura das seguintes maneiras:

- Verifique as configurações dos parâmetros de filtragem de replicação no grupo de parâmetros associado à réplica de leitura.

  Para obter instruções, consulte Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados no Amazon Aurora.
- Em um cliente MySQL, conecte-se à réplica de leitura e execute a instrução `SHOW REPLICA STATUS`.

  Na saída, os campos a seguir mostram os filtros de replicação para a réplica de leitura:
  - `Binlog_Do_DB`
  - `Binlog_Ignore_DB`
  - `Replicate_Do_DB`

- `Replicate_Ignore_DB`
- `Replicate_Do_Table`
- `Replicate_Ignore_Table`
- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

Para obter mais informações sobre esses campos, consulte Verificar o status da replicação na documentação do MySQL.

Monitorar a replicação do Amazon Aurora MySQL

A escalabilidade de leitura e a alta disponibilidade dependem de um tempo de atraso mínimo. É possível monitorar até que ponto uma réplica do Aurora está atrasada em relação à instância primária do cluster de banco de dados do Aurora MySQL, monitorando a métrica `AuroraReplicaLag` do Amazon CloudWatch. A métrica `AuroraReplicaLag` é registrada em cada réplica do Aurora.

A instância de banco de dados principal também registra as métricas `AuroraReplicaLagMaximum` e `AuroraReplicaLagMinimum` do Amazon CloudWatch. A métrica `AuroraReplicaLagMaximum` registra a quantidade máxima de atraso entre a instância de banco de dados principal e cada réplica do Aurora no cluster de banco de dados. A métrica `AuroraReplicaLagMinimum` registra a quantidade mínima de atraso entre a instância de banco de dados principal e cada réplica do Aurora no cluster de banco de dados.

Se você precisar do valor mais atual para o atraso da réplica do Aurora, confira a métrica `AuroraReplicaLag` no Amazon CloudWatch. O atraso da réplica do Aurora também é registrado em cada réplica do Aurora do cluster de banco de dados do Aurora MySQL na tabela `information_schema.replica_host_status`. Para obter mais informações sobre essa tabela, consulte information_schema.replica_host_status.

Para obter mais informações sobre como monitorar as instâncias do RDS e as métricas do CloudWatch, consulte Monitorar métricas em um cluster do Amazon Aurora.

Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS

Você pode criar um cluster de banco de dados do Amazon Aurora MySQL como uma réplica de leitura em uma Região da AWS diferente do cluster de banco de dados de origem. O uso dessa abordagem pode melhorar sua capacidade de recuperação de desastres, permite que você aumente as operações de leitura em uma Região da AWS mais próxima a seus usuários e facilita a migração de uma Região da AWS para outra.

Você pode criar réplicas de leitura de clusters de banco de dados criptografados e descriptografados. A réplica de leitura deverá ser criptografada se o cluster de banco de dados de origem estiver criptografado.

Para cada cluster de banco de dados de origem, você só pode ter até cinco clusters de banco de dados entre regiões que são réplicas de leitura.

> **Note**
>
> Como alternativa a réplicas de leitura entre regiões, é possível escalar as operações de leitura com o tempo de atraso mínimo utilizando um banco de dados global Aurora. Um Aurora Global Database tem um cluster de banco de dados primário do Aurora em uma Região da AWS e até dez clusters de banco de dados somente leitura secundários em diferentes regiões. Cada cluster de banco de dados secundário pode incluir até 16 réplicas (em vez de 15) de Aurora. A replicação do cluster de banco de dados primário para todos os secundários é tratada pela camada de armazenamento de Aurora e não pelo mecanismo de banco de dados. Portanto, o tempo de atraso para replicar as alterações é mínimo, geralmente menos de 1 segundo. Manter o mecanismo de banco de dados fora do processo de replicação significa que o mecanismo de banco de dados é dedicado ao processamento de workloads. Isso também significa que você não precisa configurar ou gerenciar a replicação de log binário do Aurora MySQL. Para saber mais, consulte Usar o Amazon Aurora Global Database.

Ao criar uma réplica de leitura de cluster de banco de dados do Aurora MySQL em outra Região da AWS, esteja ciente de que:

- Tanto o cluster de banco de dados de origem quanto o cluster de banco de dados da réplica de leitura entre regiões podem ter até 15 réplicas do Aurora em conjunto com a instância primária

para o cluster de banco de dados. Usando essa funcionalidade, é possível melhorar as operações de leitura para sua Região da AWS de origem e de sua Região da AWS de replicação de destino.

- Em um cenário entre regiões, há mais tempo de atraso entre o cluster de banco de dados de origem e a réplica de leitura devido aos canais de rede mais longos entre as Regiões da AWS.
- Os dados transferidos para a replicação entre regiões incorrem em cobranças de transferência de dados do Amazon RDS. As seguintes ações de replicação entre regiões geram cobranças referentes aos dados transferidos da Região da AWS de origem:
  - Quando você cria a réplica de leitura, o Amazon RDS gera um snapshot do cluster de origem e transfere esse snapshot para a Região da AWS que contém a réplica de leitura.
  - Para cada modificação de dados feita nos bancos de dados de origem, o Amazon RDS transfere os dados da região de origem para a Região da AWS que contém a réplica de leitura.

  Para obter mais informações sobre a definição de preço da transferência de dados do Amazon RDS, consulte Definição de preço do Amazon Aurora.
- Você pode executar várias ações simultâneas de criação ou exclusão para réplicas de leitura que referenciam o mesmo cluster de bando de dados de origem. No entanto, você deve permanecer dentro do limite de cinco réplicas de leitura para cada cluster de banco de dados de origem.
- Para que a replicação funcione efetivamente, cada réplica de leitura deve ter a mesma quantidade de recursos de computação e de armazenamento que o cluster de banco de dados de origem. Se você dimensionar o cluster de banco de dados de origem, também deverá dimensionar as réplicas de leitura.

Tópicos

- Antes de começar
- Criar um cluster de banco de dados de réplica de leitura entre regiões para o Aurora MySQL
- Promover uma réplica de leitura a um cluster de banco de dados para o Aurora MySQL
- Solucionar problemas de réplicas entre regiões do Amazon Aurora MySQL

Antes de começar

Para que você possa criar um cluster de banco de dados do Aurora MySQL que seja uma réplica de leitura entre regiões, precisa habilitar o registro em log binário no seu cluster de banco de dados do Aurora MySQL de origem. A replicação entre regiões do Aurora MySQL usa a replicação binária

do MySQL para reproduzir as mudanças no cluster de banco de dados da réplica de leitura entre regiões.

Para habilitar o registro em log binário em um cluster de banco de dados do Aurora MySQL, atualize o parâmetro `binlog_format` para o seu cluster de banco de dados de origem. O parâmetro `binlog_format` é um parâmetro em nível de cluster que se encontra no parameter group de cluster padrão. Se seu cluster de banco de dados usar o grupo de parâmetros do cluster de banco de dados padrão, crie um novo grupo de parâmetro do cluster de banco de dados para modificar as configurações do `binlog_format`. Recomendamos que você defina `binlog_format` como `MIXED`. No entanto, você também pode definir `binlog_format` como `ROW` ou `STATEMENT` se precisar de um formato específico de log binário. Reinicie seu cluster de banco de dados Aurora para que a alteração entre em vigor.

Para obter mais informações sobre o registro em log binário com o Aurora MySQL, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário). Para obter mais informações sobre como modificar parâmetros de configuração do Aurora MySQL, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora e Grupos de parâmetros para Amazon Aurora.

Criar um cluster de banco de dados de réplica de leitura entre regiões para o Aurora MySQL

Você pode criar um cluster de banco de dados do Aurora que seja uma réplica de leitura entre regiões usando o Console de gerenciamento da AWS , a AWS Command Line Interface (AWS CLI) ou a API do Amazon RDS. Você pode criar réplicas de leitura entre regiões de clusters de banco de dados criptografados e descriptografados.

Quando você cria uma réplica de leitura entre Regiões para o Aurora MySQL usando o Console de gerenciamento da AWS, o Amazon RDS cria um cluster de banco de dados na Região da AWS de destino e depois cria automaticamente uma instância de banco de dados que passa a ser a instância principal desse cluster de banco de dados.

Quando você cria uma réplica de leitura entre Regiões usando a AWS CLI ou a API do RDS, crie primeiro o cluster de banco de dados na Região da AWS de destino e aguarde até que fique ativo. Assim que estiver ativo, crie uma instância de banco de dados que seja a instância primária desse cluster de banco de dados.

A replicação começará quando a instância primária do cluster de banco de dados da réplica de leitura estiver disponível.

Use os seguintes procedimentos para criar uma réplica de leitura entre regiões a partir de um cluster de banco de dados do Aurora MySQL. Esses procedimentos funcionam para criar réplicas de leitura a partir de clusters de banco de dados criptografados ou descriptografados.

Console

Para criar um cluster de banco de dados do Aurora MySQL que seja uma réplica de leitura entre regiões com o Console de gerenciamento da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a Região da AWS que hospeda seu cluster de banco de dados de origem.
3. No painel de navegação, escolha Bancos de dados.
4. Escolha o cluster de banco de dados de origem para o qual você deseja criar uma réplica de leitura entre Regiões.
5. Para Actions (Ações), escolha Create cross region read replica (Criar réplica de leitura entre Regiões).
6. Na página Create cross region read replica (Criar réplica de leitura entre regiões), escolha as configurações das opções de seu cluster de banco de dados da réplica de leitura entre regiões, conforme descrito na tabela a seguir.

Opção	Descrição
Região de destino	Escolha a Região da AWS para hospedar o novo cluster de banco de dados da réplica de leitura entre Regiões.
Destination DB subnet group	Escolha o group de sub-redes de banco de dados para usar para o cluster de banco de dados da réplica de leitura entre regiões.
Acessível ao público	Escolha Yes (Sim) para dar um endereço IP público ao cluster de banco de dados da réplica de leitura entre regiões; do contrário, selecione No (Não).

Opção	Descrição
Criptografia	Selecione Enable Encryption (Habilitar criptografia) para habilitar a criptografia em repouso nesse cluster de banco de dados. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
AWS KMS key	Disponível apenas quando Encryption estiver definido como Enable Encryption. Escolha a AWS KMS key a ser usada para criptografar esse cluster de banco de dados. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
Classe de instância de banco de dados	Escolha uma classe de instância de banco de dados que defina os requisitos de processamento e memória para a instância principal no cluster de banco de dados. Para ter mais informações sobre as opções de classe de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.
implantação multi-AZ	Escolha Yes (Sim) para criar uma réplica de leitura do novo cluster de banco de dados em outra zona de disponibilidade na Região da AWS de destino para suporte de failover. Para obter mais informações sobre várias Zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.
Read replica source	Escolha o cluster de banco de dados de origem para criar uma réplica de leitura entre regiões.

Opção	Descrição
DB instance identifier	Digite um nome para a instância primária no seu cluster de banco de dados da réplica de leitura entre regiões. Esse identificador é usado no endereço do endpoint da instância principal do novo cluster de banco de dados. O DB instance identifier tem as seguintes restrições: • Deve conter de 1 a 63 caracteres alfanuméricos ou hífens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todas as instâncias de banco de dados para cada Conta da AWS por cada Região da AWS. Como o cluster de banco de dados da réplica de leitura entre regiões é criado a partir de um snapshot do cluster do banco de dados de origem, o nome de usuário mestre e a senha mestre da réplica de leitura são os mesmos nome e senha mestre do cluster do banco de dados de origem.

Opção	Descrição
Identificador do cluster de banco de dados	Insira um nome para o seu cluster de banco de dados da réplica de leitura entre regiões. Esse nome é exclusivo para sua conta de destino para a sua réplica. Esse identificador é usado no endereço do endpoint do cluster para o seu cluster de banco de dados. Para obter informações sobre o endpoint do cluster, consulte Conexões de endpoints do Amazon Aurora. O identificador do cluster de banco de dados tem as seguintes restrições: • Deve conter de 1 a 63 caracteres alfanuméricos ou hífens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todos os clusters de banco de dados para cada Conta da AWS por cada Região da AWS.
Priority	Escolha uma prioridade de failover para a instância primária do novo cluster de banco de dados. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Se você não selecionar um valor, o padrão será tier-1. Para obter mais informaçõ es, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

Opção	Descrição
Porta de banco de dados	Especifique a porta que as aplicações e os utilitários usam para acessar o banco de dados. Os clusters de banco de dados do Aurora são definidos para a porta padrão do MySQL, 3306. Em algumas empresas, os firewalls bloqueiam conexões a esta porta. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o novo cluster de banco de dados.
Monitoramento avançado	Escolha Enable enhanced monitoring (Habilitar monitoramento avançado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Monitoring Role (Monitoramento de perfis)	Disponível apenas quando Enhanced Monitoring (Monitoramento avançado) estiver definido como Enable enhanced monitoring (Habilitar monitoramento avançado). Escolha o perfil do IAM que você criou para permitir que o Amazon RDS se comunique com o Amazon CloudWatch Logs para você. Ou escolha Default (Padrão) para que o RDS crie um perfil para você chamado `rds-monitoring-role` . Para obter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Granularity	Disponível apenas quando Enhanced Monitoring estiver definido como Enable enhanced monitoring. Defina o intervalo, em segundos, em que as métricas são coletadas para o seu cluster de banco de dados.

Opção	Descrição
Atualização da versão secundária automática	Essa configuração não é aplicável aos clusters de banco de dados do Aurora MySQL. Para obter mais informações sobre atualizações de mecanismos para o Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

7. Escolha Create (Criar) para criar sua réplica de leitura entre regiões para o Aurora.

AWS CLI

Para criar um cluster de banco de dados do Aurora MySQL que seja uma réplica de leitura entre regiões com a CLI

1. Chame o comando create-db-cluster da AWS CLI na Região da AWS em que você deseja criar um cluster de banco de dados de réplica de leitura. Inclua a opção `--replication-source-identifier` e especifique o nome de recurso da Amazon (ARN) do cluster de banco de dados de origem para o qual criar uma réplica de leitura.

   Para replicação entre regiões em que o cluster de banco de dados identificado por `--replication-source-identifier` é criptografado, especifique as opções `--kms-key-id` e `--storage-encrypted`.

   > **Note**
   >
   > Você pode configurar a replicação entre regiões de um cluster de banco de dados descriptografado para uma réplica de leitura criptografada, especificando `--storage-encrypted` e fornecendo um valor para `--kms-key-id`.

   Não é possível especificar os parâmetros `--master-username` e `--master-user-password`. Esses valores são obtidos do cluster de banco de dados de origem.

   O exemplo de código a seguir cria uma réplica de leitura na região us-east-1 de um snapshot de cluster de banco de dados descriptografado na região us-west-2. O comando é chamado na

região us-east-1. Este exemplo especifica a opção `--manage-master-user-password` para gerar a senha mestra do usuário e gerenciá-la no Secrets Manager. Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Para Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

O exemplo de código a seguir cria uma réplica de leitura na região us-east-1 de um snapshot de cluster de banco de dados criptografado na região us-west-2. O comando é chamado na região us-east-1.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster \
  --kms-key-id my-us-east-1-key \
  --storage-encrypted
```

Para Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --storage-encrypted
```

A opção `--source-region` é necessária para a replicação entre as regiões GovCloud (Leste dos EUA) da AWS e GovCloud (Oeste dos EUA) da AWS, em que o cluster de banco de dados identificado por `--replication-source-identifier` é criptografado. Em `--source-region`, especifique a Região da AWS do cluster de banco de dados de origem.

Se a `--source-region` não estiver especificada, especifique um valor de `--pre-signed-url`. URL pré-assinado é aquele que contém uma solicitação assinada do Signature Versão 4 para o comando `create-db-cluster` chamado na Região da AWS de origem. Para saber mais sobre a opção `pre-signed-url`, consulte create-db-cluster na Referência de comandos da AWS CLI.

2. Verifique se o cluster de banco de dados está disponível para ser usado com o comando describe-db-clusters da AWS CLI, conforme mostrado no exemplo a seguir.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

Quando os resultados do **`describe-db-clusters`** mostram um status de `available`, crie a instância primária para o cluster de banco de dados para que a replicação possa começar. Para fazer isso, use o comando create-db-instance da AWS CLI, conforme mostrado no exemplo a seguir.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r5.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora-mysql
```

Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r5.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora-mysql
```

Quando a instância de banco de dados for criada e estiver disponível, a replicação começará. É possível determinar se a instância de banco de dados está disponível chamando o comando describe-db-instances da AWS CLI.

API do RDS

Para criar um cluster de banco de dados do Aurora MySQL que seja uma réplica de leitura entre regiões com a API

1. Chame a operação CreateDBCluster da API do RDS na Região da AWS em que você deseja criar o cluster de banco de dados da réplica de leitura. Inclua o parâmetro `ReplicationSourceIdentifier` e especifique o nome de recurso da Amazon (ARN) do cluster de banco de dados de origem para o qual criar uma réplica de leitura.

   Para replicação entre regiões em que o cluster de banco de dados identificado por `ReplicationSourceIdentifier` é criptografado, especifique o parâmetro `KmsKeyId` e defina o parâmetro `StorageEncrypted` como `true`.

   > **Note**
   >
   > Você pode configurar a replicação entre regiões de um cluster de banco de dados descriptografado para uma réplica de leitura criptografada, especificando `StorageEncrypted` como **`true`** e fornecendo um valor para `KmsKeyId`. Nesse caso, você não precisa especificar `PreSignedUrl`.

   Você não precisa incluir os parâmetros `MasterUsername` e `MasterUserPassword`, pois os valores são obtidos do cluster de banco de dados de origem.

O exemplo de código a seguir cria uma réplica de leitura na região us-east-1 de um snapshot de cluster de banco de dados descriptografado na região us-west-2. A ação é chamada na região us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

O exemplo de código a seguir cria uma réplica de leitura na região us-east-1 de um snapshot de cluster de banco de dados criptografado na região us-west-2. A ação é chamada na região us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &StorageEncrypted=true
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-
west-2%252Frds%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
```

```
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-
amz-content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Para a replicação entre as regiões GovCloud (Leste dos EUA) da AWS e GovCloud (Oeste dos EUA) da AWS, em que o cluster de banco de dados identificado por `ReplicationSourceIdentifier` é criptografado, especifique também o parâmetro `PreSignedUrl`. O URL pré-assinado deve ser uma solicitação válida para a operação `CreateDBCluster` da API que pode ser executada na Região da AWS de origem que contém o cluster de banco de dados criptografado a ser replicado. O identificador da chave do KMS é usado para criptografar a réplica de leitura e deve ser uma chave do KMS válida para a Região da AWS de destino. Para gerar um URL pré-assinado automaticamente, em vez de manualmente, use o comando copy-db-cluster da AWS CLI com a opção `--source-region`.

2. Verifique se o cluster de banco de dados está disponível para ser usado com a operação DescribeDBClusters da API do RDS, conforme mostrado no exemplo a seguir.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

Quando os resultados de `DescribeDBClusters` mostrarem um status de `available`, crie a instância primária para o cluster de banco de dados para que a replicação possa ser iniciada. Para fazer isso, use a ação CreateDBInstance da API do RDS, conforme mostrado no exemplo a seguir.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r5.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T003808Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

Quando a instância de banco de dados for criada e estiver disponível, a replicação começará. É possível determinar se a instância de banco de dados está disponível chamando o comando DescribeDBInstance da AWS CLI.

Visualizar réplicas entre regiões do Amazon Aurora MySQL

É possível visualizar as relações de replicação entre regiões de seus clusters de banco de dados do Amazon Aurora MySQL chamando o comando describe-db-clusters da AWS CLI ou a operação DescribeDBClusters da API do RDS. Na resposta, consulte o campo `ReadReplicaIdentifiers` para obter os identificadores de cluster de banco de dados de qualquer cluster de banco de dados de réplica de leitura entre regiões. Consulte o elemento `ReplicationSourceIdentifier` para obter o ARN do cluster de banco de dados de origem que é a origem da replicação.

Promover uma réplica de leitura a um cluster de banco de dados para o Aurora MySQL

Você pode promover uma réplica de leitura do Aurora MySQL para um cluster de banco de dados autônomo. Quando você promove uma réplica de leitura do Aurora MySQL, suas instâncias de banco de dados são reiniciadas antes de se tornarem disponíveis.

Normalmente, você promove uma réplica de leitura do Aurora MySQL para um cluster de banco de dados autônomo como um esquema de recuperação de dados se houver uma falha no cluster de banco de dados de origem.

Para isso, primeiro crie uma réplica de leitura e monitore o cluster de banco de dados de origem para identificar se há falhas. Em caso de falha, faça o seguinte:

1. Promova a réplica de leitura.
2. Direcione o tráfego do banco de dados para o cluster de banco de dados promovido.
3. Crie uma réplica de leitura de substituição com o cluster de banco de dados promovido como origem.

Quando você promove uma réplica de leitura, a réplica de leitura se torna um cluster de banco de dados do Aurora autônomo. O processo de promoção pode levar vários minutos ou mais para ser concluído, dependendo do tamanho da réplica de leitura. Depois da promoção da réplica de leitura para um novo cluster de banco de dados, ela será semelhante a qualquer outro cluster de banco de dados. Por exemplo, você pode criar réplicas de leitura dele e executar operações de recuperação pontuais. Você também pode criar réplicas do Aurora para o cluster de banco de dados.

Como o cluster de banco de dados promovido não é mais uma réplica de leitura, não é possível usá-lo como um destino de replicação.

As etapas a seguir mostram o processo geral para promover uma réplica de leitura para um cluster de banco de dados:

1. Interrompa a gravação de todas as transações no cluster de banco de dados de origem da réplica de leitura e aguarde todas as atualizações a serem feitas na réplica de leitura. As atualizações do banco de dados ocorrem na réplica de leitura após terem ocorrido no cluster de banco de dados de origem e esse atraso de replicação pode variar significativamente. Use a métrica `ReplicaLag` para determinar quando todas as atualizações foram feitas na réplica de leitura. A métrica `ReplicaLag` registra a quantidade de tempo que uma instância de banco de dados de réplica de leitura atrasa em relação à instância de banco de dados de origem. Quando a métrica `ReplicaLag` chega a `0`, isso mostra que a réplica de leitura alcançou a instância do banco de dados de origem.
2. Promova a réplica de leitura usando a opção Promote (Promover) no console do Amazon RDS, o comando promote-read-replica-db-cluster da AWS CLI ou a operação PromoteReadReplicaDBCluster da API do Amazon RDS.

Você escolhe uma instância de banco de dados do Aurora MySQL para promover a réplica de leitura. Depois que a réplica de leitura é promovida, o cluster de banco de dados do Aurora MySQL é promovido para um cluster de banco de dados autônomo. A instância de banco de dados com a prioridade de failover mais alta é promovida para a instância de banco de dados principal para o cluster de banco de dados. As outras instâncias de banco de dados se tornam réplicas do Aurora.

> **Note**
>
> O processo de promoção leva alguns minutos para ser concluído. Ao promover uma réplica de leitura, a replicação é interrompida, e as instâncias de banco de dados são reiniciadas. Quando a reinicialização é concluída, a réplica de leitura está disponível como um novo cluster de banco de dados.

Console

Para promover uma réplica de leitura do Aurora MySQL para um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No console, escolha Instances (Instâncias).

   O painel Instance (Instância) é exibido.
3. No painel Instances (Instâncias), escolha a réplica de leitura que você deseja promover.

   As réplicas de leitura aparecem como instâncias de banco de dados do Aurora MySQL.
4. Em Actions (Ações), escolha Promote read replica (Promover réplica de leitura).
5. Na página de confirmação, escolha Promote read replica (Promover réplica de leitura).

AWS CLI

Para promover uma réplica de leitura para um cluster de banco de dados, use o comando promote-read-replica-db-cluster da AWS CLI.

Example

Para Linux, macOS ou Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier mydbcluster
```

Para Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier mydbcluster
```

API do RDS

Para promover uma réplica de leitura para um cluster de banco de dados, chame PromoteReadReplicaDBCluster.

Solucionar problemas de réplicas entre regiões do Amazon Aurora MySQL

A seguir, você pode encontrar uma lista de mensagens de erro comuns que podem ocorrer ao criar uma réplica de leitura entre regiões do Amazon Aurora, e como resolver os erros especificados.

O cluster de origem [ARN de cluster do banco de dados] não possui logs binários ativados

Para resolver esse problema, habilite o registro em log binário no cluster de banco de dados de origem. Para obter mais informações, consulte Antes de começar.

O cluster de origem [ARN do cluster de banco de dados] não apresenta um grupo de parâmetros de cluster em sincronia no gravador

Você encontrará esse erro caso tenha atualizado o parâmetro do cluster de banco de dados `binlog_format`, mas não tenha reiniciado a instância primária do cluster de banco de dados. Reinicie a instância primária (ou seja, o gravador) do cluster de banco de dados e tente novamente.

O cluster de origem [ARN do cluster de banco de dados] já possui uma réplica de leitura nesta região

É possível ter até cinco clusters de banco de dados entre regiões que são réplicas de leitura para cada cluster de banco de dados de origem em qualquer Região da AWS. Se você já tiver o número máximo de réplicas de leitura para um cluster de banco de dados em uma determinada Região da AWS, será necessário excluir um existente, para poder criar um novo cluster de banco de dados entre regiões nessa região.

O cluster de banco de dados [ARN do cluster de banco de dados] requer uma atualização do mecanismo de banco de dados para oferecer suporte à replicação entre regiões

Para resolver esse problema, atualize a versão do mecanismo de banco de dados de todas as instâncias no cluster de banco de dados de origem para a versão mais recente do mecanismo de banco de dados e depois tente criar um banco de dados da réplica de leitura entre regiões novamente.

Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário)

Como o Amazon Aurora MySQL é compatível com o MySQL, você pode configurar a replicação entre um banco de dados MySQL e um cluster de banco de dados do Amazon Aurora MySQL. Esse tipo de replicação usa a replicação de log binário do MySQL e também referido como replicação de log binário. Se você usar a replicação de log binário com o Aurora, recomendamos que o banco de dados MySQL execute o MySQL versão 5.5 ou superior. É possível configurar a replicação em que o cluster de banco de dados do Aurora MySQL é a origem da replicação ou a réplica. É possível replicar com uma instância de banco de dados MySQL do Amazon RDS, um banco de dados MySQL externo ao Amazon RDS ou outro cluster de banco de dados do Aurora MySQL.

> **Note**
>
> Você não pode usar a replicação de log binário de/para determinados tipos de clusters de banco de dados Aurora. Em particular, a replicação de log binário não está disponível para clusters do Aurora Serverless v1. Se as instruções `SHOW MASTER STATUS` e `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou a instrução `SHOW REPLICA STATUS` (Aurora MySQL versão 3) não retornarem uma saída, verifique se o cluster em uso oferece suporte à replicação de log binário.

Também é possível replicar com uma instância de banco de dados do RDS para MySQL ou com um cluster de banco de dados do Aurora MySQL em outra Região da AWS. Quando estiver executando a replicação entre Regiões da AWS, verifique se os clusters e as instâncias de banco de dados estão acessíveis publicamente. Se os clusters de banco de dados do Aurora MySQL estiverem em sub-redes privadas em sua VPC, use o emparelhamento de VPC entre as Regiões da AWS. Para ter mais informações, consulte Um cluster de banco de dados em uma VPC acessada por uma instância do EC2 em uma VPC diferente.

Se você quiser configurar a replicação entre um cluster de banco de dados do Aurora MySQL e um cluster de banco de dados do Aurora MySQL em outra Região da AWS, poderá criar um cluster de banco de dados do Aurora MySQL como uma réplica de leitura em uma Região da AWS diferente da região do cluster de banco de dados de origem. Para ter mais informações, consulte Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS.

Com o Aurora MySQL versões 2 e 3, é possível replicar entre o Aurora MySQL e uma origem ou um destino externo que utilize identificadores de transação global (GTIDs) para replicação. Certifique-se de que os parâmetros relacionados ao GTID no cluster de banco de dados do Aurora MySQL tenham configurações compatíveis com o status de GTID do banco de dados externo. Para aprender a fazer isso, consulte Usar a replicação baseada em GTID. No Aurora MySQL versão 3.01 e posteriores, é possível escolher como atribuir GTIDs a transações replicadas de uma origem que não utiliza GTIDs. Para saber mais sobre o procedimento armazenado que controla essa configuração, consulte mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versão 3).

> **Warning**
>
> Ao replicar entre o Aurora MySQL e o MySQL, lembre-se de usar apenas tabelas do InnoDB. Se você tiver tabelas do MyISAM que deseja replicar, poderá convertê-las no formato do InnoDB antes de configurar a replicação, com o seguinte comando.
>
> ```
> alter table <schema>.<table_name> engine=innodb, algorithm=copy;
> ```

Nas seções a seguir, configure a replicação, interrompa a replicação, escale as leituras para o banco de dados, otimize a replicação de logs binários e configure o log binário avançado.

Tópicos

- Configurar a replicação de logs binários para o Aurora MySQL
- Interromper a replicação de logs binários para o Aurora MySQL
- Escalar leituras para o banco de dados MySQL com o Amazon Aurora
- Otimizar a replicação de logs binários para Aurora MySQL
- Configurar o log binário avançado para Aurora MySQL

Configurar a replicação de logs binários para o Aurora MySQL

Configurar a replicação do MySQL com o Aurora MySQL envolve as seguintes etapas, que são abordadas em detalhes:

Sumário

- 1. Habilitar o registro em log binário na origem de replicação
- 2. Retenha os logs binários na origem da replicação até que não seja necessário
- 3. Criar uma cópia ou um despejo da origem da replicação
- 4. Carregue o despejo em seu destino de réplica (se necessário).
- 5. Criar um usuário de replicação em sua origem de replicação
- 6. Habilitar a replicação no seu destino de réplica
  - Configurar um local para interromper a replicação para uma réplica de leitura
- 7. Monitore sua réplica
- Como sincronizar senhas entre a fonte e o destino da replicação

1. Habilitar o registro em log binário na origem de replicação

Encontre instruções a seguir sobre como habilitar o registro em log binário na origem de replicação para seu o mecanismo de banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Para habilitar o registro em log binário em um cluster de banco de dados do Aurora MySQL Defina o parâmetro de cluster de banco de dados `binlog_format` como `ROW`, `STATEMENT` ou `MIXED`. `MIXED` é recomendável, a menos que você precise de um formato de log binário específico. (O valor padrão é `OFF`.) Para alterar o parâmetro `binlog_format` , crie um grupo de parâmetros de cluster de banco de dados personalizado e associe esse grupo ao seu cluster de banco de dados. Não é possível alterar os parâmetros no grupo de parâmetros de cluster de banco de dados padrão.

Mecanismo do banco de dados	Instruções
	Se você estiver alterando o parâmetro `binlog_format` de `OFF` para outro valor, reinicialize o cluster de banco de dados Aurora para que a alteração entre em vigor. Para obter mais informações, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora e Grupos de parâmetros para Amazon Aurora.
RDS para MySQL	Para habilitar o registro em log binário em uma instância de banco de dados do Amazon RDS Não é possível habilitar o registro em log binário diretamente para uma instância de banco de dados do Amazon RDS, mas é possível habilitá-lo seguindo um destes procedimentos: • Habilite backups automáticos da instância de banco de dados. Você pode habilitar backups automáticos ao criar uma instância de banco de dados ou pode habilitar backups modificando uma instância de banco de dados existente. Para ter mais informações, consulte Criar uma instância de banco de dados no Guia do usuário do Amazon RDS. • Crie uma réplica de leitura para a instância de banco de dados. Para ter mais informações, consulte Como trabalhar com réplicas de leitura no Guia do usuário do Amazon RDS.

Mecanismo do banco de dados	Instruções
MySQL (externo)	Para configurar replicação criptografada Para replicar dados de forma segura com o Aurora MySQL versão 2, você pode usar a replicação criptografada. **Note** Se você não precisar usar a replicação criptografada, você pode pular essas etapas. Veja a seguir os pré-requisitos para usar a replicação criptografada: • O Secure Sockets Layer (SSL) deve ser habilitado no banco de dados de origem externa do MySQL. • Uma chave e certificado de cliente devem estar preparados para o cluster do banco de dados do Aurora MySQL. Durante a replicação criptografada, o cluster do banco de dados de Aurora MySQL age como um cliente para o servidor de banco de dados do MySQL. Os certificados e chaves do cliente Aurora MySQL estão nos arquivos em formato .pem. 1. Certifique-se de que você estará preparado para a replicação criptografada: • Se você não tem o SSL habilitado no banco de dados de origem externo do MySQL e não tem uma chave de cliente e um certificado de cliente preparado s, habilite o SSL no servidor de banco de dados MySQL e gere a chave de cliente e o certificado de cliente necessários. • Se o SSL estiver habilitado na origem externa, forneça uma chave e um certificado de cliente para o cluster de banco de dados do Aurora MySQL. Se você não os tiver, gere uma nova chave e certificado para o cluster de banco de dados do Aurora MySQL. Para assinar o certificado de cliente, é necessário

Mecanismo do banco de dados	Instruções

ter a chave de autoridade de certificado usada para configurar o SSL no banco de dados de origem externa do MySQL.

Para ter mais informações, consulte Creating SSL certificates and keys using openssl na documentação do MySQL.

Você precisa do certificado de autoridade de certificação, a chave do cliente e o certificado do cliente.

2. Conecte-se ao cluster de banco de dados do Aurora MySQL como o usuário mestre usando SSL.

   Para obter informações sobre como se conectar ao cluster de banco de dados do Aurora MySQL com SSL, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

3. Execute o procedimento armazenado `mysql.rds_import_binlog_ssl_material` para importar as informações de SSL para o cluster de banco de dados do Aurora MySQL.

   Para o parâmetro `ssl_material_value`, insira as informações dos arquivos em formato .pem no cluster de banco de dados do Aurora MySQL, na carga JSON correta.

   O exemplo a seguir importa informações SSL em um cluster de banco de dados Aurora MySQL. Em arquivos de formato .pem, o código do corpo geralmente é maior que o código de corpo exibido no exemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
```

Mecanismo do banco de dados	Instruções

```
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICA
TE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE
 KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pc
jqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSF
JuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJ
tjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMu
cxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Para obter mais informações, consulte mysql.rds_import_binlog_ssl_material e Conexões do TLS com clusters de banco de dados do Aurora MySQL.

> **Note**
>
> Após executar o procedimento, os segredos são armazenados em arquivos. Para apagar os arquivos posteriormente, você pode executar o procedimento armazenado mysql.rds_remove_binlog_ssl_material.

Mecanismo do banco de dados	Instruções
	Para habilitar o registro em log binário em um banco de dados MySQL externo 1. Em um shell de comando, interrompa o serviço mysql. ``` sudo service mysqld stop ``` 2. Edite o arquivo `my.cnf` (esse arquivo normalmente se encontra em `/etc`). ``` sudo vi /etc/my.cnf ``` Adicione as opções `log_bin` e `server_id` à seção `[mysqld]`. A opção `log_bin` fornece um identificador de nome de arquivo para arquivos de log binário. A opção `server_id` fornece um identificador exclusivo para o servidor em relações entre origem e réplica. Se a replicação criptografada não for necessária, verifique se o banco de dados MySQL externo é iniciado com binlogs habilitados e o SSL desabilitado. A seguir veja as entradas relevantes no arquivo `/etc/my.cnf` para obter dados não criptografados. ``` log-bin=mysql-bin server-id=2133421 innodb_flush_log_at_trx_commit=1 sync_binlog=1 ``` Se a replicação criptografada for necessária, assegure-se de que o banco de dados MySQL externo é iniciado SSL e logs binários estão habilitados. As entradas no arquivo `/etc/my.cnf` incluem os locais de arquivos .pem para servidor de banco de dados MySQL. ``` log-bin=mysql-bin server-id=2133421 innodb_flush_log_at_trx_commit=1 sync_binlog=1

Mecanismo do banco de dados	Instruções

```
# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Além disso, a opção `sql_mode` de sua Instância de banco de dados MySQL deve ser definida como 0, ou não deve ser incluída no arquivo my.cnf.

Quando conectado ao banco de dados MySQL externo, grave a posição do log binário do banco de dados externo do MySQL.

```
mysql> SHOW MASTER STATUS;
```

Sua saída deve ser similar à seguinte:

```
+------------------+----------+--------------+------------------
+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
 Executed_Gtid_Set |
+------------------+----------+--------------+------------------
+-------------------+
| mysql-bin.000031 |      107 |              |                  |
              |
+------------------+----------+--------------+------------------
+-------------------+
1 row in set (0.00 sec)
```

Para ter mais informações, consulte Setting the replication source configuration na documentação do MySQL.

3. Inicie o serviço mysql.

```
sudo service mysqld start
```

2. Retenha os logs binários na origem da replicação até que não seja necessário

Quando você usa a replicação de log binário do MySQL, o Amazon RDS não gerencia o processo de replicação. Como resultado, é necessário garantir que os arquivos de log binário na origem da replicação sejam retidos até que as alterações tenham sido aplicadas à réplica. Esta manutenção ajuda você a restaurar o banco de dados de origem em caso de falha.

Use as instruções a seguir para manter os logs binários para o mecanismo do seu banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Para manter os logs binários em um cluster de banco de dados do Aurora MySQL Você não tem acesso aos arquivos de log binário de um cluster de banco de dados do Aurora MySQL. Como resultado, é necessário escolher um período para reter os arquivos de log binário na origem da replicação o suficiente para garantir que as alterações tenham sido aplicadas à réplica antes que o arquivo de log binário seja excluído pelo Amazon RDS. Você pode manter arquivos de log binário em um cluster de banco de dados do Aurora MySQL por até 90 dias. Se você for configurar a replicação com um banco de dados MySQL ou instância de banco de dados do RDS para MySQL como a réplica e o banco de dados para o qual você estiver criando uma réplica for muito grande, escolha um período longo para manter arquivos de log binário até que a cópia inicial do banco de dados para a réplica seja concluída e o atraso da réplica chegue a 0. Para definir o período de retenção do log binário, use o procedimento mysql.rds_set_configuration e especifique um parâmetro de configuração `'binlog retention hours'`, juntamente com o número de horas de retenção dos arquivos de log binário no cluster do banco de dados. O valor máximo para o Aurora MySQL versão 2.11.0 e posterior e versão 3 é de 2.160 (90 dias). O exemplo a seguir define o período de retenção para arquivos de log binário em seis dias: `CALL mysql.rds_set_configuration('binlog retention hours', 144);`

Mecanismo do banco de dados	Instruções
	Após a replicação ter sido iniciada, você poderá verificar se as mudanças foram aplicadas à sua réplica executando o comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3) na sua réplica e verificando o campo `Seconds behind master`. Se o campo `Seconds behind master` for 0, não haverá atraso de réplica. Quando não há atraso de réplica, reduza o período em que os arquivos de log binário são mantidos definindo o parâmetro de configuração `binlog retention hours` para um período menor. Se essa configuração não for especificada, o padrão de Aurora MySQL será 24 (1 dia). Se você especificar um valor para `'binlog retention hours'` que seja maior que o valor máximo, o Aurora MySQL usará o valor máximo.
RDS para MySQL	Para manter os logs binários em uma instância de banco de dados do Amazon RDS É possível manter arquivos de log binário em uma instância de banco de dados do Amazon RDS definindo os horários de retenção do log binário, assim como é feito com o cluster de banco de dados do Aurora MySQL descrito na seção anterior. Você também pode manter arquivos de log binário em uma instância de banco de dados do Amazon RDS criando uma réplica de leitura para a instância de banco de dados. Esta réplica de leitura é temporária e tem como finalidade exclusiva manter os arquivos de log binário. Depois de criar a réplica de leitura, chame o procedimento [mysql.rds_stop_replication] nela. Enquanto a replicação estiver interrompida, o Amazon RDS não excluirá nenhum dos arquivos de log binário na origem da replicação. Após configurar a replicação com a réplica permanente, você poderá excluir a réplica de leitura quando o atraso da réplica (campo `Seconds behind master`) entre a origem da replicação e a réplica permanente chegar a 0.

Mecanismo do banco de dados	Instruções
MySQL (externo)	Para manter os logs binários em um banco de dados MySQL externo Como os arquivos de log binário em um banco de dados MySQL externo não são gerenciados pelo Amazon RDS, eles serão mantidos até você os exclua. Após a replicação ter sido iniciada, você poderá verificar se as mudanças foram aplicadas à sua réplica executando o comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3) na sua réplica e verificando o campo `Seconds behind master`. Se o campo `Seconds behind master` for 0, não haverá atraso de réplica. Quando não há atraso de réplica, você pode excluir arquivos de log binário antigos.

3. Criar uma cópia ou um despejo da origem da replicação

Use um snapshot, clone ou despejo da origem da replicação para carregar uma cópia da linha de base dos dados na réplica. Então você começa a replicar a partir desse ponto.

Use as instruções a seguir para criar uma cópia ou um despejo da origem da replicação para o mecanismo do banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Como criar uma cópia de um cluster de banco de dados do Aurora MySQL Use um dos seguintes métodos: • Restaure de um snapshot de cluster de banco de dados: 1. Crie um snapshot de cluster de banco de dados do seu cluster de banco de dados Amazon Aurora. Para ter mais informações, consulte Criar um snapshot de cluster de banco de dados. 2. Crie um novo cluster de banco de dados Aurora restaurando a partir do snapshot de cluster de banco de dados que você acabou de criar.

Mecanismo do banco de dados	Instruções

Certifique-se de manter o mesmo grupo de parâmetros de banco de dados para o cluster de banco de dados restaurado como seu cluster de banco de dados original. Isso garantirá que a cópia do seu cluster de banco de dados tenha o log binário habilitado. Para obter mais informações, consulte Restauração de um snapshot de um cluster de banco de dados.

- Clone o cluster de banco de dados. Para obter mais informações, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora.

Como determinar o nome e a posição do arquivo de log binário

Use um dos seguintes métodos:

- No Console de gerenciamento da AWS:
  1. Escolha Bancos de dados e, depois, selecione a instância primária (gravador) do seu cluster de banco de dados do Aurora para mostrar os detalhes.
  2. Role até Recent Events. Uma mensagem de evento mostra que inclui o nome e a posição do arquivo de log binário. A mensagem de evento está no seguinte formato.

     ```
     Binlog position from crash recovery is binlog-file-name  binlog-position
     ```

  3. Salve o nome do arquivo de log binário e os valores de posição para quando você iniciar a replicação.
- Chame o comando describe-events da AWS CLI, como no exemplo a seguir.

```
aws rds describe-events

{
    "Events": [
        {
            "EventCategories": [],
            "SourceType": "db-instance",
```

Mecanismo do banco de dados	Instruções

```
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:
db:sample-restored-instance",
            "Date": "2016-10-28T19:43:46.862Z",
            "Message": "Binlog position from crash recovery is mysql-
bin-changelog.000003 4278",
            "SourceIdentifier": "sample-restored-instance"
        }
    ]
}
```

- Confira o log de erros do MySQL para saber a última posição do arquivo de log binário do MySQL.

Como criar um despejo de um cluster de banco de dados do Aurora MySQL

Se o destino da réplica for um banco de dados externo do MySQL ou uma instância de banco de dados do RDS para MySQL, será necessário criar um arquivo de despejo do cluster de banco de dados do Aurora.

Execute o comando `mysqldump` na cópia do cluster de banco de dados de origem que você criou. Isso evita considerações de bloqueio durante o despejo. Se o despejo fosse feito diretamente no cluster de banco de dados de origem, seria necessário bloquear as tabelas de origem para evitar gravações simultâneas nelas durante o despejo.

1. Conecte-se ao cluster de banco de dados usando um cliente MySQL.
2. Emita o comando `mysqldump` . Por exemplo:

```
PROMPT> mysqldump --databases  database_name  --single-transaction
--order-by-primary -r backup.sql -u    local_user s -p
```

3. Depois de criar o arquivo de despejo, é possível excluir a cópia do cluster de banco de dados.

Mecanismo do banco de dados	Instruções
RDS para MySQL	Para criar um snapshot da instância de banco de dados do Amazon RDS Crie uma réplica de leitura de sua instância de banco de dados do Amazon RDS. Para ter mais informações, consulte Criar uma réplica de leitura no Guia do usuário do Amazon Relational Database Service. 1. Conecte-se à sua réplica de leitura e interrompa a replicação executando o procedimento mysql.rds_stop_replication. 2. Enquanto a réplica de leitura estiver no estado Interrompida, conecte-se a ela e execute o comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3). Recupere o nome de arquivo de log binário atual no campo `Relay_Master_Log_File` e a posição do arquivo de log no campo `Exec_Master_Log_Pos` . Salve esses valores para quando você iniciar a replicação. 3. Enquanto a réplica de leitura permanece Stopped (Interrompido), crie um snapshot de banco de dados da réplica de leitura. Para ter mais informações, consulte Criar um snapshot de banco de dados no Guia do usuário do Amazon Relational Database Service. 4. Exclua a réplica de leitura.

Mecanismo do banco de dados	Instruções

MySQL (externo)	Como criar um despejo de um banco de dados MySQL externo
	1. Antes de criar um despejo, é necessário garantir que o local de log binário do despejo seja atual com os dados na instância de origem. Para isso, primeiro você deve interromper todas as operações de gravação da instância com o seguinte comando:
	`mysql> FLUSH TABLES WITH READ LOCK;`
	2. Crie um despejo de seu banco de dados MySQL usando o comando `mysqldump`, conforme mostrado a seguir:
	`PROMPT> sudo mysqldump --databases database_name --master-data=2 --single-transaction \ --order-by-primary -r backup.sql -u local_user -p`
	3. Após criar o despejo, desbloqueie as tabelas em seu banco de dados MySQL com o seguinte comando:
	`mysql> UNLOCK TABLES;`

4. Carregue o despejo em seu destino de réplica (se necessário).

Se você planeja carregar dados de um despejo de um banco de dados do MySQL que seja externo ao Amazon RDS, é recomendável criar uma instância do EC2 na qual copiar os arquivos de despejo. Depois, é possível carregar os dados no cluster de banco de dados ou na instância de banco de dados a partir dessa instância do EC2. Com essa abordagem, você pode compactar o(s) arquivo(s) de despejo antes de copiá-los na instância do EC2 para reduzir os custos de rede associados à cópia de dados para o Amazon RDS. Você também pode criptografar o arquivo ou arquivos de despejo para proteger os dados à medida que são transferidos pela rede.

> **Note**
>
> Se você criar um cluster de banco de dados do Aurora MySQL como o destino da réplica, não precisará carregar um arquivo de despejo.
>
> - É possível restaurar de um snapshot de cluster de banco de dados para criar um cluster de banco de dados. Para obter mais informações, consulte Restauração de um snapshot de um cluster de banco de dados.
> - É possível clonar o cluster de banco de dados de origem para criar um cluster de banco de dados. Para obter mais informações, consulte Clonar um volume para um cluster de banco de dados do Amazon Aurora.
> - É possível migrar os dados de um snapshot de instância de banco de dados para um novo cluster de banco de dados. Para obter mais informações, consulte Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL.

Use as instruções a seguir para carregar o despejo da origem da replicação no destino de réplica para o mecanismo do banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Como carregar um despejo em um cluster de banco de dados do Aurora MySQL 1. Copie a saída do comando `mysqldump` da origem da replicação para um local que também possa ser conectado ao cluster de banco de dados do Aurora MySQL. 2. Conecte-se ao seu cluster de banco de dados do Aurora MySQL usando o comando `mysql`. Veja um exemplo a seguir. `PROMPT> mysql -h host_name -port=3306 -u db_master_user -p` 3. No prompt `mysql`, execute o comando `source` e passe a ele o nome do arquivo de despejo do banco de dados para carregar os dados no cluster de banco de dados Aurora MySQL, por exemplo:

Mecanismo do banco de dados	Instruções
	`mysql> source backup.sql;`

RDS para MySQL	Como carregar um despejo em uma instância de banco de dados do Amazon RDS
	1. Copie a saída do comando `mysqldump` da origem da replicação para um local que também possa ser conectado à instância de banco de dados MySQL.
	2. Conecte-se a sua instância de banco de dados MySQL usando o comando `mysql`. Veja um exemplo a seguir.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. No prompt `mysql`, execute o comando `source` e passe a ele o nome do arquivo de despejo do banco de dados para carregar os dados na instância de banco de dados MySQL, por exemplo:
	`mysql> source backup.sql;`

Mecanismo do banco de dados	Instruções

MySQL (externo)	Como carregar um despejo em um banco de dados MySQL externo
	
	Você não pode carregar um snapshot de banco de dados ou um snapshot de cluster de banco de dados em um banco de dados MySQL externo. Em vez disso, você deve usar o resultado do comando `mysqldump` .
	
	1. Copie a saída do comando `mysqldump` da origem da replicação para um local que também possa ser conectado ao banco de dados MySQL.
	2. Conecte-se ao seu banco de dados MySQL usando o comando `mysql`. Veja um exemplo a seguir.
	
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	
	3. No prompt `mysql`, execute o comando `source` e passe a ele o nome do arquivo de despejo do banco de dados para carregar os dados em seu banco de dados MySQL. Veja um exemplo a seguir.
	
	`mysql> source backup.sql;`

5. Criar um usuário de replicação em sua origem de replicação

Crie um ID de usuário na origem usado exclusivamente para replicação. O exemplo a seguir é relacionado ao RDS para MySQL ou bancos de dados de origem externa do MySQL.

```
mysql> CREATE USER 'repl_user'@'domain_name' IDENTIFIED BY 'password';
```

Para bancos de dados de origem do Aurora MySQL, o parâmetro do cluster de banco de dados `skip_name_resolve` é definido como 1 (`ON`) e não pode ser modificado, então é necessário usar um endereço IP para o host em vez de um nome de domínio. Para ter mais informações, consulte skip_name_resolve na documentação do MySQL.

```
mysql> CREATE USER 'repl_user'@'IP_address' IDENTIFIED BY 'password';
```

O usuário requer os privilégios `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concede esses privilégios ao usuário.

Se você precisar usar replicação criptografada, exija conexões SSL para o usuário de replicação. Por exemplo, é possível usar uma das declarações a seguir para solicitar conexões SSL na conta de usuário `repl_user`.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'IP_address';
```

```
GRANT USAGE ON *.* TO 'repl_user'@'IP_address' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` não está incluído, a conexão de replicação pode silenciosamente cair de volta para uma conexão não criptografada.

6. Habilitar a replicação no seu destino de réplica

Antes de habilitar a replicação, convém obter um snapshot manual do destino de réplica do cluster de banco de dados do Aurora MySQL ou da instância de banco de dados do RDS para MySQL. Se surgir um problema e for preciso restabelecer a replicação com o cluster de banco de dados ou o destino de réplica da instância de banco de dados, você poderá restaurar o cluster de banco de dados ou a instância de banco de dados desse snapshot em vez de precisar importar os dados em seu destino de réplica novamente.

Use as instruções a seguir para habilitar a replicação do mecanismo do seu banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Para habilitar a replicação de um cluster de banco de dados do Aurora MySQL 1. Encontre o ponto de partida da replicação. Você precisa do nome do arquivo de log binário e da posição do log binário. Se seu destino de réplica de cluster de banco de dados foi criado usando o seguinte:

Mecanismo do banco de dados	Instruções

- Snapshot do cluster de banco de dados o clone: recupere o nome e a posição do arquivo de log binário dos eventos recentes do novo cluster de banco de dados, conforme mostrado em 3. Criar uma cópia ou um despejo da origem da replicação.
- Snapshot de banco de dados: você recuperou o nome de log binário e a posição do comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3) ao criar o snapshot de sua origem de replicação.

2. Conecte-se ao cluster de banco de dados e chame os seguintes procedimentos para iniciar a replicação com a origem da replicação usando o nome e o local do arquivo de log binário da etapa anterior:
   - mysql.rds_set_external_source (Aurora MySQL versão 3)
   - mysql.rds_set_external_master (Aurora MySQL versão 2)
   - mysql.rds_start_replication (todas as versões)

   O exemplo a seguir é para o Aurora MySQL versão 3.

```
CALL mysql.rds_set_external_source ('mydbinstance.123456789012
.us-east-1.rds.amazonaws.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107,
 0);
CALL mysql.rds_start_replication;
```

Para usar a criptografia SSL, defina o valor final como 1 em vez de `0`.

Mecanismo do banco de dados	Instruções

RDS para MySQL	Para habilitar a replicação de uma instância de banco de dados do Amazon RDS
	
	1. Se o seu destino de réplica da instância de banco de dados foi criado a partir de um snapshot de banco de dados, você precisará do arquivo de log binário e da posição do log binário que representam o ponto inicial para a replicação. Você recuperou esses valores do comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3) ao criar o snapshot da sua origem de replicação.
	2. Conecte-se à instância de banco de dados e chame os procedimentos mysql.rds_set_external_master (Aurora MySQL versão 2) ou mysql.rds_set_external_source (Aurora MySQL versão 3) e mysql.rds_start_replication para iniciar a replicação com a origem da replicação. Use o nome e a localização do arquivo de log binário da etapa anterior. Veja um exemplo a seguir.
	
	`CALL mysql.rds_set_external_master ('mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);`
	`CALL mysql.rds_start_replication;`
	
	Para usar a criptografia SSL, defina o valor final como 1 em vez de 0.

Mecanismo do banco de dados	Instruções

MySQL (externo)	Para habilitar a replicação a partir de um banco de dados MySQL externo
	1. Recupere o arquivo de log binário e a posição de log binário que representam são o ponto de partida para a replicação. Você recuperou esses valores do comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3) ao criar o snapshot da sua origem de replicação. Se o seu destino de réplica do MySQL externo tiver sido preenchido com o resultado do comando `mysqldump` com a opção `--master-data=2` , o arquivo de log binário e a posição do log binário estão inclusos no resultado. Veja um exemplo a seguir.
	`-- -- Position to start replication or point-in-time recovery from -- -- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;`
	2. Conecte-se ao destino da réplica do MySQL externo e emita `CHANGE MASTER TO` e `START SLAVE` (Aurora MySQL versão 2) ou `START REPLICA` (Aurora MySQL versão 3) para iniciar a replicação com sua origem de replicação utilizando o nome e a localização do arquivo de log binário da etapa anterior, por exemplo:
	`CHANGE MASTER TO MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com', MASTER_PORT = 3306, MASTER_USER = 'repl_user', MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin-changelog.000031', MASTER_LOG_POS = 107; -- And one of these statements depending on your engine version: START SLAVE; -- Aurora MySQL version 2 START REPLICA; -- Aurora MySQL version 3`

Se a replicação falhar, isso pode resultar em um grande aumento na E/S não intencional na réplica, o que pode prejudicar a performance. Se a replicação falhar ou não for mais necessária, você poderá executar o procedimento armazenado mysql.rds_reset_external_master (Aurora MySQL versão 2) ou mysql.rds_reset_external_source (Aurora MySQL versão 3) para remover a configuração de replicação.

Configurar um local para interromper a replicação para uma réplica de leitura

No Aurora MySQL versão 3.04 e posterior, você pode iniciar a replicação e interrompê-la em um local especificado do arquivo de log binário usando o procedimento mysql.rds_start_replication_until(Aurora MySQL versão 3) armazenado.

Para iniciar a replicação para uma Réplica de leitura e interrompê-la em um local específico

1. Usando um cliente do MySQL, conecte-se ao cluster de banco de dados do Aurora MySQL de réplica como o usuário mestre.
2. Execute o procedimento armazenado mysql.rds_start_replication_until(Aurora MySQL versão 3).

   O exemplo a seguir inicia a replicação e replica as alterações até que ela atinja o local `120` no arquivo de log binário `mysql-bin-changelog.000777`. Em um cenário de recuperação de desastres, suponha que o local `120` é imediatamente antes do desastre.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

A replicação é interrompida automaticamente quando o ponto de interrupção é atingido. O seguinte evento do RDS é gerado: `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure`.

Caso você use a replicação baseada em GTID, use o procedimento armazenado mysql.rds_start_replication_until_gtid(Aurora MySQL versão 3) em vez do procedimento armazenado mysql.rds_start_replication_until(Aurora MySQL versão 3). Para obter mais informações sobre a replicação baseada em GTID, consulte Usar a replicação baseada em GTID.

7. Monitore sua réplica

Ao configurar a replicação do MySQL com um cluster de banco de dados do Aurora MySQL, você deve monitorar eventos de failover para o cluster de banco de dados do Aurora MySQL quando se tratar do destino da réplica. Se ocorrer um failover, o cluster de banco de dados que for seu destino de réplica poderá ser recriado em um novo host com um endereço de rede diferente. Para obter informações sobre como monitorar eventos de failover, consulte Trabalhar com a notificação de eventos do Amazon RDS.

Também é possível monitorar até que ponto o destino da réplica está atrasado em relação à origem da replicação conectando-se a esse destino e executando o comando `SHOW SLAVE STATUS` (Aurora MySQL versão 2) ou `SHOW REPLICA STATUS` (Aurora MySQL versão 3). No resultado do comando, o campo `Seconds Behind Master` indica quanto o destino da réplica está atrasado em relação à origem.

> **Important**
>
> Se você atualizar o cluster de banco de dados e especificar um grupo de parâmetros personalizado, será necessário reinicializar manualmente o cluster após a conclusão da atualização. Fazer isso faz com que o cluster use as novas configurações de parâmetros personalizados e reinicie a replicação do log binário.

Como sincronizar senhas entre a fonte e o destino da replicação

Ao alterar contas e senhas de usuário na fonte de replicação usando instruções SQL, as alterações são automaticamente replicadas para o destino de replicação.

Se você usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para alterar a senha primária na fonte de replicação, essas alterações não serão replicadas automaticamente para o destino de replicação. Se quiser sincronizar o usuário primário e a senha primária entre os sistemas de fonte e de destino, você deverá fazer a mesma alteração no destino de replicação por conta própria.

Interromper a replicação de logs binários para o Aurora MySQL

Para interromper a replicação do log binário com uma instância de banco de dados MySQL, um banco de dados MySQL externo ou outro cluster de banco de dados Aurora, siga estas etapas, abordadas a fundo no tópico a seguir.

1. Interrompa a replicação do log binário no destino da réplica

2. Desabilitar o registro em log binário na origem da replicação

1. Interrompa a replicação do log binário no destino da réplica

Use as instruções a seguir para interromper a replicação de log binário para o mecanismo do seu banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Para interromper a replicação de log binário em um destino de réplica de cluster de banco de dados do Aurora MySQL Conecte-se ao cluster de banco de dados Aurora que é o destino da réplica e chame o procedimento mysql.rds_stop_replication.
RDS para MySQL	Para interromper a replicação de log binário em uma instância de banco de dados do Amazon RDS Conecte-se à instância de banco de dados RDS que é o destino da réplica e chame o procedimento mysql.rds_stop_replication.
MySQL (externo)	Para interromper a replicação de um log binário em um banco de dados MySQL externo Conecte-se ao banco de dados MySQL e execute o comando `STOP SLAVE` (versão 5.7) ou `STOP REPLICA` (versão 8.0).

2. Desabilitar o registro em log binário na origem da replicação

Use as instruções da tabela a seguir para desativar o registro em log binário na origem da replicação para o mecanismo do seu banco de dados.

Mecanismo do banco de dados	Instruções
Aurora MySQL	Para desabilitar o registro em log binário em um cluster de banco de dados do Amazon Aurora 1. Conecte-se ao cluster de banco de dados do Aurora, que é a origem da replicação. 2. Use o procedimento mysql.rds_set_configuration e especifique o parâmetro de configuração `binlog retention hours`, com o valor `NULL`, conforme mostrado no exemplo a seguir. `CALL mysql.rds_set_configuration('binlog retention hours', NULL);` **Note** Não é possível usar o valor `0` para `binlog retention hours`. 3. Defina o parâmetro `binlog_format` como `OFF` na origem da replicação. O parâmetro `binlog_format` está no grupo de parâmetros de cluster de banco de dados associado ao seu cluster de banco de dados. Após ter alterado o valor do parâmetro `binlog_format` , reinicie seu cluster de banco de dados para que a alteração entre em vigor. Para obter mais informações, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora e Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.
RDS para MySQL	Para desabilitar o registro em log binário em uma instância de banco de dados do Amazon RDS Não é possível desabilitar o registro em log binário diretamente para uma instância de banco de dados do Amazon RDS, mas é possível desabilitá-lo seguindo um destes procedimentos:

Mecanismo do banco de dados	Instruções
	1. Desabilite backups automáticos da instância de banco de dados. É possível desabilitar backups automáticos modificando uma instância de banco de dados existente e definindo Backup Retention Period (Período de retenção de backup) como 0. Para ter mais informações, consulte Modificar uma instância de banco de dados do Amazon RDS e Trabalhar com backups no Guia do usuário do Amazon Relational Database Service. 2. Exclua todas as réplicas de leitura para a instância de banco de dados. Para ter mais informações, consulte Trabalho com réplicas de leitura do MariaDB, MySQL e instâncias de banco de dados PostgreSQL no Guia do usuário do Amazon Relational Database Service.

MySQL (externo)	Para desabilitar o registro em log binário em um banco de dados MySQL externo
	
	Conecte-se ao banco de dados MySQL e chame o comando `STOP REPLICATION` .
	
	1. Em um shell de comando, interrompa o serviço `mysqld`:
	
	`sudo service mysqld stop`
	
	2. Edite o arquivo `my.cnf` (esse arquivo normalmente se encontra em `/etc`).
	
	`sudo vi /etc/my.cnf`
	
	Exclua as opções `log_bin` e `server_id` da seção `[mysqld]`.
	
	Para ter mais informações, consulte Setting the replication source configuration na documentação do MySQL.
	
	3. Inicie o serviço mysql.
	
	`sudo service mysqld start`

Escalar leituras para o banco de dados MySQL com o Amazon Aurora

Você pode usar o Amazon Aurora com sua instância de banco de dados MySQL para aproveitar os recursos de escalabilidade de leitura do Amazon Aurora e expandir a workload de leitura de sua instância do banco de dados MySQL. Para usar o Aurora com a intenção de dimensionar as leituras da instância de banco de dados MySQL, crie um cluster de banco de dados do Amazon Aurora MySQL e faça dele uma réplica de leitura da instância de banco de dados MySQL. Isso se aplica a uma instância de banco de dados do RDS para MySQL ou a um banco de dados MySQL executado externamente em relação ao Amazon RDS.

Para obter informações sobre como criar com um cluster de banco de dados do Amazon Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Ao configurar a replicação entre sua instância de banco de dados MySQL e seu cluster de banco de dados Amazon Aurora, certifique-se de seguir estas diretrizes:

- Use o endereço de endpoint do cluster de banco de dados do Amazon Aurora ao fazer referência a seu cluster de banco de dados do Amazon Aurora MySQL. Se ocorrer um failover, a réplica do Aurora promovida a instância primária para o cluster de banco de dados do Aurora MySQL continuará a usar o endereço do endpoint do cluster de banco de dados.
- Retenha os logs binários na instância de gravador até confirmar que eles foram aplicados à réplica do Aurora. Essa manutenção garante que você possa restaurar a instância de gravador em caso de falha.

> **Important**
>
> Ao usar a replicação autogerenciada, você será responsável por monitorar e resolver quaisquer problemas de replicação que possam ocorrer. Para ter mais informações, consulte Diagnosticar e resolver atrasos entre réplicas de leitura.

> **Note**
>
> As permissões necessárias para iniciar a replicação em um cluster de banco de dados do Aurora MySQL são restritas e não estão disponíveis ao seu usuário principal do Amazon RDS. Portanto, você deve usar os procedimentos mysql.rds_set_external_master (Aurora MySQL versão 2) ou mysql.rds_set_external_source (Aurora MySQL versão 3) e

mysql.rds_start_replication para configurar a replicação entre o cluster de banco de dados do Aurora MySQL e a instância de banco de dados MySQL.

Iniciar a replicação entre uma instância de origem externa e um cluster de banco de dados do Aurora MySQL

1. Confirme que a instância do banco de dados MySQL é somente leitura:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Execute o comando `SHOW MASTER STATUS` na instância do banco de dados MySQL de origem para determinar a localização do log binário. Você recebe um resultado semelhante ao seguinte exemplo:

```
File                          Position
-------------------------------------
 mysql-bin-changelog.000031      107
-------------------------------------
```

3. Copie o banco de dados da instância de banco de dados MySQL externa para o cluster de banco de dados do Amazon Aurora MySQL usando `mysqldump`. Para bancos de dados muito grandes, pode ser conveniente usar o procedimento descrito em Importing data to an Amazon RDS for MySQL database with reduced downtime no Guia do usuário do Amazon Relational Database Service.

   Para Linux, macOS ou Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -p local_password | mysql \
        --host aurora_cluster_endpoint_address \
        --port 3306 \
        -u RDS_user_name \
        -p RDS_password
```

Para Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -p local_password | mysql ^
        --host aurora_cluster_endpoint_address ^
        --port 3306 ^
        -u RDS_user_name ^
        -p RDS_password
```

> **Note**
>
> Confirme que não há um espaço entre a opção `-p` e a senha inserida.

Use as opções `--host`, `--user (-u)`, `--port` e `-p` no comando `mysql` para especificar o nome do host, o nome do usuário, a porta e a senha para conectar-se ao cluster de banco de dados do Aurora. O nome do host é o nome de DNS do endpoint do cluster de banco de dados do Amazon Aurora, por exemplo, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. Você pode encontrar o valor do endpoint nos detalhes do cluster no Console de gerenciamento do Amazon RDS.

4. Confirme que é possível gravar na instância do banco de dados MySQL novamente:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

Para ter mais informações sobre como fazer backups para usar com a replicação, consulte Backing up a source or replica by making it read only na documentação do MySQL.

5. No Console de gerenciamento do Amazon RDS, adicione o endereço IP do servidor que hospeda o banco de dados MySQL de origem ao grupo de segurança da VPC para o cluster de banco de dados do Amazon Aurora. Para ter mais informações sobre como modificar um grupo de segurança da VPC, consulte Grupos de segurança para a VPC no Guia do usuário da Amazon Virtual Private Cloud.

Você também pode precisar configurar sua rede local para permitir conexões com o endereço IP de seu cluster de banco de dados Amazon Aurora, para que ele possa se comunicar com sua instância MySQL de origem. Para encontrar o endereço IP do cluster de banco de dados do Amazon Aurora, use o comando `host`.

```
host aurora_endpoint_address
```

O nome do host é o nome de DNS do endpoint do cluster de banco de dados Amazon Aurora.

6. Usando o cliente de sua preferência, conecte-se à instância externa do MySQL e crie um usuário do MySQL a ser usado para a replicação. Esta conta é usada unicamente para replicação e deve estar restrita ao seu domínio para melhorar a segurança. Veja um exemplo a seguir.

```
CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
```

7. Para a instância externa do MySQL, conceda privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` para seu usuário de replicação. Por exemplo, para conceder os privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` em todos os bancos de dados para o usuário '`repl_user`' de seu domínio, emita o seguinte comando.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com'
    IDENTIFIED BY 'password';
```

8. Faça um snapshot manual do cluster de banco de dados do Aurora MySQL para ser a réplica de leitura antes de configurar a replicação. Se for necessário restabelecer a replicação com o cluster de banco de dados como uma réplica de leitura, você poderá restaurar o cluster de banco de dados do Aurora MySQL desse snapshot, em vez de precisar importar os dados da instância de banco de dados MySQL para um novo cluster de banco de dados do Aurora MySQL.
9. Faça do cluster de banco Amazon Aurora a réplica. Conecte-se ao cluster de banco de dados do Amazon Aurora como o usuário principal e identifique o banco de dados MySQL de origem como a origem da replicação usando os procedimentos mysql.rds_set_external_master (Aurora MySQL versão 2) ou mysql.rds_set_external_source (Aurora MySQL versão 3) e mysql.rds_start_replication.

   Use o nome do arquivo de log binário e a posição que você determinou na Etapa 2. Veja um exemplo a seguir.

```
For Aurora MySQL version 2:
CALL mysql.rds_set_external_master ('mymasterserver.example.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);

For Aurora MySQL version 3:
CALL mysql.rds_set_external_source ('mymasterserver.example.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);
```

10. No cluster de banco de dados do Amazon Aurora, chame o procedimento mysql.rds_start_replication para iniciar a replicação.

```
CALL mysql.rds_start_replication;
```

Após ter estabelecido a replicação entre sua instância de banco de dados MySQL e seu cluster de banco de dados do Amazon Aurora, você poderá adicionar réplicas do Aurora ao seu cluster de banco de dados do Amazon Aurora. Você poderá então se conectar às réplicas do Aurora para fazer a escalabilidade de leitura de seus dados. Para obter informações sobre como criar uma réplica do Aurora, consulte Adicionar réplicas do Aurora a um cluster de banco de dados.

Otimizar a replicação de logs binários para Aurora MySQL

A seguir, você pode aprender como otimizar a performance da replicação de log binário e solucionar problemas relacionados no Aurora MySQL.

> **Tip**
>
> Esta discussão presume que você esteja familiarizado com o mecanismo de replicação de log binário do MySQL e como funciona. Para obter informações anteriores, consulte Implementação de replicação na documentação do MySQL.

Replicação de logs binários de vários threads

Com a replicação de logs binários de vários threads, um thread SQL faz a leitura de eventos do log de retransmissão e coloca esses eventos em fila para que os threads de operador SQL sejam aplicados. Os threads de trabalho SQL são gerenciados pelo thread coordenador. Os eventos de log binário são aplicados em paralelo sempre que possível. O nível de paralelismo depende de alguns fatores, como versão, parâmetros, design do esquema e características da workload.

A replicação de logs binários de vários threads não é compatível com o Aurora MySQL versão 3, o Aurora MySQL versão 2.12.1 e posterior. Para que uma réplica de vários threads processe eficientemente eventos de log binário em paralelo, você deve configurar a origem para replicação de log binário de vários threads, e a origem deve usar uma versão que inclua as informações de paralelismo nos respectivos arquivos de log binário.

Quando uma instância de banco de dados do Aurora MySQL está configurada para utilizar a replicação de logs binários, por padrão a instância de réplica utiliza a replicação de um único thread para o Aurora MySQL versões anteriores a 3.04. Para habilitar a replicação de vários threads, atualize o parâmetro `replica_parallel_workers` para um valor superior a 1 no seu grupo de parâmetros personalizado.

Para o Aurora MySQL versão 3.04 e posterior, a replicação tem vários threads por padrão, com `replica_parallel_workers` definido como 4. É possível modificar esse parâmetro no grupo de parâmetros personalizado.

Para aumentar a resiliência do banco de dados contra paradas inesperadas, recomendamos habilitar a replicação de GTID na origem e permitir GTIDs na réplica. Para permitir a replicação de GTID, defina `gtid_mode` como `ON_PERMISSIVE` na origem e na réplica. Para obter mais informações sobre a replicação baseada em GTID, consulte Usar a replicação baseada em GTID.

As opções de configuração a seguir ajudam a ajustar a replicação de vários threads. Para obter informações sobre uso, consulte o tópico sobre Opções e variáveis de replicação e registro em log binário, no Guia de referência do MySQL. Para ter mais informações sobre a replicação de vários threads, consulte Improving the Parallel Applier with Writeset-based Dependency Tracking no blog do MySQL.

Os valores ideais dos parâmetros dependem de diversos fatores. Por exemplo, a performance da replicação de log binário é influenciada pelas características da workload do seu banco de dados e pela classe de instância de banco de dados na qual a réplica está sendo executada. Por isso, recomendamos testar completamente todas as alterações nesses parâmetros de configuração antes de aplicar novas configurações de parâmetros a uma instância de produção:

- `binlog_format recommended value` – definir como `ROW`
- `binlog_group_commit_sync_delay`
- `binlog_group_commit_sync_no_delay_count`
- `binlog_transaction_dependency_history_size`
- `binlog_transaction_dependency_tracking`: o valor recomendado é `WRITESET`.

- `replica_preserve_commit_order`
- `replica_parallel_type`: o valor recomendado é `LOGICAL_CLOCK`.
- `replica_parallel_workers`
- `replica_pending_jobs_size_max`
- `transaction_write_set_extraction`: o valor recomendado é `XXHASH64`.

As características do esquema e da workload são fatores que afetam a replicação em paralelo. Os fatores mais comuns são apresentados a seguir.

- Ausência de chaves primárias: o RDS não pode estabelecer dependência de writeset para tabelas sem chaves primárias. Com o formato `ROW`, uma única instrução de várias linhas pode ser realizada com uma única verificação de tabela completa na origem, mas isso gera uma verificação de tabela completa por linha modificada na réplica. A ausência de chaves primárias diminui significativamente o throughput da replicação.
- Existência de chaves estrangeiras: se houver chaves estrangeiras (FKs), o Amazon RDS não poderá usar a dependência de writeset para paralelismo de tabelas com a relação de FK.
- Tamanho das transações: se uma única transação abranger dezenas ou centenas de megabytes ou gigabytes, o thread coordenador e um dos threads de trabalho podem passar um longo tempo processando somente essa transação. Durante esse período, todos os outros threads de trabalho podem permanecer inativos depois de concluírem o processamento das transações anteriores.

No Aurora MySQL versão 3.06 e posterior, é possível melhorar o desempenho de réplicas de log binários ao replicar transações para tabelas grandes com mais de um índice secundário. Esse recurso introduz um grupo de threads para aplicar alterações de índice secundário em paralelo em uma réplica de log binário. O recurso é controlado pelo parâmetro `aurora_binlog_replication_sec_index_parallel_workers` do cluster de banco de dados, que controla o número total de threads paralelos disponíveis para aplicar as alterações do índice secundário. O parâmetro é definido como `0` (desabilitado) por padrão. A habilitação desse recurso não exige a reinicialização da instância. Para habilitar esse recurso, interrompa a replicação contínua, defina o número desejado de threads de operadores paralelos e inicie a replicação novamente.

Otimizar a replicação de log binário

No Aurora MySQL 2.10 e posteriores, o Aurora aplica automaticamente uma otimização conhecida como cache de E/S de binlog à replicação de log binário. Ao armazenar em cache os eventos de

binlog confirmados mais recentemente, essa otimização foi projetada para melhorar a performance do processo de despejo de binlog, limitando o impacto nas transações em primeiro plano na instância de origem do binlog.

> **Note**
>
> Essa memória usada para esse recurso é independente da configuração `binlog_cache` do MySQL.
> Esse recurso não se aplica a instâncias de banco de dados do Aurora que usam as classes de instância `db.t2` e `db.t3`.

Você não precisa ajustar nenhum parâmetro de configuração para ativar essa otimização. Especificamente, se você tiver ajustado o parâmetro de configuração `aurora_binlog_replication_max_yield_seconds` para um valor diferente de zero em versões anteriores do Aurora MySQL, defina-o de volta para zero nas versões atualmente disponíveis.

As variáveis de status `aurora_binlog_io_cache_reads` e `aurora_binlog_io_cache_read_requests` ajudam você a monitorar a frequência com que os dados são lidos do cache de E/S do binlog.

- `aurora_binlog_io_cache_read_requests`: mostra o número de solicitações de leitura de E/S para binlog provenientes do cache.
- `aurora_binlog_io_cache_reads`: mostra o número de leituras de E/S para binlog que recuperam informações do cache.

A consulta SQL a seguir calcula a porcentagem de solicitações de leitura de binlog que aproveitam as informações armazenadas em cache. Nesse caso, quanto mais próxima a proporção for de 100, melhor ela é.

```
mysql> SELECT
  (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_reads')
  / (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_read_requests')
  * 100
  as binlog_io_cache_hit_ratio;
```

```
+---------------------------+
| binlog_io_cache_hit_ratio |
+---------------------------+
|         99.99847949080622 |
+---------------------------+
```

O recurso de cache de E/S de binlog também inclui novas métricas relacionadas aos processos de despejo de binlog. Processos de despejo são os processos criados quando novas réplicas de binlog são conectadas à instância de origem de binlog.

As métricas de processos de despejo são impressas no log do banco de dados a cada 60 segundos com o prefixo `[Dump thread metrics]`. As métricas incluem informações para cada réplica de binlog `Secondary_id`, como `Secondary_uuid`, nome do arquivo de binlog e a posição que cada réplica está lendo. As métricas também incluem `Bytes_behind_primary`, que representa a distância em bytes entre a origem da replicação e a réplica. Essa métrica mede o atraso do processo de E/S da réplica. Essa figura é diferente do lag do processo do aplicador SQL da réplica, que é representado pela métrica `seconds_behind_master` na réplica do binlog. Você pode determinar se as réplicas de binlog estão alcançando a origem ou ficando para trás, verificando se a distância diminui ou aumenta.

Log de retransmissão na memória

No Aurora MySQL versão 3.10 e posterior, o Aurora introduz uma otimização conhecida como log de retransmissão na memória para melhorar o throughput de replicação. Essa otimização melhora o desempenho de E/S do log de retransmissão ao armazenar em cache todo o conteúdo intermediário do log de retransmissão na memória. Desse modo, ela reduz a latência de confirmação ao minimizar as operações de E/S de armazenamento, pois o conteúdo do log de retransmissão pode ser facilmente acessado na memória.

Por padrão, o recurso de log de retransmissão na memória é habilitado automaticamente para cenários de replicação gerenciados pelo Aurora (como implantações azul-verde, replicação do Aurora para o Aurora e réplicas entre regiões) quando a réplica atende a qualquer uma destas configurações:

- Modo de replicação de thread único (replica_parallel_workers = 0)
- Replicação de vários threads com o modo de GTID habilitado:
  - Posicionamento automático habilitado
  - Modo de GTID definido como ON na réplica

- Replicação baseada em arquivos com replica_preserve_commit_order = ON

O recurso de log de retransmissão na memória é compatível com classes de instância maiores que t3.large, mas não está disponível em instâncias do Aurora Serverless. O buffer circular do log de retransmissão tem um tamanho fixo de 128 MB. Para monitorar o consumo de memória desse recurso, você pode executar a seguinte consulta:

```
SELECT event_name, current_alloc FROM sys.memory_global_by_current_bytes WHERE
 event_name = 'memory/sql/relaylog_io_cache';
```

O recurso de log de retransmissão na memória é controlado pelo parâmetro aurora_in_memory_relaylog, que pode ser definido em nível de cluster ou de instância de banco de dados. Você pode habilitar ou desabilitar esse recurso dinamicamente sem reiniciar a instância:

1. Interrompa a replicação em andamento.
2. Defina aurora_in_memory_relaylog como ON (para habilitá-lo) ou OFF (para desabilitá-lo) no grupo de parâmetros
3. Reinicie a replicação.

Exemplo:

```
CALL mysql.rds_stop_replication;
set aurora_in_memory_relaylog to ON to enable or OFF to disable in cluster parameter
 group
CALL mysql.rds_start_replication;
```

Mesmo que aurora_in_memory_relaylog esteja definido como ON, em determinadas condições o recurso de log de retransmissão na memória ainda pode estar desabilitado. Para verificar o status atual do recurso, você pode usar o seguinte comando:

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_status';
```

Se o recurso for desabilitado inesperadamente, você pode identificar o motivo executando:

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_disabled_reason';
```

Esse comando exibe uma mensagem explicando por que o recurso está desabilitado no momento.

Configurar o log binário avançado para Aurora MySQL

O log binário avançado reduz a sobrecarga de performance computacional causada pela ativação do log binário, que pode chegar a até 50% em certos casos. Com o log binário avançado, essa sobrecarga pode ser reduzida para cerca de 13%. Para reduzir a sobrecarga, o log binário avançado grava os registros binários e de transações no armazenamento em paralelo, o que minimiza os dados gravados no momento da confirmação da transação.

O uso do log binário avançado também melhora o tempo de recuperação do banco de dados após reinicializações e failovers em até 99% em comparação com o log binário do MySQL da comunidade. O log binário avançado é compatível com as workloads existentes baseadas em log binário e você interage com ele da mesma forma que interage com o log binário do MySQL da comunidade.

O log binário avançado está disponível no Aurora MySQL versão 3.03.1 e posterior.

Tópicos

- Configuração de parâmetros de log binário avançado
- Outros parâmetros relacionados
- Diferenças entre o log binário avançado e o log binário do MySQL da comunidade
- Métricas do Amazon CloudWatch para o log binário avançado
- Limitações de log binário avançado

Configuração de parâmetros de log binário avançado

Você pode alternar entre o log binário do MySQL da comunidade e o log binário avançado ativando/desativando os parâmetros aprimorados do log binário. Os consumidores de log binário existentes podem continuar lendo e consumindo os arquivos de log binário sem nenhuma lacuna na sequência do arquivo de log binário.

Para ativar o log binário avançado, defina os seguintes parâmetros:

Parameter	Padrão	Descrição
`binlog_format`	–	Defina o `binlog_format` parâmetro no formato de registro em log binário escolhido para ativar o log

Parameter	Padrão	Descrição
		binário avançado. Garanta que `binlog_format parameter` não esteja definido como DESLIGADO. Para obter mais informações, consulte Configurar o registro em log binário do Aurora MySQL.
`aurora_enhanced_binlog`	`0`	Defina o valor desse parâmetro como `1` no grupo de parâmetros do cluster de banco de dados associado ao cluster do Aurora MySQL. Ao alterar o valor desse parâmetro, você deve reinicializar a instância do gravador quando o valor `DBClusterParameterGroupStatus` for exibido como `pending-reboot` .
`binlog_backup`	`1`	Desative esse parâmetro para ativar o log binário avançado. Para fazer isso, defina o valor deste parâmetro como `0`.
`binlog_replication_globaldb`	`1`	Desative esse parâmetro para ativar o log binário avançado. Para fazer isso, defina o valor deste parâmetro como `0`.

> **Important**
>
> É possível desativar os parâmetros `binlog_backup` e `binlog_replication_globaldb` somente ao usar o log binário avançado.

Para desativar o log binário avançado, defina os seguintes parâmetros:

Parameter	Descrição
`aurora_enhanced_binlog`	Defina o valor desse parâmetro como `0` no grupo de parâmetros do cluster de banco de dados associado ao cluster do Aurora MySQL. Sempre que você alterar o valor desse parâmetro, reinicialize a instância do gravador quando o valor `DBClusterParameterGroupStatus` for exibido como `pending-reboot`.
`binlog_backup`	Ative esse parâmetro ao desativar o log binário avançado. Para fazer isso, defina o valor deste parâmetro como 1.
`binlog_replication_globaldb`	Ative esse parâmetro ao desativar o log binário avançado. Para fazer isso, defina o valor deste parâmetro como 1.

Para verificar se o log binário avançado está ativado, use o seguinte comando no cliente MySQL:

```
mysql>show status like 'aurora_enhanced_binlog';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| aurora_enhanced_binlog | ACTIVE |
+------------------------+--------+
1 row in set (0.00 sec)
```

Quando o log binário avançado está ativado, a saída mostra `ACTIVE` para `aurora_enhanced_binlog`.

Outros parâmetros relacionados

Quando você ativa o log binário avançado, os seguintes parâmetros são afetados:

- O parâmetro `max_binlog_size` é visível, mas não pode ser modificado. O valor padrão `134217728` é automaticamente ajustado para `268435456` quando o log binário avançado é ativado.
- Ao contrário do log binário do MySQL da comunidade, o `binlog_checksum` não atua como um parâmetro dinâmico quando o log binário avançado está ativado. Para que a alteração desse parâmetro tenha efeito, você deverá reinicializar manualmente o cluster de banco de dados mesmo quando `ApplyMethod` for `immediate`.
- O valor definido no parâmetro `binlog_order_commits` não tem efeito na ordem das confirmações quando o log binário avançado é ativado. As confirmações são sempre ordenadas sem implicações adicionais na performance.

Diferenças entre o log binário avançado e o log binário do MySQL da comunidade

O log binário avançado interage de forma diferente com clones, backups e o banco de dados global do Aurora quando comparado ao log binário do MySQL da comunidade. Recomendamos que você entenda as seguintes diferenças antes de usar o log binário avançado.

- Os arquivos de log binário avançado do cluster de banco de dados de origem não estão disponíveis em um cluster de banco de dados clonado.
- Os arquivos de log binário avançado não estão incluídos nos backups do Aurora. Portanto, os arquivos de log binário avançado do cluster de banco de dados de origem não estão disponíveis após a restauração de um cluster de banco de dados, apesar de qualquer período de retenção definido nele.
- Quando usados com um banco de dados global do Aurora, os arquivos de log binário avançado do cluster de banco de dados primário não são replicados para o cluster de banco de dados nas regiões secundárias.

Exemplos

Os exemplos a seguir ilustram as diferenças entre o log binário avançado e o log binário do MySQL da comunidade.

Em um cluster de banco de dados restaurado ou clonado

Quando o log binário avançado está ativado, os arquivos de log binário históricos não estão disponíveis no cluster de banco de dados restaurado ou clonado. Depois de uma operação de restauração ou clonagem, se o log binário estiver ativado, o novo cluster de banco de dados começará a gravar sua própria sequência de arquivos de log binário, começando com 1 (mysql-bin-changelog.000001).

Para ativar o log binário avançado após uma operação de restauração ou clonagem, defina os parâmetros necessários do cluster de banco de dados no cluster de banco de dados restaurado ou clonado. Para ter mais informações, consulte Configuração de parâmetros de log binário avançado.

Example Exemplo: operação de clonagem ou de restauração executada quando o log binário avançado está ativado

Cluster de banco de dados de origem:

```
mysql> show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog turned on
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Em um cluster de banco de dados restaurado ou clonado, os arquivos de log binário não são copiados quando o log binário avançado é ativado. Para evitar a descontinuidade nos dados do log binário, os arquivos de log avançado gravados antes de ativar o log binário aprimorado também não estão disponíveis.

```
mysql>show binary logs;
```

```
+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> New sequence of Binlog files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Exemplo: operação de clonagem ou de restauração executada quando o log binário avançado está desativado

Cluster de banco de dados de origem:

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

O log binário aprimorado é desativado depois de `mysql-bin-changelog.000003`. Em um cluster de banco de dados restaurado ou clonado, os arquivos de log binário gravados após a desativação do log binário avançado estão disponíveis.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
```

```
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Em um Amazon Aurora Global Database

Em um Amazon Aurora Global Database, os dados de log binário do cluster de banco de dados primário não são replicados para os clusters de banco de dados secundários. Após um processo de failover entre regiões, os dados do log binário não estão disponíveis no cluster de banco de dados primário recém-promovido. Se o log binário estiver ativado, o cluster de banco de dados recém-promovido iniciará sua própria sequência de arquivos de log binário, começando com 1 (mysql-bin-changelog.000001).

Para ativar o log binário avançado após o failover, defina os parâmetros necessários do cluster de banco de dados no cluster de banco de dados secundário. Para ter mais informações, consulte Configuração de parâmetros de log binário avançado.

Example Exemplo: a operação global de failover do banco de dados é executada quando o log binário avançado é ativado

Cluster de banco de dados primário antigo (antes do failover):

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog enabled
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Novo cluster de banco de dados primário (após o failover):

Os arquivos de log binário não são replicados para regiões secundárias quando o log binário avançado está ativado. Para evitar a descontinuidade nos dados do log binário, os arquivos de log binário gravados antes de ativar o log binário avançado não estão disponíveis.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> Fresh sequence of Binlog
 files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Exemplo: a operação global de failover do banco de dados é executada quando o log binário avançado é desativado

Cluster de banco de dados de origem:

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Cluster de banco de dados restaurado ou clonado:

O log binário aprimorado é desativado depois de `mysql-bin-changelog.000003`. Os arquivos de log binário gravados após a desativação do log binário avançado são replicados e estão disponíveis no cluster de banco de dados recém-promovido.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
3 rows in set (0.00 sec)
```

Métricas do Amazon CloudWatch para o log binário avançado

As métricas a seguir do Amazon CloudWatch são publicadas somente quando o binlog avançado está ativado.

Métrica do CloudWatch	Descrição	Unidades
ChangeLogBytesUsed	A quantidade de armazenamento usada pelo log binário avançado.	Bytes
ChangeLogReadIOPs	O número de operações de E/S de leitura executadas no log binário avançado em um intervalo de cinco minutos.	Contagem a cada 5 minutos
ChangeLogWriteIOPs	O número de operações de E/S de disco de gravação executadas no log binário avançado em um intervalo de cinco minutos.	Contagem a cada 5 minutos

Limitações de log binário avançado

As limitações a seguir se aplicam aos clusters de banco de dados Amazon Aurora quando o log binário avançado é ativado.

- O log binário avançado é aceito apenas no Aurora MySQL versão 3.03.1 e posterior.
- Os arquivos de log binário avançado gravados no cluster de banco de dados primário não são copiados para os clusters de banco de dados clonados ou restaurados.
- Quando usados com o Amazon Aurora Global Database, os arquivos de log binário avançado do cluster de banco de dados primário não são replicados para os clusters de banco de dados secundários. Portanto, após o processo de failover, os dados históricos do log binário não estão disponíveis no novo cluster de banco de dados primário.
- Os seguintes parâmetros de configuração do log binário são ignorados:
  - `binlog_group_commit_sync_delay`
  - `binlog_group_commit_sync_no_delay_count`
  - `binlog_max_flush_queue_time`
- Você não pode eliminar ou renomear uma tabela corrompida em um banco de dados. Para eliminar essas tabelas, você pode entrar em contato com o Suporte.
- O cache de E/S do log binário é desabilitado quando o registro binário avançado é ativado. Para ter mais informações, consulte Otimizar a replicação de logs binários para Aurora MySQL.

> **Note**
>
> O log binário avançado fornece melhorias de performance de leitura semelhantes ao cache de E/S do log binário e avanços melhores na performance de gravação.

- O recurso de retrocesso não é compatível. O log binário avançado não pode ser ativado em um cluster de banco de dados nas seguintes condições:
  - Cluster de banco de dados com o recurso de retrocesso atualmente habilitado.
  - O cluster de banco de dados com o recurso de retrocesso era habilitado anteriormente, mas agora está desabilitado.
  - Cluster de banco de dados restaurado com base em um cluster de banco de dados de origem ou de um snapshot com o recurso de retrocesso habilitado.

Usar a replicação baseada em GTID

O conteúdo a seguir explica como usar identificadores de transações globais (GTIDs) com a replicação de logs binários (binlog) entre um cluster do Aurora MySQL e uma fonte externa.

> **Note**
>
> Para o Aurora, você só pode usar esse recurso com clusters do Aurora MySQL que usem replicação de logs binários para ou de um banco de dados MySQL externo. O outro banco de dados pode ser uma instância do Amazon RDS MySQL, um banco de dados MySQL on-premises ou um cluster de banco de dados do Aurora em uma Região da AWS diferente. Para saber mais sobre como configurar esse tipo de replicação, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Se você usa a replicação de logs binários e não conhece a replicação baseada em GTID com o MySQL, consulte Replication with global transaction identifiers na documentação do MySQL.

A replicação baseada em GTID é compatível com o Aurora MySQL versões 2 e 3.

Tópicos

- Visão geral dos identificadores de transações globais (GTIDs)
- Parâmetros para replicação baseada em GTID
- Ativar a replicação baseada em GTID para um cluster do Aurora MySQL
- Desabilitação da replicação baseada em GTID para um cluster de banco de dados do Aurora MySQL

Visão geral dos identificadores de transações globais (GTIDs)

GTIDs são identificadores exclusivos gerados para transações MySQL confirmadas. Você pode usar GTIDs para tornar a replicação de log binário mais simples e fácil de solucionar.

> **Note**
>
> Quando o Aurora sincroniza dados entre as instâncias de banco de dados em um cluster, esse mecanismo de replicação não envolve o log binário (binlog). Para o Aurora MySQL, a replicação baseada em GTID somente é aplicável quando você também usa a replicação de log binário para replicar dentro ou fora de um cluster de banco de dados do Aurora MySQL a partir de um banco de dados externo compatível com MySQL.

O MySQL usa dois tipos diferentes de transações para a replicação de log binário:

- Transações GTID – Transações identificadas por um GTID.
- Transações anônimas – transações que não têm um GTID atribuído.

Em uma configuração de replicação, GTIDs são exclusivos entre todas as instâncias de banco de dados. Os GTIDs simplificam a configuração da replicação porque ao usá-los você não precisa consultar posições de arquivo de log. Os GTIDs também facilitam o rastreamento de transações replicadas e a determinação da consistência da instância de origem e de réplicas.

Normalmente, você usa a replicação baseada em GTID com o Aurora ao replicar de um banco de dados externo compatível com MySQL para um cluster do Aurora. É possível configurar essa configuração de replicação como parte de uma migração de um banco de dados no local ou do Amazon RDS para o Aurora MySQL. Se o banco de dados externo já usa GTIDs, a ativação da replicação baseada em GTID para o cluster do Aurora simplificará o processo de replicação.

Você configura a replicação baseada em GTID para um cluster do Aurora MySQL definindo primeiro os parâmetros de configuração relevantes em um grupo de parâmetros de cluster de banco de dados. Em seguida, você associa esse grupo de parâmetros ao cluster.

Parâmetros para replicação baseada em GTID

Use os parâmetros a seguir para configurar a replicação baseada em GTID.

Parâmetro	Valores válidos	Descrição
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF` especifica que novas transações são anônimas (ou seja, não têm GTIDs) e que uma transação deve ser anônima para ser replicada. `OFF_PERMISSIVE` especifica que novas transações são anônimas, mas todas podem ser replicadas. `ON_PERMISSIVE` especifica que novas transações são GTID, mas todas podem ser replicadas.

Parâmetro	Valores válidos	Descrição
		`ON` especifica que novas transações são GTID e que uma transação deve ser GTID para ser replicada.
`enforce_gtid_consistency`	`OFF`, `ON`, `WARN`	`OFF` permite que as transações violem a consistência GTID. `ON` evita que as transações violem a consistência GTID. `WARN` permite que as transações violem a consistência GTID, mas gera um aviso quando ocorre uma violação.

> **Note**
>
> No Console de gerenciamento da AWS, o parâmetro `gtid_mode` aparece como `gtid-mode`.

Para a replicação baseada em GTID, use essas configurações para o grupo de parâmetros de cluster de banco de dados do seu cluster de banco de dados do Aurora MySQL:

- `ON` e `ON_PERMISSIVE` somente são aplicáveis à replicação de saída de um cluster do Aurora MySQL. Esses dois valores fazem com que o seu cluster de banco de dados do Aurora use GTIDs para transações que são replicadas para um banco de dados externo. O `ON` requer que o banco de dados externo também use a replicação baseada em GTID. O `ON_PERMISSIVE` torna a replicação baseada em GTID opcional no banco de dados externo.
- `OFF_PERMISSIVE`, se definido, significa que o cluster de banco de dados do Aurora pode aceitar a replicação de entrada de um banco de dados externo. Ele pode fazer isso independentemente de o banco de dados externo usar a replicação baseada em GTID ou não.
- `OFF`, se definido, significa que seu cluster de banco de dados do Aurora apenas aceita a replicação de entrada de bancos de dados externos que não usam a replicação baseada em GTID.

> **Tip**
>
> A replicação de entrada é o cenário de replicação de log binário mais comum para clusters do Aurora MySQL. Para a replicação de entrada, recomendamos definir o modo de GTID como `OFF_PERMISSIVE`. Essa configuração permite a replicação de entrada de bancos de dados externos, independentemente das configurações de GTID na origem de replicação.

Para obter mais informações sobre parameter groups, consulte Grupos de parâmetros para Amazon Aurora.

Ativar a replicação baseada em GTID para um cluster do Aurora MySQL

Quando a replicação baseada em GTID está habilitada para um cluster de banco de dados do Aurora MySQL, as configurações de GTID aplicam-se à replicação de log binário de entrada e saída.

Para habilitar a replicação baseada em GTID para um cluster do Aurora MySQL

1. Crie ou edite um grupo de parâmetros de cluster de banco de dados usando as seguintes configurações de parâmetros:
   - `gtid_mode` – `ON` ou `ON_PERMISSIVE`
   - `enforce_gtid_consistency` – `ON`
2. Associe o grupo de parâmetros de cluster de banco de dados ao cluster do Aurora MySQL. Para isso, siga o procedimento em Grupos de parâmetros para Amazon Aurora.
3. (Opcional) Especifique como atribuir GTIDs a transações que não os incluem. Para isso, chame o procedimento armazenado em mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versão 3).

Desabilitação da replicação baseada em GTID para um cluster de banco de dados do Aurora MySQL

Você pode desabilitar a replicação baseada em GTID para um cluster de banco de dados do Aurora MySQL. Isso significa que o cluster do Aurora não pode realizar a replicação de log binário de entrada ou saída com bancos de dados externos que usam a replicação baseada em GTID.

> **Note**
>
> No procedimento a seguir, réplica de leitura significa o destino de replicação em uma configuração do Aurora com replicação de log binário para ou de um banco de dados externo. O termo não significa as instâncias de banco de dados de réplica somente leitura do Aurora. Por exemplo, quando um cluster do Aurora aceita a replicação de entrada de uma origem externa, a instância primária do Aurora atua como a réplica de leitura para replicação de log binário.

Para obter mais detalhes sobre os procedimentos armazenados mencionados nesta seção, consulte Referência de procedimentos armazenados do Aurora MySQL.

Para desabilitar a replicação baseada em GTID para um cluster de banco de dados do Aurora MySQL

1. Nas réplicas do Aurora, execute o seguinte procedimento:

   Para a versão 3

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   Para a versão 2

   ```
   CALL mysql.rds_set_master_auto_position(0);
   ```

2. Redefina o `gtid_mode` como `ON_PERMISSIVE`.

   a. Certifique-se que o grupo de parâmetros de cluster de banco de dados associado ao cluster do Aurora MySQL tenha `gtid_mode` definido como `ON_PERMISSIVE`.

      Para obter mais informações sobre a definição de parâmetros de configuração usando grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.

   b. Reinicie o cluster de banco de dados do Aurora MySQL.

3. Redefina o `gtid_mode` como `OFF_PERMISSIVE`.

   a. Certifique-se que o grupo de parâmetros de cluster de banco de dados associado ao cluster do Aurora MySQL tenha `gtid_mode` definido como `OFF_PERMISSIVE`.

b. Reinicie o cluster de banco de dados do Aurora MySQL.

4. Aguarde a aplicação de todas as transações de GTID na instância primária do Aurora. Para conferir se elas foram aplicadas, siga estas etapas:

   a. Na instância de primária do Aurora, execute o comando `SHOW MASTER STATUS`.

      Sua saída deve ser semelhante ao resultado a seguir.

```
File                        Position
-----------------------------------
mysql-bin-changelog.000031      107
-----------------------------------
```

      Observe o arquivo e posicione na saída.

   b. Em cada réplica de leitura, use as informações de arquivo e posição de sua instância de origem na etapa anterior para executar a seguinte consulta:

      Para a versão 3

```
SELECT SOURCE_POS_WAIT('file', position);
```

      Para a versão 2

```
SELECT MASTER_POS_WAIT('file', position);
```

      Por exemplo, caso o nome do arquivo seja `mysql-bin-changelog.000031` e a posição seja `107`, execute a seguinte declaração:

      Para a versão 3

```
SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
```

      Para a versão 2

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

5. Redefina os parâmetros de GTID para desabilitar a replicação baseada em GTID.

a. Verifique se o grupo de parâmetros de cluster de banco de dados associado ao cluster do Aurora MySQL tem as seguintes configurações de parâmetros:

   - `gtid_mode` – `OFF`
   - `enforce_gtid_consistency` – `OFF`

b. Reinicie o cluster de banco de dados do Aurora MySQL.

Usar o encaminhamento de gravação local em um cluster de banco de dados do Amazon Aurora MySQL

O encaminhamento de gravação local (no cluster) permite que as aplicações emitam transações de leitura/gravação diretamente em uma réplica do Aurora. Essas transações são então encaminhadas para a instância de banco de dados do gravador para serem confirmadas. Você pode usar o encaminhamento de gravação local quando as aplicações exigirem consistência de leitura após gravação, que é a capacidade de ler a última gravação em uma transação.

As réplicas de leitura recebem atualizações de forma assíncrona do gravador. Sem o encaminhamento de gravação, você precisa realizar todas as leituras que exigem consistência de leitura após gravação na instância de banco de dados do gravador. Ou é necessário desenvolver uma lógica de aplicação personalizada complexa para aproveitar as vantagens de várias réplicas de leitura para escalabilidade. As aplicações devem dividir todo o tráfego de leitura e gravação, mantendo dois conjuntos de conexões de banco de dados para enviar o tráfego ao endpoint correto. Essas despesas indiretas de desenvolvimento complicam o design da aplicação quando as consultas fazem parte de uma única sessão lógica, ou transação, na aplicação. Além disso, como o atraso na replicação pode diferir entre as réplicas de leitura, é difícil obter consistência de leitura global em todas as instâncias do banco de dados.

O encaminhamento de gravação evita a necessidade de dividir essas transações ou de enviá-las exclusivamente ao gravador, o que simplifica o desenvolvimento de aplicações. Esse novo recurso facilita a obtenção de escala de leitura para workloads que precisam ler a última gravação em uma transação e não são sensíveis à latência de gravação.

O encaminhamento de gravação local é diferente do encaminhamento de gravação global, que encaminha gravações de um cluster de banco de dados secundário para o cluster de banco de dados primário em um banco de dados global do Aurora. Você pode usar o encaminhamento de gravação local em um cluster de banco de dados que faz parte de um banco de dados global do Aurora. Para ter mais informações, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

O encaminhamento de gravação local exige o Aurora MySQL versão 3.04 ou posterior.

Tópicos

- Habilitar o encaminhamento de gravação local
- Conferir se um cluster de banco de dados tem o encaminhamento de gravação habilitado
- Compatibilidade SQL e de aplicações com o encaminhamento de gravação

- Níveis de isolamento para encaminhamento de gravação
- Consistência de leitura para encaminhamento de gravação
- Executar instruções de várias partes com o encaminhamento de gravação
- Transações com o encaminhamento de gravação
- Parâmetros de configuração para o encaminhamento de gravação
- Métricas do Amazon CloudWatch e variáveis de status do Aurora MySQL para encaminhamento de gravação
- Identificar transações e consultas encaminhadas

Habilitar o encaminhamento de gravação local

Por padrão, o encaminhamento de gravação local não está habilitado para clusters de banco de dados do Aurora MySQL. Você habilita o encaminhamento de gravação local no cluster, não na instância.

> **Important**
>
> Você também pode habilitar o encaminhamento de gravação local para réplicas de leitura entre regiões que usam registro em log binário, mas as operações de gravação não são encaminhadas à Região da AWS de origem. Elas são encaminhadas à instância de banco de dados do gravador do cluster de réplica de leitura do binlog.
> Use esse método somente se você tiver um caso de uso para gravar na réplica de leitura do binlog na Região da AWS secundária. Caso contrário, você pode acabar enfrentando um cenário de “cérebro dividido”, caso em que os conjuntos de dados replicados são inconsistentes entre si.
> Recomendamos usar o encaminhamento de gravação global com bancos de dados globais, em vez do encaminhamento de gravação local em réplicas de leitura entre regiões, a menos que seja absolutamente necessário. Para ter mais informações, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Console

Usando o Console de gerenciamento da AWS, marque a caixa de seleção Ativar o encaminhamento de gravação local abaixo de Encaminhamento de gravação de réplica de leitura quando você criar ou modificar um cluster de banco de dados.

AWS CLI

Para habilitar o encaminhamento de gravação com a AWS CLI, use a opção `--enable-local-write-forwarding`. Essa opção funciona quando você cria um cluster de banco de dados usando o comando `create-db-cluster`. Ela também funciona quando você modifica um cluster de banco de dados usando o comando `modify-db-cluster`. É possível desabilitar o encaminhamento de gravação usando a opção `--no-enable-local-write-forwarding` com esses mesmos comandos da CLI.

O exemplo a seguir cria um cluster de banco de dados do Aurora MySQL com encaminhamento de gravação habilitado.

```
aws rds create-db-cluster \
  --db-cluster-identifier write-forwarding-test-cluster \
  --enable-local-write-forwarding \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.04.0 \
  --master-username myuser \
  --master-user-password mypassword \
  --backup-retention 1
```

Depois, você cria instâncias de banco de dados do gravador e do leitor para poder usar o encaminhamento de gravação. Para ter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.

API do RDS

Para habilitar o encaminhamento de gravação usando a API do Amazon RDS, defina o parâmetro `EnableLocalWriteForwarding` como `true`. Esse parâmetro funciona quando você cria um cluster de banco de dados usando a operação `CreateDBCluster`. Ele também funciona quando você modifica um cluster de banco de dados usando a operação `ModifyDBCluster`. É possível desabilitar o encaminhamento de gravação definindo o parâmetro `EnableLocalWriteForwarding` como `false`.

Habilitar o encaminhamento de gravação para sessões de banco de dados

`aurora_replica_read_consistency` é um parâmetro de banco de dados e de cluster de banco de dados que permite o encaminhamento de gravação. Você pode especificar `EVENTUAL`, `SESSION` ou `GLOBAL` para o nível de consistência de leitura. Para saber mais sobre os níveis de consistência, consulte Consistência de leitura para encaminhamento de gravação.

As seguintes regras se aplicam a esse parâmetro:

- O valor padrão é '' (null).
- O encaminhamento de gravação estará disponível apenas se você definir `aurora_replica_read_consistency` como `EVENTUAL`, `SESSION` ou `GLOBAL`. Esse parâmetro é relevante somente em instâncias do leitor de clusters de banco de dados que têm o encaminhamento de gravação habilitado.
- Não é possível definir esse parâmetro (quando vazio) ou cancelar a definição (quando já estiver definido) em uma transação de várias declarações. Você pode alterá-lo de um valor válido para outro valor válido durante essa transação, mas não recomendamos essa ação.

Conferir se um cluster de banco de dados tem o encaminhamento de gravação habilitado

Para determinar se você pode usar o encaminhamento de gravação em um cluster de banco de dados, confirme se o cluster tem o atributo `LocalWriteForwardingStatus` definido como `enabled`.

No Console de gerenciamento da AWS, na guia Configuração da página de detalhes do cluster, você vê o status Habilitado para Encaminhamento de gravação de réplica de leitura local.

Para ver o status da configuração de encaminhamento de gravação para todos os clusters, execute o comando da AWS CLI a seguir.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
    {
        "LocalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-1"
    },
    {
        "LocalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-2"
    },
```

```
    {
        "LocalWriteForwardingStatus": "requested",
        "DBClusterIdentifier": "test-global-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "null",
        "DBClusterIdentifier": "aurora-mysql-v2-cluster"
    }
]
```

Um cluster de banco de dados pode ter os seguintes valores para `LocalWriteForwardingStatus`:

- `disabled`: o encaminhamento de gravação está desabilitado.
- `disabling`: o encaminhamento de gravação está sendo desabilitado.
- `enabled`: o encaminhamento de gravação está habilitado.
- `enabling`: o encaminhamento de gravação está sendo habilitado.
- `null`: o encaminhamento de gravação não está disponível para esse cluster de banco de dados.
- `requested`: o encaminhamento de gravação foi solicitado mas ainda não está ativo.

Compatibilidade SQL e de aplicações com o encaminhamento de gravação

É possível usar os seguintes tipos de instruções SQL com o encaminhamento de gravação:

- Instruções de linguagem de manipulação de dados (DML), como `INSERT`, `DELETE` e `UPDATE`. Existem algumas restrições com relação às propriedades dessas instruções que podem ser usadas com o encaminhamento de gravação, conforme descrito a seguir.
- Instruções `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Instruções `PREPARE` e `EXECUTE`.

Certas declarações não são permitidas ou podem gerar resultados obsoletos ao serem usadas em um cluster de banco de dados com o encaminhamento de gravação. Além disso, funções definidas pelo usuário e procedimentos definidos pelo usuário não são compatíveis. Por isso, normalmente a configuração `EnableLocalWriteForwarding` está desabilitada para clusters de banco de dados. Antes de habilitá-la, confira se o código da aplicação não é afetado por nenhuma dessas restrições.

As restrições a seguir se aplicam às instruções SQL usadas com o encaminhamento de gravação. Em alguns casos, é possível usar as declarações em clusters de banco de dados com o encaminhamento de gravação habilitado. Essa abordagem funcionará se o encaminhamento de gravação não for habilitado na sessão pelo parâmetro de configuração `aurora_replica_read_consistency`. Se você tentar usar uma declaração quando ela não for permitida devido ao encaminhamento de gravação, será exibida uma mensagem de erro semelhante à seguinte:

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Linguagem de definição de dados (DDL)

Conecte-se à instância de banco de dados do gravador para executar declarações DDL. Não é possível executá-las em instâncias de banco de dados do leitor.

Atualizar uma tabela permanente usando dados de uma tabela temporária

Você pode usar tabelas temporárias em clusters de banco de dados com o encaminhamento de gravação habilitado. No entanto, não é possível usar uma instrução DML para modificar uma tabela permanente se a instrução se referir a uma tabela temporária. Por exemplo, não é possível usar uma instrução `INSERT ... SELECT` que usa os dados de uma tabela temporária.

Transações XA

Não é possível usar as declarações a seguir em um cluster de banco de dados quando o encaminhamento de gravação está habilitado na sessão. Essas declarações podem ser usadas em clusters de banco de dados que não tenham o encaminhamento de gravação habilitado ou em sessões em que a configuração `aurora_replica_read_consistency` está vazia. Antes de habilitar o encaminhamento de gravação em uma sessão, confira se o código usa essas declarações.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instruções LOAD para tabelas permanentes

Não é possível usar as declarações a seguir em um cluster de banco de dados com o encaminhamento de gravação habilitado.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Instruções de plugin

Não é possível usar as declarações a seguir em um cluster de banco de dados com o encaminhamento de gravação habilitado.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Instruções SAVEPOINT

Não é possível usar as declarações a seguir em um cluster de banco de dados quando o encaminhamento de gravação está habilitado na sessão. Você pode usar essas declarações em clusters de banco de dados que não tenham o encaminhamento de gravação habilitado ou em sessões em que a configuração `aurora_replica_read_consistency` está em branco. Confira se o código usa essas declarações antes de habilitar o encaminhamento de gravação em uma sessão.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Níveis de isolamento para encaminhamento de gravação

Em sessões que usam o encaminhamento de gravação, só é possível usar o nível de isolamento `REPEATABLE READ`. Embora também seja possível usar o nível de isolamento `READ COMMITTED` com réplicas do Aurora, esse nível de isolamento não funciona com o encaminhamento de gravação. Para obter informações sobre os níveis de isolamento `REPEATABLE READ` e `READ COMMITTED`, consulte Níveis de isolamento do Aurora MySQL.

Consistência de leitura para encaminhamento de gravação

É possível controlar o grau de consistência de leitura em um cluster de banco de dados. O nível de consistência de leitura determina quanto o cluster de banco de dados espera antes de cada operação de leitura para garantir que algumas ou todas as alterações sejam replicadas do gravador. Você pode ajustar o nível de consistência de leitura para garantir que todas as operações de gravação encaminhadas da sessão estejam visíveis no cluster de banco de dados antes de qualquer consulta subsequente. Você também pode usar essa configuração para garantir que as consultas no cluster de banco de dados sempre vejam as atualizações mais recentes do gravador. Essa configuração também se aplica às consultas enviadas por outras sessões ou outros clusters. Para especificar esse tipo de comportamento para a aplicação, escolha um valor para o parâmetro de banco de dados ou o parâmetro de cluster de banco de dados `aurora_replica_read_consistency`.

> **Important**
>
> Sempre defina o parâmetro de banco de dados ou o parâmetro de cluster de banco de dados `aurora_replica_read_consistency` quando você desejar encaminhar gravações. Se você não fizer isso, o Aurora não encaminhará as gravações. Esse parâmetro tem um valor vazio por padrão, então escolha um valor específico quando você usar esse parâmetro. O parâmetro `aurora_replica_read_consistency` afeta somente clusters de banco de dados ou instâncias com o encaminhamento de gravação habilitado.

À medida que você aumenta o nível de consistência, mais tempo a aplicação aguarda a propagação das alterações entre as instâncias de banco de dados. Você pode escolher um contrapeso entre o tempo de resposta rápido e a garantia de que as alterações feitas em outras instâncias de banco de dados estejam totalmente disponíveis antes da execução das consultas.

Você pode especificar os seguintes valores para o parâmetro `aurora_replica_read_consistency`:

- `EVENTUAL`: os resultados das operações de gravação na mesma sessão ficam visíveis apenas quando a operação de gravação é executada na instância de banco de dados do gravador. A consulta não espera que os resultados atualizados estejam disponíveis. Assim, ela pode recuperar os dados mais antigos ou os dados atualizados, dependendo do tempo das declarações e da quantidade de atraso da replicação. Essa é a mesma consistência dos clusters de banco de dados do Aurora MySQL que não usam encaminhamento de gravação.

- `SESSION`: todas as consultas que usam o encaminhamento de gravação veem os resultados de todas as alterações feitas nessa sessão. As alterações são visíveis independentemente de a transação ser confirmada. Se necessário, a consulta aguardará a replicação dos resultados das operações de gravação encaminhadas.
- `GLOBAL`: uma sessão vê todas as alterações confirmadas em todas as sessões e instâncias no cluster de banco de dados. Cada consulta pode aguardar por um período que varia de acordo com a quantidade de atraso da sessão. A consulta prossegue quando o cluster de banco de dados está atualizado com todos os dados confirmados do gravador, a partir do momento em que a consulta foi iniciada.

Para receber informações sobre os parâmetros de configuração envolvidos no encaminhamento de gravação, consulte Parâmetros de configuração para o encaminhamento de gravação.

> **Note**
>
> Você também pode usar `aurora_replica_read_consistency` como uma variável de sessão; por exemplo:
>
> ```
> mysql> set aurora_replica_read_consistency = 'session';
> ```

Exemplos de uso do encaminhamento de gravação

O exemplo a seguir mostra os efeitos do parâmetro `aurora_replica_read_consistency` na execução de instruções `INSERT` seguidas por declarações `SELECT`. Os resultados podem diferir dependendo do valor de `aurora_replica_read_consistency` e do horário das declarações.

Para obter maior consistência, você pode esperar brevemente antes de emitir a instrução `SELECT`. Ou Aurora pode esperar automaticamente até que os resultados terminem a replicação antes de prosseguir `SELECT`.

Para receber informações sobre como definir parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Example com **`aurora_replica_read_consistency`** definido como **`EVENTUAL`**

A execução de uma declaração `INSERT`, seguida imediatamente de uma declaração `SELECT`, retorna um valor para `COUNT(*)` com o número de linhas antes de a nova linha

ser inserida. Executar SELECT novamente, pouco tempo depois, retornará a contagem de linhas atualizada. As declarações SELECT não aguardam.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Example com **aurora_replica_read_consistency** definido como **SESSION**

Uma declaração SELECT imediatamente após INSERT aguarda até que as alterações da declaração INSERT fiquem visíveis. Declarações SELECT subsequentes não aguardam.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)

mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
```

```
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Com a configuração de consistência de leitura ainda definida como `SESSION`, a introdução de uma breve espera após a execução de uma instrução `INSERT` torna a contagem de linhas atualizada disponível no momento em que a próxima instrução `SELECT` é executada.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Example com **`aurora_replica_read_consistency`** definido como **`GLOBAL`**

Antes de executar a consulta, cada declaração `SELECT` aguarda que todas as alterações de dados desde a hora de início da declaração fiquem visíveis. O tempo de espera para cada declaração `SELECT` varia, dependendo da quantidade de atraso na replicação.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
```

```
+----------+
1 row in set (0.75 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Executar instruções de várias partes com o encaminhamento de gravação

Uma declaração DML pode consistir em várias partes, como uma declaração `INSERT ... SELECT` ou `DELETE ... WHERE`. Nesse caso, a declaração inteira é encaminhada para a instância de banco de dados do gravador e é executada nela.

Transações com o encaminhamento de gravação

Se o modo de acesso à transação estiver definido como somente leitura, o encaminhamento de gravação não será usado. É possível especificar o modo de acesso da transação usando a instrução `SET TRANSACTION` ou a instrução `START TRANSACTION`. Você também pode especificar o modo de acesso da transação alterando o valor da sessão transaction_read_only. Você só pode alterar esse valor de sessão enquanto estiver conectado a um cluster de banco de dados que tenha o encaminhamento de gravação habilitado.

Se uma transação de longa duração não emitir nenhuma instrução por um período substancial, ela poderá exceder o tempo limite ocioso. Este período tem um padrão de um minuto. Você pode definir o parâmetro `aurora_fwd_writer_idle_timeout` para aumentá-lo em até um dia. Uma transação que excede o tempo limite ocioso é cancelada pela instância do gravador. A instrução subsequente enviada recebe um erro de tempo limite. Depois, o Aurora reverte a transação.

Esse tipo de erro pode ocorrer em outros casos quando o encaminhamento de gravação fica indisponível. Por exemplo, o Aurora cancelará todas as transações que usam o encaminhamento de gravação se você reiniciar o cluster de banco de dados ou se desabilitar o encaminhamento de gravação.

Quando uma instância do gravador em um cluster que use o encaminhamento de gravação local é reiniciada, todas as transações e consultas ativas e encaminhadas nas instâncias do leitor que usam o encaminhamento de gravação local são fechadas automaticamente. Quando a instância do gravador estiver novamente disponível, você poderá repetir essas transações.

Parâmetros de configuração para o encaminhamento de gravação

Os grupos de parâmetros de banco de dados do Aurora incluem configurações para o atributo de encaminhamento de gravação. Detalhes sobre esses parâmetros são resumidos na tabela a seguir, com notas de uso após a tabela.

Parameter	Escopo	Type	Valor padrão	Valores válidos
`aurora_fwd_writer_idle_timeout`	Cluster	Inteiro não assinado	60	1–86.400
`aurora_fwd_writer_max_connections_pct`	Cluster	inteiro longo não assinado	10	0–90
`aurora_replica_read_consistency`	Cluster ou instância	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Para controlar as solicitações de gravação recebidas, use estas configurações:

- `aurora_fwd_writer_idle_timeout`: o número de segundos que a instância de banco de dados do gravador espera pela atividade em uma conexão que é encaminhada de uma instância do leitor antes de fechá-la. Se a sessão permanecer ociosa além desse período, o Aurora cancelará a sessão.
- `aurora_fwd_writer_max_connections_pct`: o limite máximo em conexões de banco de dados que pode ser usado em uma instância de banco de dados do gravador

para lidar com consultas encaminhadas de instâncias do leitor. Ele é expresso como uma porcentagem da configuração `max_connections` para o gravador. Por exemplo, se `max_connections` for 800 e `aurora_fwd_master_max_connections_pct` ou `aurora_fwd_writer_max_connections_pct` for 10, o gravador permitirá um máximo de 80 sessões encaminhadas simultâneas. Essas conexões vêm do mesmo grupo de conexões gerenciado pela configuração `max_connections`.

Essa configuração se aplica somente ao gravador quando o encaminhamento de gravação está habilitado. Se você diminuir o valor, as conexões existentes não serão afetadas. O Aurora leva o novo valor da configuração em conta ao tentar criar uma conexão por meio de um cluster de banco de dados. O valor padrão é 10, representando 10% do valor `max_connections`.

> **Note**
>
> Como `aurora_fwd_writer_idle_timeout` e `aurora_fwd_writer_max_connections_pct` sã
> parâmetros de cluster de banco de dados, todas as instâncias de banco de dados em cada cluster têm os mesmos valores para esses parâmetros.

Para obter mais informações sobre o `aurora_replica_read_consistency`, consulte Consistência de leitura para encaminhamento de gravação.

Para obter mais informações sobre grupos de parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Métricas do Amazon CloudWatch e variáveis de status do Aurora MySQL para encaminhamento de gravação

As métricas do Amazon CloudWatch e as variáveis de status do Aurora MySQL a seguir se aplicam quando você usa o encaminhamento de gravação para Aurora MySQL. Para ter mais informações sobre métricas para instâncias de banco de dados de leitor e de gravador do Aurora MySQL, consulte os tópicos a seguir.

Tópicos

- Métricas para encaminhamento de gravação para instâncias de banco de dados de gravador do Aurora MySQL

- Métricas para encaminhamento de gravação para instâncias de banco de dados de leitor do Aurora MySQL

Métricas para encaminhamento de gravação para instâncias de banco de dados de gravador do Aurora MySQL

As métricas do Amazon CloudWatch a seguir se aplicam quando você usa o encaminhamento de gravação em um ou mais clusters de banco de dados. Todas essas métricas são medidas na instância de banco de dados.

métrica do cloudwatch	Unidade	Descrição
`ForwardingWriterDMLLatency`	Milissegundos	Tempo médio para processar cada declaração DML encaminhada na instância de banco de dados de gravador. Não inclui o tempo para o cluster de banco de dados encaminhar a solicitação de gravação nem o tempo para replicar as alterações de volta no gravador.
`ForwardingWriterDMLThroughput`	Contagem por segundo	Número de instruções DML encaminhadas processadas a cada segundo por essa instância de banco de dados de gravador.
`ForwardingWriterOpenSessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador.

As variáveis de status do Aurora MySQL a seguir se aplicam quando você usa o encaminhamento de gravação em um ou mais clusters de banco de dados. Todas essas variáveis de status são medidas na instância de banco de dados do gravador.

Variável de status do Aurora MySQL	Unidade	Descrição
`Aurora_fwd_writer_dml_stmt_count`	Contagem	Número total de instruções DML encaminhadas para essa instância de banco de dados de gravador.
`Aurora_fwd_writer_dml_stmt_duration`	Microssegundos	Duração total das instruções DML encaminhadas para essa instância de banco de dados de gravador.
`Aurora_fwd_writer_open_sessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador.
`Aurora_fwd_writer_select_stmt_count`	Contagem	Número total de instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador.
`Aurora_fwd_writer_select_stmt_duration`	Microssegundos	Duração total das instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador.

Métricas para encaminhamento de gravação para instâncias de banco de dados de leitor do Aurora MySQL

As métricas do CloudWatch a seguir são medidas em cada instância de banco de dados do leitor em um cluster de banco de dados com o encaminhamento de gravação habilitado.

métrica do cloudwatch	Unidade	Descrição
`ForwardingReplicaDMLLatency`	Milissegundos	Tempo médio de resposta de DMLs encaminhadas na réplica.
`ForwardingReplicaDMLThroughput`	Contagem por segundo	Número de instruções DML encaminhadas processadas por segundo.
`ForwardingReplicaOpenSessions`	Contagem	O número de sessões que estão usando o encaminhamento de gravação em uma instância de banco de dados do leitor.
`ForwardingReplicaReadWaitLatency`	Milissegundos	Tempo médio de espera que uma declaração `SELECT` em uma instância de banco de dados do leitor aguarda para alcançar o gravador. O grau em que a instância de banco de dados de leitor aguarda antes de processar uma consulta depende da configuração `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Contagem por segundo	Número total de instruções `SELECT` processadas a cada segundo em todas as sessões que estão encaminhando gravações.
`ForwardingReplicaSelectLatency`	Milissegundos	Encaminhamento de latência de `SELECT`, com média sobre

métrica do cloudwatch	Unidade	Descrição
		todas as declarações SELECT encaminhadas no período de monitoramento.
ForwardingReplicaSelectThroughput	Contagem por segundo	Encaminhamento de throughput de SELECT por média de segundos no período de monitoramento.

As variáveis de status do Aurora MySQL a seguir são medidas em cada instância de banco de dados do leitor em um cluster de banco de dados com o encaminhamento de gravação habilitado.

Variável de status do Aurora MySQL	Unidade	Descrição
Aurora_fwd_replica_dml_stmt_count	Contagem	Número total de instruções DML encaminhadas dessa instância de banco de dados de leitor.
Aurora_fwd_replica_dml_stmt_duration	Microssegundos	Duração total de todas as instruções DML encaminha das dessa instância de banco de dados de leitor.
Aurora_fwd_replica_errors_session_limit	Contagem	Número de sessões rejeitadas pelo cluster primário devido a uma das seguintes condições de erro: • gravador completo • Muitas declarações encaminhadas em andamento.

Variável de status do Aurora MySQL	Unidade	Descrição
`Aurora_fwd_replica_open_sessions`	Contagem	O número de sessões que estão usando o encaminhamento de gravação em uma instância de banco de dados do leitor.
`Aurora_fwd_replica_read_wait_count`	Contagem	Número total de esperas de leitura-após-gravação nessa instância de banco de dados de leitor.
`Aurora_fwd_replica_read_wait_duration`	Microssegundos	Duração total das esperas devido à configuração de consistência de leitura nessa instância de banco de dados de leitor.
`Aurora_fwd_replica_select_stmt_count`	Contagem	Número total de instruções `SELECT` encaminhadas dessa instância de banco de dados de leitor.
`Aurora_fwd_replica_select_stmt_duration`	Microssegundos	Duração total das instruções `SELECT` encaminhadas dessa instância de banco de dados de leitor.

Identificar transações e consultas encaminhadas

Você pode usar a tabela `information_schema.aurora_forwarding_processlist` para identificar transações e consultas encaminhadas. Para obter mais informações sobre essa tabela, consulte information_schema.aurora_forwarding_processlist.

O exemplo a seguir mostra todas as conexões encaminhadas em uma instância de banco de dados do gravador.

```
mysql> select * from information_schema.AURORA_FORWARDING_PROCESSLIST where
 IS_FORWARDED=1 order by REPLICA_SESSION_ID;

+-----+----------+-------------------+----------+---------+------+--------------
+-----------------------------------------+--------------+-------------------
+---------------------------------+----------------------+----------------+
| ID  | USER     | HOST              | DB       | COMMAND | TIME | STATE        |
 INFO                                    | IS_FORWARDED | REPLICA_SESSION_ID |
 REPLICA_INSTANCE_IDENTIFIER     | REPLICA_CLUSTER_NAME | REPLICA_REGION |
+-----+----------+-------------------+----------+---------+------+--------------
+-----------------------------------------+--------------+-------------------
+---------------------------------+--------------------------------------+
| 648 | myuser   | IP_address:port1  | sysbench | Query   |    0 | async commit |
 UPDATE sbtest58 SET k=k+1 WHERE id=4802579 |            1 |                637 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
| 650 | myuser   | IP_address:port2  | sysbench | Query   |    0 | async commit |
 UPDATE sbtest54 SET k=k+1 WHERE id=2503953 |            1 |                639 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
+-----+----------+-------------------+----------+---------+------+--------------
+-----------------------------------------+--------------+-------------------
+---------------------------------+----------------------+----------------+
```

Na instância de banco de dados do leitor de encaminhamento, você pode ver os segmentos associados a essas conexões de banco de dados do gravador executando `SHOW PROCESSLIST`. Os valores `REPLICA_SESSION_ID` no gravador, 637 e 639, são iguais aos valores `Id` no leitor.

```
mysql> select @@aurora_server_id;

+---------------------------------+
| @@aurora_server_id              |
+---------------------------------+
| my-db-cluster-instance-2        |
+---------------------------------+
1 row in set (0.00 sec)

mysql> show processlist;

+-----+----------+-------------------+----------+---------+------+--------------
+------------------------------------------+
| Id  | User     | Host              | db       | Command | Time | State        | Info
                                        |
```

```
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
| 637 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest12 SET k=k+1 WHERE id=4802579  |
| 639 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest61 SET k=k+1 WHERE id=2503953  |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
12 rows in set (0.00 sec)
```

Integração do Amazon Aurora MySQL com outros produtos da AWS

O Amazon Aurora MySQL integra-se a outros produtos da AWS para que você possa estender seu cluster de banco de dados Aurora MySQL de forma a usar recursos adicionais na Nuvem AWS. O cluster de banco de dados Aurora MySQL pode usar produtos da AWS para fazer o seguinte:

- Invoque de forma síncrona ou assíncrona uma função do AWS Lambda usando as funções nativas `lambda_sync` ou `lambda_async`. Para obter mais informações, consulte Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL.
- Carregue os dados de arquivos de texto ou XML armazenados em um bucket do Amazon Simple Storage Service (Amazon S3) em seu cluster de banco de dados usando o comando `LOAD DATA FROM S3` ou `LOAD XML FROM S3`. Para obter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.
- Salve os dados em arquivos de texto armazenados em um bucket do Amazon S3 a partir de seu cluster de banco de dados usando o comando `SELECT INTO OUTFILE S3`. Para obter mais informações, consulte Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3.
- Adicione ou remova réplicas do Aurora automaticamente com o Application Auto Scaling. Para obter mais informações, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.
- Realize análises de sentimentos com o Amazon Comprehend ou uma grande variedade de algoritmos de machine learning com o SageMaker AI. Para obter mais informações, consulte Usar o machine learning do Amazon Aurora.

O Aurora garante a capacidade de acesso a outros produtos da AWS usando o AWS Identity and Access Management (IAM). Conceda permissão para acessar outros serviços da AWS criando uma função do IAM com as permissões necessárias e associe a função a seu cluster de banco de dados. Para obter detalhes e instruções sobre como permitir que seu cluster de banco de dados Aurora MySQL acesse outros produtos da AWS em seu nome, consulte Autorizar o Amazon Aurora MySQL a acessar outros produtos da AWS em seu nome.

Autorizar o Amazon Aurora MySQL a acessar outros produtos da AWS em seu nome

Para que o cluster de banco de dados Aurora MySQL acesse outros serviços em seu nome, crie e configure uma função do AWS Identity and Access Management (IAM). Essa função autoriza os usuários de banco de dados em seu cluster de banco de dados a acessar outros serviços da AWS. Para obter mais informações, consulte Configurar funções do IAM para acessar outros produtos da AWS.

Você também deve configurar seu cluster de banco de dados Aurora para permitir conexões de saída com o produto da AWS de destino. Para obter mais informações, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

Se fizer isso, os usuários do seu banco de dados poderão realizar essas ações usando outros serviços da AWS:

- Invoque de forma síncrona ou assíncrona uma função do AWS Lambda usando as funções nativas `lambda_sync` ou `lambda_async`. Ou invoque de forma assíncrona uma função do AWS Lambda usando o procedimento `mysql.lambda_async`. Para obter mais informações, consulte Como invocar uma função do Lambda com uma função nativa do Aurora MySQL.
- Carregue os dados de arquivos de texto ou XML armazenados em um bucket do Amazon S3 no seu cluster de banco de dados usando a instrução `LOAD DATA FROM S3` ou `LOAD XML FROM S3`. Para obter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.
- Salve os dados do cluster de banco de dados em arquivos de texto armazenados em um bucket do Amazon S3 usando a instrução `SELECT INTO OUTFILE S3`. Para obter mais informações, consulte Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3.
- Exportar dados do log para o MySQL do Amazon CloudWatch Logs. Para obter mais informações, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.
- Adicione ou remova réplicas do Aurora automaticamente com o Application Auto Scaling. Para obter mais informações, consulte Amazon Aurora Auto Scaling com réplicas do Aurora.

Configurar funções do IAM para acessar outros produtos da AWS

Para permitir que seu cluster de banco de dados Aurora acesse outro produto da AWS, faça o seguinte:

1. Crie uma política do IAM que conceda permissão ao serviço da AWS. Para obter mais informações, consulte os tópicos a seguir.
   - Criar uma política do IAM para acessar recursos do Amazon S3
   - Criar uma política do IAM para acessar recursos do AWS Lambda
   - Criar uma política do IAM para acessar recursos do CloudWatch Logs
   - Criar uma política do IAM para acessar recursos do AWS KMS
2. Crie uma função do IAM e anexe a política que você criou. Para obter mais informações, consulte Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.
3. Associe essa função do IAM ao seu cluster de banco de dados Aurora. Para obter mais informações, consulte Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.

Criar uma política do IAM para acessar recursos do Amazon S3

O Aurora pode acessar recursos do Amazon S3 para carregar dados ou salvar dados em/de um cluster de banco de dados Aurora. No entanto, primeiro você deve criar uma política do IAM que forneça as permissões de bucket e objeto que permitem ao Aurora acessar o Amazon S3.

A seguinte tabela lista os recursos do Aurora que podem acessar um bucket do Amazon S3 em seu nome, bem com as permissões mínimas de bucket e objetos necessárias por cada recurso.

Recurso	Permissões do bucket	Permissões de objeto
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload`
		`DeleteObject`
		`GetObject`

Recurso	Permissões do bucket	Permissões de objeto

		`ListMultipartUploadParts`
		
		`PutObject`

A política a seguir adiciona as permissões que podem ser exigidas pelo Aurora para acessar um bucket do Amazon S3 em seu nome.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

> **Note**
>
> Certifique-se de incluir ambas as entradas para o valor `Resource`. O Aurora precisa das permissões no bucket em si e em todos os objetos dentro do bucket.

> Com base no seu caso de uso, talvez você não precise adicionar todas as permissões na política de exemplo. Outras permissões também podem ser necessárias. Por exemplo, se o bucket do Amazon S3 estiver criptografado, você precisará adicionar permissões `kms:Decrypt`.

Você pode usar as seguintes etapas para criar uma política do IAM que fornece as permissões mínimas necessárias para o Aurora acessar um bucket do Amazon S3 em seu nome. Para permitir que o Aurora acesse todos os seus buckets do Amazon S3, você pode pular essas etapas e usar a política do IAM predefinida `AmazonS3ReadOnlyAccess` ou `AmazonS3FullAccess`, em vez de criar sua própria.

Para criar uma política do IAM para conceder acesso aos seus recursos do Amazon S3

1. Abra o Console de Gerenciamento do IAM.
2. No painel de navegação, selecione Policies (Políticas).
3. Escolha Create policy (Criar política).
4. Na guia Editor visual, selecione Escolher um serviço e, em seguida, escolha S3.
5. Em Actions (Ações), escolha Expand all (Expandir tudo) e escolha as permissões de bucket e de objeto necessárias para a política do IAM.

   Permissões de objeto são permissões para operações de objeto no Amazon S3 e precisam ser concedidas para objetos em um bucket, não no próprio bucket. Para obter mais informações sobre permissões para operações de objeto no Amazon S3, consulte Permissões para operações de objeto.
6. Escolha Resources (Recursos) e Add ARN (Adicionar ARN) para bucket.
7. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), forneça os detalhes sobre seu recurso e escolha Selecionar.

   Especifique bucket do Amazon S3 ao qual permitir acesso. Por exemplo, para permitir que o Aurora acesse o bucket do Amazon S3 chamado *`amzn-s3-demo-bucket`*, defina o valor do nome do recurso da Amazon (ARN) como `arn:aws:s3:::`*`amzn-s3-demo-bucket`*.
8. Se o recurso objeto estiver listado, escolha Adicionar Nome de recurso da Amazon (ARN) para objeto.
9. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), forneça os detalhes sobre seu recurso.

Para o bucket do Amazon S3, especifique o bucket do Amazon S3 ao qual permitir acesso. Para o objeto, você poderá escolher Qualquer um para conceder permissões para qualquer objeto no bucket.

> **Note**
>
> Você pode definir o Amazon Resource Name (ARN) (Nome de recurso da Amazon – ARN) como um valor de ARN mais específico para permitir que o Aurora acesse apenas arquivos ou pastas específicos em um bucket do Amazon S3. Para obter mais informações sobre como definir uma política de acesso para o Amazon S3, consulte Gerenciar permissões de acesso aos recursos do Amazon S3.

10. (Opcional) Escolha Add ARN (Adicionar ARN) em bucket para adicionar outro bucket do Amazon S3 à política e repita as etapas anteriores para esse bucket.

> **Note**
>
> Você pode repetir esta etapa para adicionar instruções de permissão de bucket correspondentes à sua política para cada bucket do Amazon S3 que o Aurora deve acessar. Opcionalmente, você também pode conceder acesso a todos os buckets e objetos no Amazon S3.

11. Escolha Review policy (Revisar política).
12. Em Name (Nome), insira um nome para a sua política do IAM, por exemplo `AllowAuroraToExampleBucket`. Você usa esse nome ao criar uma função do IAM a ser associada ao seu cluster de banco de dados Aurora. Você também pode adicionar um valor opcional para Description (Descrição).
13. Escolha Create policy (Criar política).
14. Siga as etapas em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.

Criar uma política do IAM para acessar recursos do AWS Lambda

Você pode criar uma política do IAM que fornece as permissões mínimas necessárias para o Aurora invocar uma função do AWS Lambda em seu nome.

A política a seguir adiciona as permissões exigidas pelo Aurora para invocar uma função do AWS Lambda em seu nome.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToExampleFunction",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:123456789012:function:example_function"
    }
  ]
}
```

Você pode usar as seguintes etapas para criar uma política do IAM que fornece as permissões mínimas necessárias para o Aurora invocar uma função do AWS Lambda em seu nome. Para permitir que o Aurora invoque todas as suas funções do AWS Lambda, você pode pular essas etapas e usar a política `AWSLambdaRole` predefinida em vez de criar sua própria.

Para criar uma política do IAM para conceder às suas funções do AWS Lambda

1. Abra o console do IAM.
2. No painel de navegação, selecione Policies (Políticas).
3. Escolha Create policy (Criar política).
4. Na guia Editor visual, selecione Escolher um serviço e, em seguida, escolha Lambda.
5. Em Actions (Ações), escolha Expand all (Expandir tudo) e escolha as permissões do AWS Lambda necessárias para a política do IAM.

   Verifique se `InvokeFunction` está selecionado. É a permissão mínima necessária para habilitar o Amazon Aurora a invocar uma função do AWS Lambda.
6. Escolha Recursos e Adicionar Nome de recurso da Amazon (ARN) para função.

7. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), forneça os detalhes sobre seu recurso.

   Especifique a função do Lambda à qual permitir acesso. Por exemplo, se quiser permitir que o Aurora acesse uma função do Lambda denominada `example_function`, defina o valor do ARN como `arn:aws:lambda:::function:example_function`.

   Para obter mais informações sobre como definir uma política de acesso para o AWS Lambda, consulte Autenticação e controle de acesso para o AWS Lambda.
8. Opcionalmente, escolha Add additional permissions (Adicionar mais permissões) para incluir outra função do AWS Lambda na política e repita as etapas anteriores para a função.

   > **Note**
   >
   > Você pode repetir esta etapa para adicionar instruções de permissão de função correspondentes a sua política para cada função do AWS Lambda que o Aurora deve acessar.
9. Escolha Review policy (Revisar política).
10. Defina Name como um nome para a sua política do IAM, por exemplo `AllowAuroraToExampleFunction`. Você usa esse nome ao criar uma função do IAM a ser associada ao seu cluster de banco de dados do Aurora. Você também pode adicionar um valor opcional para Description (Descrição).
11. Escolha Create policy (Criar política).
12. Siga as etapas em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.

Criar uma política do IAM para acessar recursos do CloudWatch Logs

O Aurora pode acessar o CloudWatch Logs para exportar dados de log de auditoria de um cluster de banco de dados Aurora. No entanto, você precisa primeiro criar uma política do IAM que forneça as permissões do grupo de log e do fluxo de log que possibilitam que o Aurora acesse o CloudWatch Logs.

A seguinte política adiciona as permissões exigidas pelo Aurora para acessar o Amazon CloudWatch Logs em seu nome e as permissões mínimas necessárias para criar grupos de logs e exportar dados.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
        },
        {
            "Sid":
 "EnableCreationAndManagementOfRDSCloudwatchLogGroupsAndStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutRetentionPolicy",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*"
        }
    ]
}
```

É possível modificar os ARNs na política para restringir o acesso a uma região e a uma conta específicas da AWS.

Você pode usar as seguintes etapas para criar uma política do IAM que forneça as permissões mínimas necessárias para o Aurora acessar o CloudWatch Logs em seu nome. Para permitir que o Aurora tenha acesso total ao CloudWatch Logs, você pode pular essas etapas e usar a política do IAM predefinida `CloudWatchLogsFullAccess` em vez de criar sua própria. Para obter mais informações, consulte Usar políticas baseadas em identidade (políticas do IAM) para o CloudWatch Logs no Manual do usuário do Amazon CloudWatch.

Para criar uma política do IAM para conceder acesso aos seus recursos do CloudWatch Logs

1. Abra o console do IAM.
2. No painel de navegação, selecione Policies (Políticas).
3. Escolha Create policy (Criar política).
4. Na guia Editor visual, selecione Escolher um serviço e, em seguida, escolha CloudWatch Logs.
5. Em Actions (Ações), escolha Expand all (Expandir tudo) (à direita) e escolha as permissões do Amazon CloudWatch Logs necessárias para a política do IAM.

   Certifique-se de que as seguintes permissões sejam selecionadas:

   - `CreateLogGroup`
   - `CreateLogStream`
   - `DescribeLogStreams`
   - `GetLogEvents`
   - `PutLogEvents`
   - `PutRetentionPolicy`
6. Escolha Recursos e Adicionar Nome de recurso da Amazon (ARN) para grupo de logs.
7. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), insira os seguintes valores:

   - Region (Região): uma região da AWS ou *
   - Account (Conta) – um número de conta ou *
   - Log Group Name (Nome do grupo de logs – `/aws/rds/*`
8. Na caixa de diálogo Add ARN(s) (Adicionar ARN(s)), escolha Add (Adicionar).
9. Escolha Adicionar Nome de recurso da Amazon (ARN) para fluxo de logs.
10. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), insira os seguintes valores:

    - Region (Região): uma região da AWS ou *
    - Account (Conta) – um número de conta ou *
    - Log Group Name (Nome do grupo de logs – `/aws/rds/*`
    - Log Stream Name (Nome do fluxo de logs – *
11. Na caixa de diálogo Add ARN(s) (Adicionar ARN(s)), escolha Add (Adicionar).

12. Escolha Review policy (Revisar política).
13. Defina Name como um nome para a sua política do IAM, por exemplo `AmazonRDSCloudWatchLogs`. Você usa esse nome ao criar uma função do IAM a ser associada ao seu cluster de banco de dados do Aurora. Você também pode adicionar um valor opcional para Description (Descrição).
14. Escolha Create policy (Criar política).
15. Siga as etapas em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.

Criar uma política do IAM para acessar recursos do AWS KMS

O Aurora pode acessar as AWS KMS keys usadas para criptografar os seus backups de banco de dados. No entanto, primeiro você precisa criar uma política do IAM que forneça as permissões que possibilitem que o Aurora acesse as chaves do KMS.

A política a seguir adiciona as permissões exigidas pelo Aurora para acessar chaves do KMS em seu nome.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToAccessKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-ID"
    }
  ]
}
```

Você pode usar as etapas a seguir para criar uma política do IAM que forneça as permissões mínimas exigidas para que o Aurora acesse as chaves do KMS em seu nome.

Para criar uma política do IAM para conceder acesso a suas chaves do KMS

1. Abra o console do IAM.
2. No painel de navegação, selecione Policies (Políticas).
3. Escolha Create policy (Criar política).
4. Na guia Editor visual, selecione Escolher um serviço e, em seguida, escolha KMS.
5. Em Actions (Ações), expanda Write (Gravação) e escolha Decrypt (Descriptografar).
6. Escolha Resources (Recursos) e Add ARN (Adicionar Nome de recurso da Amazon (ARN) para bucket).
7. Na caixa de diálogo Adicionar Nome(s) de recurso(s) da Amazon (ARN(s)), insira os seguintes valores:
   - Region (Região): digite a região da AWS, como `us-west-2`.
   - Account (Conta) – insira o número de conta do usuário.
   - Nome do fluxo de log: digite o identificador da chave do KMS.
8. Na caixa de diálogo Add ARN(s) (Adicionar ARN(s)), escolha Add (Adicionar).
9. Escolha Review policy (Revisar política).
10. Defina Name como um nome para a sua política do IAM, por exemplo `AmazonRDSKMSKey`. Você usa esse nome ao criar uma função do IAM a ser associada ao seu cluster de banco de dados do Aurora. Você também pode adicionar um valor opcional para Description (Descrição).
11. Escolha Create policy (Criar política).
12. Siga as etapas em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.

Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS

Depois de criar uma política do IAM para permitir que o Aurora acesse recursos da AWS, você deve criar uma função do IAM e anexar a política do IAM a essa nova função do IAM.

Para criar uma função do IAM a fim de permitir que o cluster do Amazon RDS se comunique com outros produtos da AWS em seu nome, siga estas etapas.

Para criar uma função do IAM a fim de permitir que o Amazon RDS acesse produtos da AWS

1. Abra o console do IAM.
2. No painel de navegação, selecione Roles.

3. Selecione Create role.
4. Em Serviço do AWS, escolha RDS.
5. Em Select your use case (Selecionar caso de uso), escolha RDS – Add Role to Database (Adicionar função ao banco de dados).
6. Escolha Próximo.
7. Na página Permissions policies (Políticas de permissões), insira o nome da sua política no campo Search (Pesquisar).
8. Quando aparecer na lista, selecione a política que você definiu anteriormente utilizando as instruções em uma das seções:

   - Criar uma política do IAM para acessar recursos do Amazon S3
   - Criar uma política do IAM para acessar recursos do AWS Lambda
   - Criar uma política do IAM para acessar recursos do CloudWatch Logs
   - Criar uma política do IAM para acessar recursos do AWS KMS
9. Escolha Próximo.
10. Em Role name (Nome da função), insira um nome para sua função do IAM, por exemplo, `RDSLoadFromS3`. Você também pode adicionar um valor opcional para Description (Descrição).
11. Selecione Criar função.
12. Siga as etapas em Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.

Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL

Para permitir que os usuários do banco de dados em um cluster de banco de dados do Amazon Aurora acessem outros serviços da AWS, associe o perfil do IAM criado em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS a esse cluster de banco de dados. A AWS também pode criar para você um perfil do IAM associando o serviço diretamente.

> **Note**
>
> Não é possível associar uma função do IAM a um cluster de banco de dados do Aurora Serverless v1. Para obter mais informações, consulte Usar o Amazon Aurora Serverless v1. É possível associar um perfil do IAM a um cluster de banco de dados do Aurora Serverless v2.

Para associar uma função do IAM a um cluster de banco de dados, é necessário fazer duas coisas:

1. Adicione a função à lista de funções associadas para um cluster de banco de dados usando o console do RDS, o comando add-role-to-db-cluster da AWS CLI ou a operação AddRoleToDBCluster da API do RDS.

   É possível adicionar no máximo cinco funções do IAM para cada cluster de banco de dados Aurora.

2. Defina o parâmetro em nível de cluster do serviço da AWS relacionado como o ARN da função do IAM associada.

   A tabela a seguir descreve os nomes de parâmetros em nível de cluster para as funções do IAM usadas para acessar outros serviços da AWS.

Parâmetro no nível do cluster	Descrição
`aws_default_lambda_role`	Usado ao invocar uma função Lambda no seu cluster de banco de dados.
`aws_default_logs_role`	Esse parâmetro não é mais necessário para exportar dados de log do cluster de banco de dados para o Amazon CloudWatch Logs. O Aurora MySQL agora usa uma função vinculada a serviço para as permissões necessárias. Para obter mais informações sobre funções vinculadas ao serviço, consulte Usar funções vinculadas ao serviço do Amazon Aurora.
`aws_default_s3_role`	Usado ao invocar a instrução `LOAD DATA FROM S3`, `LOAD XML FROM S3` ou `SELECT INTO OUTFILE S3` do seu cluster de banco de dados. No Aurora MySQL versão 2, o perfil do IAM especificado nesse parâmetro será usado se não for especificado um perfil do IAM

Parâmetro no nível do cluster	Descrição	
	para `aurora_load_from_s3_role` ou `aurora_select_into_s3_role` para a instrução apropriada. No Aurora MySQL versão 3, o perfil do IAM especificado para esse parâmetro sempre é usado.	
`aurora_load_from_s3_role`	Usado ao invocar a instrução `LOAD DATA FROM S3` ou `LOAD XML FROM S3` de seu cluster de banco de dados. Se uma função do IAM não for especificada para esse parâmetro, a função do IAM especificada em `aws_default_s3_role` será usada. No Aurora MySQL versão 3, esse parâmetro não está disponível.	
`aurora_select_into_s3_role`	Usado ao invocar a instrução `SELECT INTO OUTFILE S3` de seu cluster de banco de dados. Se uma função do IAM não for especificada para esse parâmetro, a função do IAM especificada em `aws_default_s3_role` será usada. No Aurora MySQL versão 3, esse parâmetro não está disponível.	

Para associar uma função do IAM a fim de permitir que o cluster do Amazon RDS se comunique com outros produtos da AWS em seu nome, siga estas etapas.

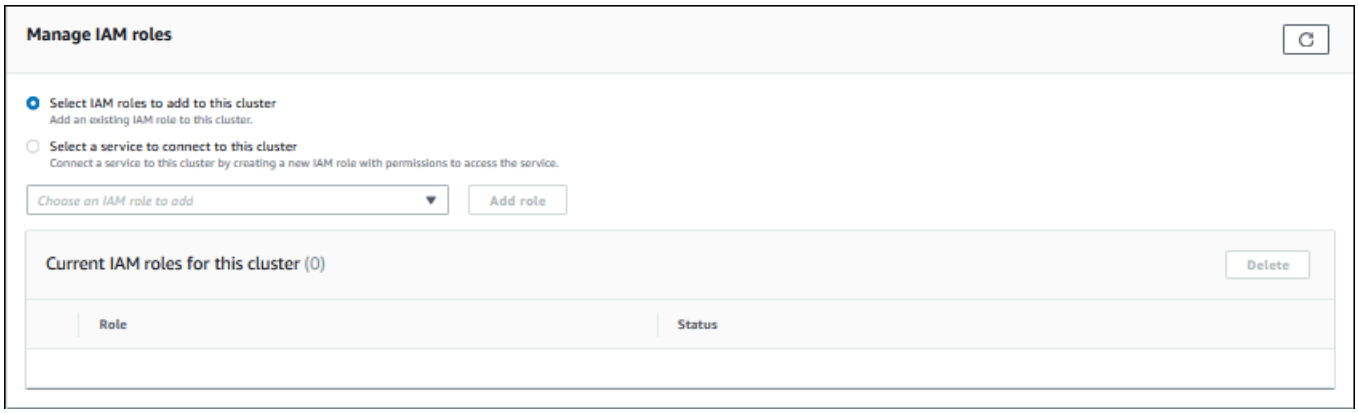
Console

Para associar uma função do IAM a um cluster de banco de dados Aurora usando o console

1. Abra o console do RDS em https://console.aws.amazon.com/rds/.

2. Escolha Databases (Bancos de dados).
3. Escolha o nome do cluster de banco de dados Aurora ao qual você deseja associar uma função do IAM para mostrar seus detalhes.
4. Na guia Connectivity & security (Conectividade e segurança), na seção Manage IAM roles (Gerenciar perfis do IAM), faça o seguinte:

   - Select IAM roles to add to this cluster (Selecionar perfis do IAM a serem adicionados a esse cluster) (padrão)
   - Select a service to connect to this cluster (Selecionar um serviço a ser conectado a esse cluster)



5. Para usar um perfil existente do IAM, selecione-o no menu e depois selecione Add role (Adicionar perfil).

   Se a adição do perfil for bem-sucedida, seu status será exibido como `Pending` e, então, `Available`.

6. Como conectar um serviço diretamente:

   a. Selecione Select a service to connect to this cluster (Selecionar um serviço a ser conectado a esse cluster).

   b. Selecione o serviço no menu e depois selecione Connect service (Conectar serviço).

   c. Em Connect cluster to ***`Service Name`*** [Conectar cluster ao (nome do serviço)], insira o nome do recurso da Amazon (ARN) que será utilizado para se conectar ao serviço e depois selecione Connect service (Conectar serviço).

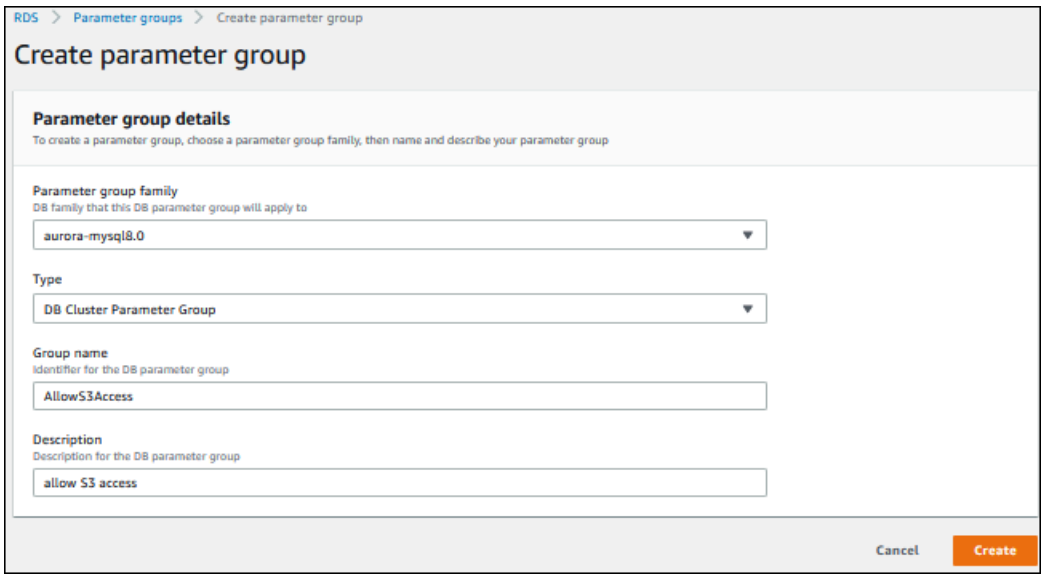
A AWS cria um perfil do IAM para se conectar ao serviço. Seu status é exibido como `Pending` e depois `Available`.

7. (Opcional) Para parar de associar um perfil do IAM a um cluster de banco de dados e remover a permissão relacionada, selecione o perfil e depois selecione Delete (Excluir).

Como definir o parâmetro em nível de cluster para o perfil do IAM

1. No console do RDS, escolha Parameter groups no painel de navegação.
2. Se já estiver usando um parameter group de banco de dados personalizado, você poderá selecionar esse grupo para usar em vez de criar um novo. Se você estiver usando o parameter group do cluster de banco de dados padrão, crie um novo parameter group de cluster de banco de dados, conforme descrito nas etapas a seguir:

   a. Escolha Create parameter group (Criar parameter group).
   b. Em Família de grupos de parâmetros, escolha `aurora-mysql8.0` para um cluster de banco de dados compatível com o Aurora MySQL 8.0 ou escolha `aurora-mysql5.7` para um cluster de banco de dados compatível com o Aurora MySQL 5.7.
   c. Em Type (Tipo), escolha DB Cluster Parameter Group (Grupo de parâmetros do cluster de banco de dados).
   d. Para Group name, digite o nome do seu novo parameter group de cluster de banco de dados.
   e. Para Description, digite uma descrição para o seu novo parameter group de cluster de banco de dados.



f. Escolha Criar.

3. Na página Parameter groups (Grupos de parâmetros), selecione o grupo de parâmetros do cluster do banco de dados e escolha Edit (Editar) em Parameter group actions (Ações em grupos de parâmetros).
4. Defina os parâmetros apropriados em nível de cluster como os valores de ARN do perfil do IAM relacionados.

   Por exemplo, você pode definir apenas o parâmetro `aws_default_s3_role` como `arn:aws:iam::123456789012:role/AllowS3Access`.
5. Selecione Save changes.
6. Para alterar o grupo de parâmetros do cluster de banco de dados para seu cluster de banco de dados, conclua as etapas a seguir:

   a. Escolha Databases (Bancos de dados) e o cluster de banco de dados Aurora.

   b. Selecione Modify.

   c. Role até Database options (Opções de banco de dados) e defina DB cluster parameter group (Grupo de parâmetros de cluster de banco de dados) como o grupo de parâmetros de cluster de banco de dados.

   d. Escolha Continue.

   e. Verifique suas alterações e escolha Apply immediately.

   f. Selecione Modify Cluster (Modificar cluster).

   g. Escolha Databases (Bancos de dados) e a instância primária de seu cluster de banco de dados.

h. Em Actions (Ações), escolha Reboot (Reiniciar).

Quando a instância tiver sido reinicializada, a função do IAM será associada a seu cluster de banco de dados.

Para obter mais informações sobre parameter groups de cluster, consulte Parâmetros de configuração do Aurora MySQL.

CLI

Para associar uma função do IAM a um cluster de banco de dados usando a AWS CLI

1. Chame o comando `add-role-to-db-cluster` da AWS CLI para adicionar os ARNs de suas funções do IAM ao cluster de banco de dados, conforme mostrado a seguir.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraS3Role
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. Se você estiver usando o parameter group do cluster de banco de dados padrão, precisará criar um novo parameter group do cluster de banco de dados. Se já estiver usando um parameter group de banco de dados personalizado, você poderá usar esse grupo em vez de criar um novo.

   Para criar um novo parameter group de cluster de banco de dados, chame o comando `create-db-cluster-parameter-group` da AWS CLI, conforme mostrado a seguir.

```
PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-
name AllowAWSAccess \
     --db-parameter-group-family aurora5.7 --description "Allow access to Amazon S3
 and AWS Lambda"
```

   No caso de um cluster compatível com o MySQL 5.7 do Aurora, especifique `aurora-mysql5.7` para `--db-parameter-group-family`. No caso de um cluster de banco de dados compatível com Aurora MySQL 8.0, especifique `aurora-mysql8.0` para `--db-parameter-group-family`.

3. Defina um ou mais parâmetros apropriados em nível de cluster e também os valores de ARN da função do IAM relacionados no seu parameter group de cluster de banco de dados, conforme mostrado a seguir.

```
PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
 AllowAWSAccess \
    --parameters
 "ParameterName=aws_default_s3_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraS3Role,method=pending-reboot" \
    --parameters
 "ParameterName=aws_default_lambda_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraLambdaRole,method=pending-reboot"
```

4. Modifique o cluster de banco de dados para usar o novo parameter group de cluster de banco de dados e depois reinicialize o cluster, conforme mostrado a seguir.

```
PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
parameter-group-name AllowAWSAccess
PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
```

   Quando a instância tiver sido reinicializada, suas funções do IAM serão associadas ao seu cluster de banco de dados.

   Para obter mais informações sobre parameter groups de cluster, consulte Parâmetros de configuração do Aurora MySQL.

Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS

Para usar determinados produtos da AWS com o Amazon Aurora, a configuração de rede do cluster de banco de dados Aurora deve permitir conexões de saída para os endpoints desses serviços. As operações a seguir exigem essa configuração de rede.

- Chamada de funções do AWS Lambda. Para saber mais sobre esse recurso, consulte Como invocar uma função do Lambda com uma função nativa do Aurora MySQL.
- Acesso a arquivos do Amazon S3. Para saber mais sobre esse recurso, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3 e Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3.
- Acesso a endpoints do AWS KMS. O acesso ao AWS KMS é necessário para usar fluxos de atividade de banco de dados com o Aurora MySQL. Para saber mais sobre esse recurso, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.

- Acessar endpoints do SageMaker AI. O acesso ao SageMaker AI é necessário para usar o machine learning do SageMaker AI com o Aurora MySQL. Para saber mais sobre esse recurso, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.

O Aurora retornará as seguintes mensagens de erro se não puder se conectar a um endpoint de serviço.

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

```
ERROR 1815 (HY000): Internal error: Unable to initialize S3Stream
```

Para fluxos de atividade de banco de dados que usam o Aurora MySQL, o fluxo de atividades para de funcionar se o cluster de banco de dados não puder acessar o endpoint do AWS KMS. O Aurora notifica você sobre esse problema usando eventos do RDS.

Se encontrar essas mensagens ao usar os produtos da AWScorrespondentes, verifique se o cluster de banco de dados Aurora é público ou privado. Se o seu cluster de banco de dados Aurora for privado, você deverá configurá-lo para permitir conexões.

Para que um cluster de banco de dados Aurora seja público, ele deve ser marcado como publicamente acessível. Se você examinar os detalhes do cluster de banco de dados no Console de gerenciamento da AWS, Publicly Accessible (Acessível publicamente) será Yes (Sim), se esse for o caso. O cluster de banco de dados também deve estar em uma sub-rede pública de Amazon VPC. Para obter mais informações sobre instâncias de bancos de dados publicamente acessíveis, consulte Trabalhar com um cluster de banco de dados em uma VPC. Para obter mais informações sobre sub-redes públicas de Amazon VPC, consulte Sua VPC e suas sub-redes.

Se o cluster de banco de dados Aurora não for publicamente acessível e não estiver em uma sub-rede pública de VPC, então ele é privado. Você pode ter um cluster de banco de dados que é privado e desejar usar um dos recursos que exigem essa configuração de rede. Nesse caso, configure o cluster de forma que ele possa conectar-se a endereços de Internet por meio da conversão de endereços de rede (NAT). Como alternativa ao Amazon S3, Amazon SageMaker AI e AWS Lambda, é possível configurar a VPC a fim de ter um endpoint da VPC para o outro serviço associado à tabela de rotas do cluster de banco de dados. Consulte Trabalhar com um cluster de banco de dados em uma VPC. Para obter mais informações sobre como configurar a NAT na sua VPC, consulte

Gateways NAT. Para obter mais informações sobre como configurar VPC endpoints, consulte VPC endpoints. Você também pode criar um endpoint de gateway do S3 para acessar o bucket do S3. Para obter mais informações, consulte Endpoints de gateway para o Amazon S3.

Talvez você também precise abrir as portas efêmeras de suas listas de controle de acesso à rede (ACLs) nas regras de saída do grupo de segurança da VPC. Para obter mais informações sobre portas efêmeras para ACLs da rede, consulte Portas efêmeras no Guia do usuário da Amazon Virtual Private Cloud.

Tópicos relacionados

- Integração do Aurora com outros produtos da AWS
- Como gerenciar um cluster de banco de dados do Amazon Aurora

Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3

Você pode usar a instrução `LOAD DATA FROM S3` ou a `LOAD XML FROM S3` para carregar dados de arquivos armazenados em um bucket do Amazon S3. No Aurora MySQL, os arquivos são armazenados primeiramente no disco local e depois importados para o banco de dados. Depois que as importações para o banco de dados forem concluídas, os arquivos locais serão excluídos.

> **Note**
>
> O carregamento de dados de arquivos de texto em uma tabela não é compatível com o Aurora Serverless v1. É compatível com o Aurora Serverless v2.

Sumário

- Concessão de acesso ao Amazon S3 para o Aurora
- Conceder privilégios para carregar dados no Amazon Aurora MySQL
- Especificar o caminho (URI) para um bucket do Amazon S3
- LOAD DATA FROM S3
  - Sintaxe
  - Parâmetros
  - Usar um manifesto para especificar arquivos de dados para carregamento

    - Verificar arquivos carregados usando a tabela aurora_s3_load_history
  - Exemplos
- LOAD XML FROM S3
  - Sintaxe
  - Parâmetros

Concessão de acesso ao Amazon S3 para o Aurora

Para poder carregar dados de um bucket do Amazon S3, você deve primeiro fornecer permissão a seu cluster de banco de dados Aurora MySQL para acessar o Amazon S3.

Para conceder acesso ao Amazon S3 para o Aurora MySQL

1. Crie uma política do AWS Identity and Access Management (IAM) que forneça as permissões de bucket e de objeto que permitem que o cluster de banco de dados Aurora MySQL acesse o Amazon S3. Para instruções, consulte Criar uma política do IAM para acessar recursos do Amazon S3.

   > **Note**
   >
   > No Aurora MySQL versão 3.05 e superior, você pode carregar objetos criptografados usando a chave AWS KMS keys gerenciada pelo cliente. Para isso, inclua a permissão `kms:Decrypt` na sua política do IAM. Para obter mais informações, consulte Criar uma política do IAM para acessar recursos do AWS KMS.
   > Você não precisa dessa permissão para carregar objetos que são criptografados com a chave Chaves gerenciadas pela AWS ou chaves gerenciadas pelo Amazon S3 (SSE-S3).

2. Crie um perfil do IAM e anexe a política do IAM criada em Criar uma política do IAM para acessar recursos do Amazon S3 ao novo perfil do IAM. Para obter instruções, consulte Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.
3. Certifique-se de que o cluster de banco de dados esteja usando um grupo de parâmetros de cluster de banco de dados personalizado.

   Para ter mais informações sobre como criar um grupo de parâmetros de cluster de banco de dados personalizado, consulte Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

4. No caso do Aurora MySQL versão 2, defina o parâmetro de cluster de banco de dados `aurora_load_from_s3_role` ou `aws_default_s3_role` como o nome do recurso da Amazon (ARN) do novo perfil do IAM. Se um perfil do IAM não for especificado para `aurora_load_from_s3_role`, o Aurora usará o perfil do IAM especificado em `aws_default_s3_role`.

   No caso do Aurora MySQL versão 3, use `aws_default_s3_role`.

   Se o cluster fizer parte de um banco de dados global Aurora, defina esse parâmetro para cada cluster do Aurora no banco de dados global. Embora somente o cluster primário em um banco de dados global Aurora possa carregar dados, outro cluster pode ser promovido pelo mecanismo de failover e se tornar o cluster primário.

   Para ter mais informações sobre parâmetros de cluster de banco de dados, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora.
5. Para permitir que os usuários do banco de dados em um cluster de banco de dados Aurora MySQL acessem o Amazon S3, associe a função criada em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS ao cluster de banco de dados. Para um banco de dados global Aurora, associe a função a cada cluster do Aurora no banco de dados global. Para obter informações sobre como associar um perfil do IAM a um cluster de banco de dados, consulte Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.
6. Configure o cluster de banco de dados Aurora MySQL para permitir conexões de saída com o Amazon S3. Para instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

   Se o cluster de banco de dados do não for publicamente acessível e não estiver em uma sub-rede pública de VPC, então ele é privado. Você pode criar um endpoint de gateway do S3 para acessar o bucket do S3. Para obter mais informações, consulte Endpoints de gateway para o Amazon S3.

   Para um banco de dados global Aurora, habilite conexões de saída para cada cluster do Aurora no banco de dados global.

Conceder privilégios para carregar dados no Amazon Aurora MySQL

O usuário do banco de dados que emite a instrução `LOAD DATA FROM S3` ou `LOAD XML FROM S3` deve ter uma função ou um privilégio específico para emitir qualquer uma dessas instruções. No

Aurora MySQL versão 3, é concedida a função `AWS_LOAD_S3_ACCESS`. No Aurora MySQL versão 2, é concedido o privilégio `LOAD FROM S3`. O usuário administrativo de um cluster de banco de dados recebe a devida função ou privilégio por padrão. É possível conceder o privilégio a outro usuário, utilizando uma das instruções a seguir.

Use a instrução a seguir para o Aurora MySQL versão 3:

```
GRANT AWS_LOAD_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Ao utilizar a técnica de perfil no Aurora MySQL versão 3, você também pode ativar o perfil usando a instrução `SET ROLE role_name` ou `SET ROLE ALL`. Se não estiver familiarizado com o sistema de funções do MySQL 8.0, é possível saber mais em Modelo de privilégios baseados em funções. Para ver mais detalhes, consulte Using roles no MySQL Reference Manual.
>
> Isso se aplica somente à sessão ativa atual. Ao se reconectar, execute a declaração `SET ROLE` novamente para conceder privilégios. Para ter mais informações, consulte a instrução SET ROLE no Guia de referência do MySQL.
>
> Você pode usar o parâmetro `activate_all_roles_on_login` de cluster de banco de dados para ativar automaticamente todos os perfis quando um usuário se conecta a uma instância de banco de dados. Quando esse parâmetro está definido, não é necessário chamar a declaração `SET ROLE` explicitamente para ativar um perfil. Para ter mais informações, consulte activate_all_roles_on_login no Guia de referência do MySQL.
>
> No entanto, é necessário chamar `SET ROLE ALL` explicitamente no início de um procedimento armazenado para ativar o perfil, quando o procedimento armazenado é chamado por um usuário diferente.

Use a seguinte instrução para o Aurora MySQL versão 2:

```
GRANT LOAD FROM S3 ON *.* TO 'user'@'domain-or-ip-address'
```

O perfil `AWS_LOAD_S3_ACCESS` e o privilégio `LOAD FROM S3` são específicos do Amazon Aurora e não estão disponíveis para bancos de dados MySQL externos ou instâncias de banco de dados do RDS para MySQL. Se você tiver configurado a replicação entre um cluster de banco de dados do Aurora como a origem de replicação e um banco de dados MySQL como o cliente de replicação, a declaração `GRANT` do perfil ou do privilégio fará com que a replicação pare devido a um erro. Você

pode ignorar o erro com segurança para retomar a replicação. Para ignorar o erro em uma instância do RDS para MySQL, use o procedimento mysql_rds_skip_repl_error. Para ignorar o erro em um banco de dados MySQL externo, use a variável de sistema slave_skip_errors (Aurora MySQL versão 2) ou a variável de sistema replica_skip_errors (Aurora MySQL versão 3).

> **Note**
>
> O usuário do banco de dados deve ter privilégios `INSERT` para o banco de dados no qual está carregando dados.

Especificar o caminho (URI) para um bucket do Amazon S3

A sintaxe para especificar o caminho (URI) para arquivos armazenados em um bucket do Amazon S3 é mostrada a seguir.

```
s3-region://amzn-s3-demo-bucket/file-name-or-prefix
```

O caminho inclui os seguintes valores:

- `region` (opcional): a região da AWS que contém o bucket do Amazon S3 do qual carregar. Este valor é opcional. Se você não especificar um valor para `region`, o Aurora carregará o arquivo do Amazon S3 na mesma região que o seu cluster de banco de dados.
- `bucket-name`: o nome do bucket do Amazon S3 que contém os dados a serem carregados. Há suporte para prefixos do objeto que identificam um caminho de pasta virtual.
- `file-name-or-prefix`: o nome do arquivo de texto ou arquivo XML do Amazon S3, ou um prefixo que identifica um ou mais arquivos de texto ou XML para carregar. Você também pode especificar um arquivo manifesto que identifique um ou mais arquivos de texto para carregar. Para ter mais informações sobre como usar um arquivo manifesto para carregar arquivos de texto a partir do Amazon S3, consulte Usar um manifesto para especificar arquivos de dados para carregamento.

Como copiar o URI para arquivos em um bucket do S3

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. No painel de navegação, escolha Buckets e selecione o bucket cujo URI você deseja copiar.

3. Selecione o prefixo ou arquivo que deseja carregar do S3.
4. Escolha Copiar URI do S3.

LOAD DATA FROM S3

Você pode usar a instrução `LOAD DATA FROM S3` para carregar dados de qualquer formato de arquivo de texto com suporte pela instrução LOAD DATA INFILE, como dados de texto que são delimitados por vírgulas. Não há suporte para arquivos compactados.

> **Note**
>
> Verifique se o cluster de banco de dados do Aurora MySQL permite conexões de saída com o S3. Para obter mais informações, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

Sintaxe

```
LOAD DATA [FROM] S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

> **Note**
>
> No Aurora MySQL versão 3.05 e superior, a palavra-chave `FROM` é opcional.

Parâmetros

A instrução `LOAD DATA FROM S3` usa os seguintes parâmetros obrigatórios e opcionais. Você pode encontrar mais detalhes sobre alguns desses parâmetros em Instrução LOAD DATA, na documentação do MySQL.

FILE | PREFIX | MANIFEST

Identifica se os dados devem ser carregados de um único arquivo, de todos os arquivos que correspondem a determinado prefixo ou de todos os arquivos em um manifesto especificado. `FILE` é o padrão.

S3-URI

Especifica o URI de um arquivo de texto ou manifesto a ser carregado, ou um prefixo do Amazon S3 a ser usado. Especifique o URI usando a sintaxe descrita em Especificar o caminho (URI) para um bucket do Amazon S3.

REPLACE | IGNORE

Determina a ação a ser tomada se uma linha de entrada tiver os mesmos valores de chave exclusivos que os de uma linha existente na tabela de banco de dados.

- Especifique `REPLACE` se quiser que a linha de entrada substitua a linha existente na tabela.
- Especifique `IGNORE` se quiser descartar a linha de entrada.

INTO TABLE

Identifica o nome da tabela do banco de dados na qual carregar as linhas de entrada.

PARTITION

Requer que todas as linhas de entrada sejam inseridas nas partições identificadas pela lista especificada de nomes de partição separados por vírgulas. Se uma linha de entrada não puder ser inserida em uma das partições especificadas, a instrução falhará e um erro será retornado.

CHARACTER SET

Identifica o conjunto de caracteres dos dados no arquivo de entrada.

FIELDS | COLUMNS

Identifica como os campos ou colunas no arquivo de entrada são delimitados. Os campos são delimitados por tabulação por padrão.

LINES

Identifica como as linhas no arquivo de entrada são delimitadas. As linhas são delimitadas por um caractere de nova linha (`'\n'`) por padrão.

IGNORE *número* LINES | ROWS

Especifica que determinado número de linhas no início do arquivo de entrada devem ser ignoradas. Por exemplo, você pode usar `IGNORE 1 LINES` para ignorar uma linha de cabeçalho inicial contendo nomes de coluna ou `IGNORE 2 ROWS` para ignorar as duas primeiras linhas de dados no arquivo de entrada. Se você também usar `PREFIX`, `IGNORE` ignorará determinado número de linhas no início do primeiro arquivo de entrada.

col_name_or_user_var, ...

Especifica uma lista separada por vírgulas de um ou mais nomes de colunas ou variáveis de usuário que identificam quais colunas devem ser carregadas, por nome. O nome de uma variável de usuário usada para esse fim deve corresponder ao nome de um elemento do arquivo de texto, com o prefixo @. Você pode aplicar variáveis de usuário para armazenar valores de campos correspondentes para reutilização.

Por exemplo, a seguinte instrução carrega a primeira coluna do arquivo de entrada na primeira coluna de `table1` e define o valor da coluna `table_column2` em `table1` como o valor da entrada da segunda coluna dividida por 100.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

SET

Especifica uma lista de operações de atribuição separadas por vírgulas que define os valores de colunas na tabela como valores não incluídos no arquivo de entrada.

Por exemplo, a seguinte instrução define as duas primeiras colunas de `table1` como os valores nas duas primeiras colunas do arquivo de entrada e, em seguida, define o valor do `column3` em `table1` como o carimbo de data/hora atual.

```
LOAD DATA FROM S3  's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, column2)
```

```
    SET column3 = CURRENT_TIMESTAMP;
```

Você pode usar subconsultas no lado direito das atribuições de SET. Para uma subconsulta que retorna um valor a ser atribuído a uma coluna, você pode usar apenas uma subconsulta escalar. Além disso, você não pode usar uma subconsulta para selecionar a tabela que está sendo carregada.

Você não poderá usar a palavra-chave LOCAL da instrução LOAD DATA FROM S3 se estiver carregando dados de um bucket do Amazon S3.

Usar um manifesto para especificar arquivos de dados para carregamento

Você pode usar a instrução LOAD DATA FROM S3 com a palavra-chave MANIFEST para especificar um arquivo manifesto no formato JSON que lista os arquivos de texto a serem carregados em uma tabela no seu cluster de banco de dados.

O seguinte esquema JSON descreve o formato e o conteúdo de um arquivo manifesto.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false",
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
                    }
```

```
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Cada `url` no manifesto deve especificar um URL com o nome do bucket e o caminho de objeto completo para o arquivo, e não apenas um prefixo. Você pode usar um manifesto para carregar vários arquivos de buckets diferentes, diferentes regiões ou arquivos que não compartilham o mesmo prefixo. Se uma região não for especificada no URL, a região do cluster de banco de dados Aurora de destino será usada. O exemplo a seguir mostra um arquivo manifesto que carrega quatro arquivos de diferentes buckets.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

O sinalizador opcional `mandatory` especifica se `LOAD DATA FROM S3` deve retornar um erro se o arquivo não for localizado. O sinalizador `mandatory` assume como padrão `false`. Independentemente de como `mandatory` está definido, `LOAD DATA FROM S3` será encerrado se nenhum arquivo for encontrado.

Arquivos manifestos podem ter qualquer extensão. O exemplo a seguir executa a instrução `LOAD DATA FROM S3` com o manifesto do exemplo anterior, que se chama **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

Após a conclusão da instrução, uma entrada para cada arquivo carregado com êxito é gravada na tabela `aurora_s3_load_history`.

Verificar arquivos carregados usando a tabela aurora_s3_load_history

Cada instrução `LOAD DATA FROM S3` bem-sucedida atualiza a tabela `aurora_s3_load_history` no esquema `mysql` com uma entrada para cada arquivo que foi carregado.

Depois de executar a instrução `LOAD DATA FROM S3`, você pode verificar quais arquivos foram carregados consultando a tabela `aurora_s3_load_history`. Para ver os arquivos que foram carregados de uma iteração da instrução, use a cláusula `WHERE` para filtrar os registros na URI do Amazon S3 para o arquivo manifesto usado na instrução. Se você usou o mesmo arquivo manifesto antes, filtre os resultados usando o campo `timestamp`.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

A tabela a seguir descreve os campos na tabela `aurora_s3_load_history`.

Campo	Descrição
`load_prefix`	O URI que foi especificado na instrução de carregamento. Este URI pode ser mapeado para: • Um arquivo de dados único para uma instrução `LOAD DATA FROM S3 FILE`

Campo	Descrição
	• Um prefixo do Amazon S3 que mapeia para vários arquivos de dados para uma instrução `LOAD DATA FROM S3 PREFIX` • Um arquivo manifesto único que contém os nomes de arquivos a serem carregados para uma instrução `LOAD DATA FROM S3 MANIFEST`
`file_name`	O nome de um arquivo que foi carregado no Aurora a partir do Amazon S3 usando a URI identificada no campo `load_prefix` .
`version_number`	O número da versão do arquivo identificado pelo campo `file_name` que foi carregado, se o bucket do Amazon S3 tiver um número de versão.
`bytes_loaded`	O tamanho do arquivo carregado, em bytes.
`load_timestamp`	O carimbo de data/hora de quando a instrução `LOAD DATA FROM S3` foi concluída.

Exemplos

A seguinte instrução carrega dados de um bucket do Amazon S3 que está na mesma região que o cluster de banco de dados Aurora. A declaração lê os dados delimitados por vírgula no arquivo `customerdata.txt` que está no bucket *`amzn-s3-demo-bucket`* do Amazon S3 e, depois, carrega os dados na tabela `store-schema.customer-table`.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

A seguinte instrução carrega dados de um bucket do Amazon S3 que está em uma região diferente do cluster de banco de dados Aurora. A declaração lê os dados delimitados por vírgula de todos os arquivos que correspondem ao prefixo do objeto `employee-data` no bucket *`amzn-s3-demo-bucket`* do Amazon S3 na região `us-west-2` e, depois, carrega os dados na tabela `employees`.

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://amzn-s3-demo-bucket/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

A seguinte instrução carrega dados dos arquivos especificados em um arquivo de manifesto JSON chamado q1_sales.json na tabela `sales`.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://amzn-s3-demo-bucket1/q1_sales.json'
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOAD XML FROM S3

Você pode usar a instrução `LOAD XML FROM S3` para carregar dados de arquivos XML armazenados em um bucket do Amazon S3 em um dos três formatos XML diferentes:

- Nomes de coluna como atributos de um elemento `<row>`. O valor do atributo identifica o conteúdo do campo da tabela.

  ```
  <row column1="value1" column2="value2" .../>
  ```

- Nomes de colunas como elementos filho de um elemento `<row>`. O valor do elemento filho identifica o conteúdo do campo da tabela.

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- Nomes de colunas no atributo `name` de elementos `<field>` em um elemento `<row>`. O valor do elemento `<field>` identifica o conteúdo do campo da tabela.

  ```
  <row>
    <field name='column1'>value1</field>
    <field name='column2'>value2</field>
  </row>
  ```

Sintaxe

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parâmetros

A instrução `LOAD XML FROM S3` usa os seguintes parâmetros obrigatórios e opcionais. Você pode encontrar mais detalhes sobre alguns desses parâmetros em Instrução LOAD XML, na documentação do MySQL.

FILE | PREFIX

Identifica se os dados devem ser carregados de um único arquivo ou de todos os arquivos que correspondam ao prefixo especificado. `FILE` é o padrão.

REPLACE | IGNORE

Determina a ação a ser tomada se uma linha de entrada tiver os mesmos valores de chave exclusivos que os de uma linha existente na tabela de banco de dados.

- Especifique `REPLACE` se quiser que a linha de entrada substitua a linha existente na tabela.
- Especifique `IGNORE` se quiser descartar a linha de entrada. `IGNORE` é o padrão.

INTO TABLE

Identifica o nome da tabela do banco de dados na qual carregar as linhas de entrada.

PARTITION

Requer que todas as linhas de entrada sejam inseridas nas partições identificadas pela lista especificada de nomes de partição separados por vírgulas. Se uma linha de entrada não puder ser inserida em uma das partições especificadas, a instrução falhará e um erro será retornado.

CHARACTER SET

Identifica o conjunto de caracteres dos dados no arquivo de entrada.

ROWS IDENTIFIED BY

Indica o nome do elemento que identifica uma linha no arquivo de entrada. O padrão é `<row>`.

IGNORE ***número*** LINES | ROWS

Especifica que determinado número de linhas no início do arquivo de entrada devem ser ignoradas. Por exemplo, você pode usar `IGNORE 1 LINES` para ignorar a primeira linha no arquivo de texto ou `IGNORE 2 ROWS` para ignorar as duas primeiras linhas de dados no XML de entrada.

field_name_or_user_var, ...

Especifica uma lista separada por vírgulas de um ou mais nomes de elementos XML ou variáveis de usuário que identificam quais elementos devem ser carregados, por nome. O nome de uma variável de usuário usada para esse fim deve corresponder ao nome de um elemento do arquivo XML, com o prefixo @. Você pode aplicar variáveis de usuário para armazenar valores de campos correspondentes para reutilização.

Por exemplo, a seguinte instrução carrega a primeira coluna do arquivo de entrada na primeira coluna de `table1` e define o valor da coluna `table_column2` em `table1` como o valor da entrada da segunda coluna dividida por 100.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

SET

Especifica uma lista de operações de atribuição separadas por vírgulas que define os valores de colunas na tabela como valores não incluídos no arquivo de entrada.

Por exemplo, a seguinte instrução define as duas primeiras colunas de `table1` como os valores nas duas primeiras colunas do arquivo de entrada e, em seguida, define o valor do `column3` em `table1` como o carimbo de data/hora atual.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

Você pode usar subconsultas no lado direito das atribuições de SET. Para uma subconsulta que retorna um valor a ser atribuído a uma coluna, você pode usar apenas uma subconsulta escalar. Além disso, você não pode usar uma subconsulta de seleção da tabela que está sendo carregada.

Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3

É possível usar a declaração `SELECT INTO OUTFILE S3` para consultar dados de um cluster de banco de dados do Amazon Aurora MySQL e salvá-los em arquivos de texto armazenados em um bucket do Amazon S3. No Aurora MySQL, os arquivos são armazenados primeiramente no disco local e depois exportados para o S3. Depois que as exportações forem concluídas, os arquivos locais serão excluídos.

É possível criptografar o bucket do Amazon S3 usando uma chave gerenciada pelo Amazon S3 (SSE-S3) ou AWS KMS key (SSE-KMS: Chave gerenciada pela AWS ou chave gerenciada pelo cliente).

A declaração `LOAD DATA FROM S3` pode usar os arquivos criados pela declaração `SELECT INTO OUTFILE S3` para carregar dados em um cluster de banco de dados do Aurora. Para obter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.

> **Note**
>
> Esse recurso não é compatível com clusters de banco de dados do Aurora Serverless v1. É compatível com clusters de banco de dados do Aurora Serverless v2.
> É possível salvar o cluster de banco de dados e os dados de snapshot do cluster de banco de dados no Amazon S3 usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS. Para obter mais informações, consulte Exportar dados do cluster de banco de dados para o Amazon S3 e Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.

Sumário

- Concessão de acesso ao Amazon S3 para o Aurora MySQL
- Conceder privilégios para salvar dados no Aurora MySQL

- Especificação de um caminho para um bucket do Amazon S3
- Criar um manifesto para listar arquivos de dados
- SELECT INTO OUTFILE S3
  - Sintaxe
  - Parâmetros
  - Considerações
  - Exemplos

Concessão de acesso ao Amazon S3 para o Aurora MySQL

Antes de poder salvar dados em um bucket do Amazon S3, você deve primeiro dar permissão para o seu cluster de banco de dados Aurora MySQL acessar o Amazon S3.

Para conceder acesso ao Amazon S3 para o Aurora MySQL

1. Crie uma política do AWS Identity and Access Management (IAM) que forneça as permissões de bucket e de objeto que permitem que o cluster de banco de dados Aurora MySQL acesse o Amazon S3. Para instruções, consulte Criar uma política do IAM para acessar recursos do Amazon S3.

   > **Note**
   >
   > No Aurora MySQL versão 3.05 e superior, você pode criptografar objetos usando chaves AWS KMSgerenciadas pelo cliente. Para isso, inclua a permissão `kms:GenerateDataKey` na sua política do IAM. Para obter mais informações, consulte Criar uma política do IAM para acessar recursos do AWS KMS.
   > Não é necessária essa permissão para criptografar objetos usando Chaves gerenciadas pela AWS ou chaves gerenciadas pelo Amazon S3 (SSE-S3).

2. Crie um perfil do IAM e anexe a política do IAM criada em Criar uma política do IAM para acessar recursos do Amazon S3 ao novo perfil do IAM. Para instruções, consulte Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.
3. No caso do Aurora MySQL versão 2, defina o parâmetro de cluster de banco de dados `aurora_select_into_s3_role` ou `aws_default_s3_role` como o nome do recurso da Amazon (ARN) do novo perfil do IAM. Se um perfil do IAM não for especificado para

`aurora_select_into_s3_role`, o Aurora usará o perfil do IAM especificado em `aws_default_s3_role`.

No caso do Aurora MySQL versão 3, use `aws_default_s3_role`.

Se o cluster fizer parte de um banco de dados global Aurora, defina esse parâmetro para cada cluster do Aurora no banco de dados global.

Para ter mais informações sobre parâmetros de cluster de banco de dados, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora.

4. Para permitir que os usuários do banco de dados em um cluster de banco de dados Aurora MySQL acessem o Amazon S3, associe a função criada em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS ao cluster de banco de dados.

   Para um banco de dados global Aurora, associe a função a cada cluster do Aurora no banco de dados global.

   Para obter informações sobre como associar um perfil do IAM a um cluster de banco de dados, consulte Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.

5. Configure o cluster de banco de dados Aurora MySQL para permitir conexões de saída com o Amazon S3. Para obter instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

   Para um banco de dados global Aurora, habilite conexões de saída para cada cluster do Aurora no banco de dados global.

Conceder privilégios para salvar dados no Aurora MySQL

O usuário do banco de dados que emite a instrução `SELECT INTO OUTFILE S3` deve ter uma função ou privilégio específico. No Aurora MySQL versão 3, é concedida a função `AWS_SELECT_S3_ACCESS`. No Aurora MySQL versão 2, é concedido o privilégio `SELECT INTO S3`. O usuário administrativo de um cluster de banco de dados recebe a devida função ou privilégio por padrão. É possível conceder o privilégio a outro usuário, utilizando uma das instruções a seguir.

Use a instrução a seguir para o Aurora MySQL versão 3:

```
GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Ao utilizar a técnica de perfil no Aurora MySQL versão 3, você também pode ativar o perfil usando a instrução `SET ROLE role_name` ou `SET ROLE ALL`. Se não estiver familiarizado com o sistema de funções do MySQL 8.0, é possível saber mais em Modelo de privilégios baseados em funções. Para ver mais detalhes, consulte Using roles no MySQL Reference Manual.
>
> Isso se aplica somente à sessão ativa atual. Ao se reconectar, execute a declaração `SET ROLE` novamente para conceder privilégios. Para ter mais informações, consulte a instrução SET ROLE no Guia de referência do MySQL.
>
> Você pode usar o parâmetro `activate_all_roles_on_login` de cluster de banco de dados para ativar automaticamente todos os perfis quando um usuário se conecta a uma instância de banco de dados. Quando esse parâmetro está definido, não é necessário chamar a declaração `SET ROLE` explicitamente para ativar um perfil. Para ter mais informações, consulte activate_all_roles_on_login no Guia de referência do MySQL.
>
> No entanto, é necessário chamar `SET ROLE ALL` explicitamente no início de um procedimento armazenado para ativar o perfil, quando o procedimento armazenado é chamado por um usuário diferente.

Use a seguinte instrução para o Aurora MySQL versão 2:

```
GRANT SELECT INTO S3 ON *.* TO 'user'@'domain-or-ip-address'
```

A função `AWS_SELECT_S3_ACCESS` e o privilégio `SELECT INTO S3` são específicos do Amazon Aurora MySQL e não estão disponíveis para bancos de dados do MySQL ou instâncias de banco de dados RDS para MySQL. Se você tiver configurado a replicação entre um cluster de banco de dados do Aurora MySQL como a origem da replicação e um banco de dados MySQL como o cliente de replicação, a declaração `GRANT` do perfil ou do privilégio fará com que essa replicação pare devido a um erro. Você pode ignorar o erro com segurança para retomar a replicação. Para ignorar o erro em uma instância de banco de dados RDS para MySQL, use o procedimento mysql_rds_skip_repl_error. Para ignorar o erro em um banco de dados MySQL externo, use a variável de sistema slave_skip_errors (Aurora MySQL versão 2) ou a variável de sistema replica_skip_errors (Aurora MySQL versão 3).

Especificação de um caminho para um bucket do Amazon S3

A sintaxe para especificar um caminho para armazenar os dados e os arquivos de manifesto em um bucket do Amazon S3 é semelhante à usada na instrução `LOAD DATA FROM S3 PREFIX`, conforme mostrado a seguir.

```
s3-region://bucket-name/file-prefix
```

O caminho inclui os seguintes valores:

- `region` (opcional): a região da AWS que contém o bucket do Amazon S3 no qual salvar os dados. Este valor é opcional. Se você não especificar um valor para `region`, o Aurora salvará seus arquivos no Amazon S3, na mesma região que o seu cluster de banco de dados.
- `bucket-name`: o nome do bucket do Amazon S3 no qual salvar os dados. Há suporte para prefixos do objeto que identificam um caminho de pasta virtual.
- `file-prefix`: o prefixo de objeto do Amazon S3 que identifica os arquivos a serem salvos no Amazon S3.

Os arquivos de dados criados pela instrução `SELECT INTO OUTFILE S3` usam o seguinte caminho, no qual *`00000`* representa um número inteiro de 5 dígitos com base em zeros.

```
s3-region://bucket-name/file-prefix.part_00000
```

Por exemplo, suponha que uma instrução `SELECT INTO OUTFILE S3` especifique `s3-us-west-2://bucket/prefix` como o caminho no qual armazenar arquivos de dados e crie três arquivos de dados. O bucket do Amazon S3 especificado contém os seguintes arquivos de dados.

- s3-us-west-2://bucket/prefix.part_00000
- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Criar um manifesto para listar arquivos de dados

Você pode usar a instrução `SELECT INTO OUTFILE S3` com a opção `MANIFEST ON` para criar um arquivo manifesto no formato JSON que lista os arquivos de texto criados pela instrução. A instrução `LOAD DATA FROM S3` pode usar o arquivo manifesto para carregar os arquivos de dados de volta

para um cluster de banco de dados Aurora MySQL. Para ter mais informações sobre como usar um manifesto para carregar arquivos de dados do Amazon S3 em um cluster de banco de dados Aurora MySQL, consulte Usar um manifesto para especificar arquivos de dados para carregamento.

Os arquivos de dados incluídos no manifesto criado pela instrução `SELECT INTO OUTFILE S3` são listados na ordem em que são criados pela instrução. Por exemplo, suponha que uma instrução `SELECT INTO OUTFILE S3` tenha especificado `s3-us-west-2://bucket/prefix` como o caminho no qual armazenar arquivos de dados e criado três arquivos de dados e um arquivo manifesto. O bucket do Amazon S3 especificado contém um arquivo manifesto chamado `s3-us-west-2://bucket/prefix.manifest`, que contém as seguintes informações.

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECT INTO OUTFILE S3

Você pode usar a instrução `SELECT INTO OUTFILE S3` para consultar dados de um cluster de banco de dados e salvá-los diretamente em arquivos de texto delimitados armazenados em um bucket do Amazon S3.

Não há suporte para arquivos compactados. Os arquivos criptografados têm suporte a partir do Aurora MySQL 2.09.0.

Sintaxe

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
```

```
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]
    [ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['cmk_id']}]

export_options:
    [FORMAT {CSV|TEXT} [HEADER]]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
]
```

Parâmetros

A instrução `SELECT INTO OUTFILE S3` usa os seguintes parâmetros obrigatórios e opcionais que são específicos do Aurora.

s3-uri

Especifica o URI de um prefixo do Amazon S3 a ser usado. Use a sintaxe descrita em Especificação de um caminho para um bucket do Amazon S3.

FORMAT {CSV|TEXT} [HEADER]

Opcionalmente, salva os dados em formato CSV.

A opção `TEXT` é o padrão e produz o formato de exportação MySQL existente.

A opção `CSV` produz valores de dados separados por vírgulas. O formato CSV segue a especificação em RFC-4180. Se você especificar a palavra-chave opcional `HEADER`, o arquivo de saída conterá uma linha de cabeçalho. Os rótulos na linha de cabeçalho correspondem aos nomes de coluna da instrução `SELECT`. É possível usar os arquivos CSV como modelos de dados de treinamento para uso com os serviços do AWS ML. Para ter mais informações sobre como usar dados exportados do Aurora com os produtos de ML da AWS, consulte Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado).

MANIFEST {ON | OFF}

Indica se um arquivo de manifesto é criado no Amazon S3. O arquivo manifesto é um arquivo JavaScript Object Notation (JSON) que pode ser usado para carregar dados em um cluster de banco de dados Aurora com a instrução `LOAD DATA FROM S3 MANIFEST`. Para obter mais informações sobre o `LOAD DATA FROM S3 MANIFEST`, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.

Se `MANIFEST ON` for especificado na consulta, o arquivo manifesto será criado no Amazon S3 depois que todos os arquivos de dados tiverem sido criados e carregados. O arquivo manifesto é criado usando o seguinte caminho:

```
s3-region://bucket-name/file-prefix.manifest
```

Para ter mais informações sobre o formato do conteúdo do arquivo manifesto, consulte Criar um manifesto para listar arquivos de dados.

OVERWRITE {ON | OFF}

Indica se os arquivos existentes no bucket especificado do Amazon S3 são substituídos. Se `OVERWRITE ON` for especificado, os arquivos existentes que corresponderem ao prefixo de arquivo no URI especificado em `s3-uri` são substituídos. Caso contrário, ocorrerá um erro.

CRIPTOGRAFIA {LIGADO | DESLIGADO | SSE_S3 | SSE_KMS ['*cmk_id*']}

Indica se deve usar criptografia do lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) ou (SSE-S3) ou AWS KMS keys (SSE-KMS, incluindo Chaves gerenciadas pela AWS e chaves gerenciadas pelo cliente). As configurações `SSE_S3` e `SSE_KMS` estão disponíveis no Aurora MySQL versão 3.05 e posteriores.

Você também pode usar a variável de sessão `aurora_select_into_s3_encryption_default` em vez da cláusula `ENCRYPTION`, conforme mostrado no exemplo a seguir. Use a cláusula SQL ou a variável de sessão, mas não ambas.

```
set session set session aurora_select_into_s3_encryption_default={ON | OFF | SSE_S3
 | SSE_KMS};
```

As configurações `SSE_S3` e `SSE_KMS` estão disponíveis no Aurora MySQL versão 3.05 e posteriores.

Quando você define `aurora_select_into_s3_encryption_default` com o seguinte valor:

- `OFF`: a política de criptografia padrão do bucket do S3 é seguida. O valor padrão de `aurora_select_into_s3_encryption_default` é `OFF`.
- `ON` ou `SSE_S3`: o objeto do S3 é criptografado usando chaves gerenciadas pelo Amazon S3 (SSE-S3).
- `SSE_KMS`: o objeto do S3 é criptografado usando uma chave AWS KMS key.

  Nesse caso, você também inclui a variável de sessão `aurora_s3_default_cmk_id`, por exemplo:

  ```
  set session aurora_select_into_s3_encryption_default={SSE_KMS};
  set session aurora_s3_default_cmk_id={NULL | 'cmk_id'};
  ```

  - Quando `aurora_s3_default_cmk_id` é `NULL`, o objeto S3 é criptografado usando uma chave Chave gerenciada pela AWS.
  - Quando `aurora_s3_default_cmk_id` é uma string `cmk_id` não vazia, o objeto S3 é criptografado usando uma chave gerenciada pelo cliente.

    O valor de `cmk_id` pode ser uma string vazia.

Quando você usa o comando `SELECT INTO OUTFILE S3`, o Aurora determina a criptografia da seguinte forma:

- Se a cláusula `ENCRYPTION` estiver presente no comando SQL, o Aurora vai se basear somente no valor de `ENCRYPTION` e não usará uma variável de sessão.
- Se a cláusula `ENCRYPTION` não estiver presente, o Aurora vai se basear no valor da variável de sessão.

Para obter mais informações, consulte Usar a criptografia no lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) e Usar a criptografia no lado do servidor com as chaves do AWS KMS (SSE-KMS) no Guia do usuário do Amazon Simple Storage Service.

Você pode encontrar mais detalhes sobre outros parâmetros em Instrução SELECT e Instrução LOAD DATA, na documentação do MySQL.

Considerações

O número de arquivos gravados no bucket do Amazon S3 depende da quantidade de dados selecionados pela instrução `SELECT INTO OUTFILE S3` e do limite de tamanho do arquivo do Aurora MySQL. O limite de tamanho de arquivo padrão é de 6 gigabytes (GB). Se os dados selecionados pela instrução forem inferiores ao limite do tamanho do arquivo, um único arquivo será criado; caso contrário, vários arquivos serão criados. Outras considerações para os arquivos criados por esta instrução incluem:

- O Aurora MySQL garante que as linhas em arquivos de dados não sejam divididas em limites de arquivos. Para vários arquivos, o tamanho de cada arquivo de dados, exceto o último, geralmente está próximo ao limite do tamanho do arquivo. No entanto, ficar ocasionalmente abaixo do limite de tamanho do arquivo faz com que uma linha seja dividida em dois arquivos de dados. Nesse caso, o Aurora MySQL cria um arquivo de dados que mantém a linha intacta, mas pode ser maior que o limite do tamanho do arquivo.
- Como cada instrução `SELECT` no Aurora MySQL é executada como uma transação atômica, uma instrução `SELECT INTO OUTFILE S3` que seleciona um conjunto de dados grande pode ser executada por um certo tempo. Se a instrução falhar por qualquer motivo, talvez seja necessário reiniciar e emiti-la novamente. Entretanto, se a instrução falhar, os arquivos já carregados no Amazon S3 permanecerão no bucket do Amazon S3 especificado. Você pode usar outra instrução para fazer upload dos dados restantes em vez de começar de novo.
- Se a quantidade de dados a serem selecionados for grande (mais de 25 GB), recomendamos que você use várias instruções `SELECT INTO OUTFILE S3` para salvar os dados no Amazon S3. Cada instrução deve selecionar uma porção diferente dos dados a serem salvos e também especificar um `file_prefix` diferente no parâmetro `s3-uri` a ser usado ao salvar os arquivos de dados. Particionar os dados a serem selecionados com várias instruções facilita a recuperação de um erro em uma instrução. Se ocorrer um erro para uma instrução, apenas uma parte dos dados precisa ser selecionada novamente e carregada no Amazon S3. O uso de várias instruções também ajuda a evitar uma única transação de longa execução, o que pode melhorar a performance.

- Se várias instruções `SELECT INTO OUTFILE S3` que usam o mesmo `file_prefix` no parâmetro `s3-uri` forem executadas em paralelo para selecionar dados no Amazon S3, o comportamento será indefinido.
- Metadados, como o esquema de tabela ou metadados de arquivos, não são carregados pelo Aurora MySQL no Amazon S3.
- Em alguns casos, você pode executar novamente uma consulta `SELECT INTO OUTFILE S3`, de modo a se recuperar de uma falha. Nesses casos, você deve remover todos os arquivos de dados existentes no bucket do Amazon S3 com o mesmo prefixo de arquivo especificado em `s3-uri` ou incluir `OVERWRITE ON` na consulta `SELECT INTO OUTFILE S3`.

A instrução `SELECT INTO OUTFILE S3` retorna um número de erro típico e uma resposta MySQL em caso de êxito ou falha. Se você não tiver acesso ao número de erro e à resposta MySQL, a maneira mais fácil de determinar a conclusão é especificando `MANIFEST ON` na instrução. O arquivo manifesto é o último arquivo escrito pela instrução. Em outras palavras, se você tiver um arquivo manifesto, a instrução foi concluída.

Atualmente, não há como monitorar diretamente o progresso da instrução `SELECT INTO OUTFILE S3` durante a execução. No entanto, suponha que você esteja gravando uma grande quantidade de dados do Aurora MySQL no Amazon S3 usando essa instrução e conheça a quantidade de dados selecionados pela instrução. Nesse caso, você pode estimar o progresso monitorando a criação de arquivos de dados no Amazon S3.

Para fazer isso, você pode usar o fato de que um arquivo de dados é criado no bucket do Amazon S3 especificado para cerca de cada 6 GB de dados selecionados pela instrução. Divida o tamanho dos dados selecionados por 6 GB para obter o número estimado de arquivos de dados a serem criados. Depois disso, é possível estimar o progresso da instrução monitorando o número de arquivos carregados no Amazon S3 durante a execução da instrução.

Exemplos

A seguinte instrução seleciona todos os dados na tabela `employees` e os salva em um bucket do Amazon S3 que está em uma região diferente do cluster de banco de dados do Aurora MySQL. A instrução cria arquivos de dados nos quais cada campo é encerrado por um caractere de vírgula (`,`) e cada linha é encerrada por um caractere de nova linha (`\n`). A instrução retornará um erro se os arquivos que corresponderem ao prefixo do arquivo `sample_employee_data` existirem no bucket do Amazon S3 especificado.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

A seguinte instrução seleciona todos os dados na tabela `employees` e os salva em um bucket do Amazon S3 que está na mesma região que o cluster de banco de dados do Aurora MySQL. A instrução cria arquivos de dados nos quais cada campo é encerrado por um caractere de vírgula (`,`) e cada linha é encerrada por um caractere de nova linha (`\n`), bem como um arquivo manifesto. A instrução retornará um erro se os arquivos que corresponderem ao prefixo do arquivo `sample_employee_data` existirem no bucket do Amazon S3 especificado.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

A seguinte instrução seleciona todos os dados na tabela `employees` e os salva em um bucket do Amazon S3 que está em uma região diferente do cluster de banco de dados do Aurora. A instrução cria arquivos de dados nos quais cada campo é encerrado por um caractere de vírgula (`,`) e cada linha é encerrada por um caractere de nova linha (`\n`). A instrução substitui todos os arquivos existentes que correspondem ao prefixo do arquivo `sample_employee_data` no bucket do Amazon S3 especificado.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    OVERWRITE ON;
```

A seguinte instrução seleciona todos os dados na tabela `employees` e os salva em um bucket do Amazon S3 que está na mesma região que o cluster de banco de dados do Aurora MySQL. A instrução cria arquivos de dados nos quais cada campo é encerrado por um caractere de vírgula (`,`) e cada linha é encerrada por um caractere de nova linha (`\n`), bem como um arquivo manifesto. A instrução substitui todos os arquivos existentes que correspondem ao prefixo do arquivo `sample_employee_data` no bucket do Amazon S3 especificado.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL

É possível invocar uma função do AWS Lambda em um cluster de banco de dados Amazon Aurora, edição compatível com MySQL, com uma função `lambda_sync` ou `lambda_async` nativa. Antes de invocar uma função do Lambda em um Aurora MySQL, o cluster de banco de dados Aurora deve ter acesso ao Lambda. Para obter detalhes sobre como conceder acesso para o Aurora MySQL, consulte Concessão de acesso ao Lambda para o Aurora. Para obter informações sobre as funções armazenadas `lambda_sync` e `lambda_async`, consulte Como invocar uma função do Lambda com uma função nativa do Aurora MySQL.

Você também pode chamar uma função do AWS Lambda utilizando um procedimento armazenado. No entanto, o uso de um procedimento armazenado está defasado. É altamente recomendável usar uma função nativa do Aurora MySQL se você estiver usando uma das seguintes versões do Aurora MySQL:

- Aurora MySQL versão 2 para clusters compatíveis com o MySQL 5.7.
- Aurora MySQL versão 3.01 e posteriores, para clusters compatíveis com o MySQL 8.0. O procedimento armazenado não está disponível no Aurora MySQL versão 3.

Para ter informações sobre como conceder ao Aurora acesso ao Lambda e invocar uma função do Lambda, consulte os tópicos a seguir.

Tópicos

- Concessão de acesso ao Lambda para o Aurora
- Como invocar uma função do Lambda com uma função nativa do Aurora MySQL
- Invocar uma função do Lambda com um procedimento armazenado do Aurora MySQL (defasado)

Concessão de acesso ao Lambda para o Aurora

Para poder invocar funções do Lambda em um cluster de banco de dados Aurora MySQL, você deve primeiro fornecer permissão ao seu cluster para acessar o Lambda.

Para conceder acesso ao Lambda para o Aurora MySQL

1. Crie uma política do AWS Identity and Access Management (IAM) que forneça as permissões que permitem que o cluster de banco de dados Aurora MySQL invoque funções do Lambda. Para obter instruções, consulte Criar uma política do IAM para acessar recursos do AWS Lambda.
2. Crie um perfil do IAM e anexe a política do IAM criada em Criar uma política do IAM para acessar recursos do AWS Lambda ao novo perfil do IAM. Para obter instruções, consulte Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS.
3. Defina o parâmetro de cluster de banco de dados `aws_default_lambda_role` como o nome de recurso da Amazon (ARN) do novo perfil do IAM.

   Se o cluster fizer parte de um banco de dados global Aurora, aplique a mesma configuração para cada cluster do Aurora no banco de dados global.

   Para obter mais informações sobre parâmetros de cluster de banco de dados, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora.
4. Para permitir que os usuários do banco de dados em um cluster de banco de dados Aurora MySQL invoquem funções do Lambda, associe a função que você criou em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS ao cluster de banco de dados. Para obter informações sobre como associar um perfil do IAM a um cluster de banco de dados, consulte Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.

   Se o cluster fizer parte de um banco de dados global Aurora, associe a função a cada cluster do Aurora no banco de dados global.
5. Configure o cluster de banco de dados Aurora MySQL para permitir conexões de saída com o Lambda. Para obter instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

   Se o cluster fizer parte de um banco de dados global Aurora, habilite conexões de saída para cada cluster do Aurora no banco de dados global.

Como invocar uma função do Lambda com uma função nativa do Aurora MySQL

> **Note**
>
> É possível chamar as funções nativas `lambda_sync` e `lambda_async` ao utilizar o Aurora MySQL versão 2 ou o Aurora MySQL versão 3.01 e posteriores. Para obter mais informações sobre as versões do Aurora MySQL, consult Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

Você pode invocar uma função do AWS Lambda a partir de um cluster de banco de dados Aurora MySQL chamando as funções nativas `lambda_sync` e `lambda_async`. Essa abordagem pode ser útil quando você desejar integrar o banco de dados em execução no Aurora MySQL a outros produtos da AWS. Por exemplo, você pode querer enviar uma notificação usando o Amazon Simple Notification Service (Amazon SNS) sempre que uma linha for inserida em uma tabela específica no seu banco de dados.

Sumário

- Como trabalhar com funções nativas para uma função do Lambda
  - Conceder o perfil no Aurora MySQL versão 3
  - Conceder o privilégio no Aurora MySQL versão 2
  - Sintaxe para a função lambda_sync
  - Parâmetros para a função lambda_sync
  - Exemplo da função lambda_sync
  - Sintaxe para a função lambda_async
  - Parâmetros para a função lambda_async
  - Exemplo da função lambda_async
  - Invocar uma função do Lambda em um gatilho

Como trabalhar com funções nativas para uma função do Lambda

As funções `lambda_sync` e `lambda_async` são funções nativas incorporadas que invocam uma função do Lambda de forma síncrona ou assíncrona. Quando for necessário saber o resultado da função do Lambda antes de passar para outra ação, use a função síncrona `lambda_sync`. Quando

não for necessário saber o resultado da função Lambda antes de passar para outra ação, use a função assíncrona `lambda_async`.

Conceder o perfil no Aurora MySQL versão 3

No Aurora MySQL versão 3, o usuário que invoca uma função nativa deve receber a função `AWS_LAMBDA_ACCESS`. Para conceder essa função a um usuário, conecte-se à instância de banco de dados como usuário administrativo e execute o seguinte comando.

```
GRANT AWS_LAMBDA_ACCESS TO user@domain-or-ip-address
```

É possível revogar essa função executando a seguinte instrução.

```
REVOKE AWS_LAMBDA_ACCESS FROM user@domain-or-ip-address
```

> **Tip**
>
> Ao utilizar a técnica de perfil no Aurora MySQL versão 3, você também pode ativar o perfil usando a instrução `SET ROLE role_name` ou `SET ROLE ALL`. Se não estiver familiarizado com o sistema de funções do MySQL 8.0, é possível saber mais em Modelo de privilégios baseados em funções. Para ver mais detalhes, consulte Using roles no MySQL Reference Manual.
>
> Isso se aplica somente à sessão ativa atual. Ao se reconectar, execute a declaração `SET ROLE` novamente para conceder privilégios. Para ter mais informações, consulte a instrução SET ROLE no Guia de referência do MySQL.
>
> Você pode usar o parâmetro `activate_all_roles_on_login` de cluster de banco de dados para ativar automaticamente todos os perfis quando um usuário se conecta a uma instância de banco de dados. Quando esse parâmetro está definido, não é necessário chamar a declaração `SET ROLE` explicitamente para ativar um perfil. Para ter mais informações, consulte activate_all_roles_on_login no Guia de referência do MySQL.
>
> No entanto, é necessário chamar `SET ROLE ALL` explicitamente no início de um procedimento armazenado para ativar o perfil, quando o procedimento armazenado é chamado por um usuário diferente.

Se você receber um erro como o seguinte ao tentar invocar uma função do Lambda, execute uma instrução `SET ROLE`.

```
SQL Error [1227] [42000]: Access denied; you need (at least one of) the Invoke Lambda
 privilege(s) for this operation
```

Verifique se você está concedendo o perfil ao usuário correto, conforme mostrado nas entradas da tabela `mysql.users`. Pode haver vários usuários com o mesmo nome, mas em hosts diferentes. Dependendo de qual aplicação ou host está invocando a função `lambda_sync`, o MySQL seleciona o usuário com a melhor correspondência de acordo com as entradas da coluna `host`.

Conceder o privilégio no Aurora MySQL versão 2

No Aurora MySQL versão 2, o usuário que invoca uma função nativa deve receber o privilégio `INVOKE LAMBDA`. Para conceder esse privilégio a um usuário, conecte-se à instância de banco de dados como usuário administrativo e execute o seguinte comando.

```
GRANT INVOKE LAMBDA ON *.* TO user@domain-or-ip-address
```

Você pode revogar esse privilégio executando o seguinte comando.

```
REVOKE INVOKE LAMBDA ON *.* FROM user@domain-or-ip-address
```

Sintaxe para a função lambda_sync

Invoque a função `lambda_sync` de forma síncrona com o tipo de invocação `RequestResponse`. A função retorna o resultado da invocação do Lambda em uma carga JSON. A função tem a seguinte sintaxe.

```
lambda_sync (
  lambda_function_ARN,
  JSON_payload
)
```

Parâmetros para a função lambda_sync

A função `lambda_sync` tem os parâmetros a seguir.

lambda_function_ARN

O nome de recurso da Amazon (ARN) da função Lambda a ser invocada.

JSON_payload

A carga para a função do Lambda, no formato JSON.

> **Note**
>
> O Aurora MySQL versão 3 oferece suporte às funções de análise JSON do MySQL 8.0. No entanto, o Aurora MySQL versão 2 não inclui essas funções. A análise JSON não é necessária quando uma função do Lambda retorna um valor atômico, como um número ou uma string.

Exemplo da função lambda_sync

A consulta a seguir, baseada em `lambda_sync`, invoca a função do Lambda `BasicTestLambda` de forma síncrona usando o ARN da função. A carga para a função é `{"operation": "ping"}`.

```
SELECT lambda_sync(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Sintaxe para a função lambda_async

Invoque a função `lambda_async` de forma assíncrona com o tipo de invocação `Event`. A função retorna o resultado da invocação do Lambda em uma carga JSON. A função tem a seguinte sintaxe.

```
lambda_async (
  lambda_function_ARN,
  JSON_payload
)
```

Parâmetros para a função lambda_async

A função `lambda_async` tem os parâmetros a seguir.

lambda_function_ARN

O nome de recurso da Amazon (ARN) da função Lambda a ser invocada.

JSON_payload

A carga para a função do Lambda, no formato JSON.

> **Note**
>
> O Aurora MySQL versão 3 oferece suporte às funções de análise JSON do MySQL 8.0. No entanto, o Aurora MySQL versão 2 não inclui essas funções. A análise JSON não é necessária quando uma função do Lambda retorna um valor atômico, como um número ou uma string.

Exemplo da função lambda_async

A consulta a seguir, baseada em `lambda_async`, invoca a função do Lambda `BasicTestLambda` de forma assíncrona usando o ARN da função. A carga para a função é `{"operation": "ping"}`.

```
SELECT lambda_async(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Invocar uma função do Lambda em um gatilho

Você pode usar triggers para chamar o Lambda em instruções de modificação de dados. O exemplo a seguir usa a função `lambda_async` nativa e armazena o resultado em uma variável.

```
mysql>SET @result=0;
mysql>DELIMITER //
mysql>CREATE TRIGGER myFirstTrigger
      AFTER INSERT
          ON Test_trigger FOR EACH ROW
      BEGIN
      SELECT lambda_async(
          'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
          '{"operation": "ping"}')
          INTO @result;
      END; //
mysql>DELIMITER ;
```

> **Note**
>
> Os gatilhos não são executados uma vez a cada instrução SQL, mas uma vez por linha alterada, uma linha por vez. Quando um trigger é executado, o processo é síncrono. A instrução de modificação de dados só retorna quando o trigger é concluído.

Tenha cuidado ao invocar uma função do AWS Lambda de triggers em tabelas que apresentam alto tráfego de gravação. Os triggers `INSERT`, `UPDATE` e `DELETE` são ativados por linha. Uma workload com muitas gravações em uma tabela com triggers `INSERT`, `UPDATE` ou `DELETE` resulta em um grande número de chamadas para a sua função do AWS Lambda.

Invocar uma função do Lambda com um procedimento armazenado do Aurora MySQL (defasado)

Você pode invocar uma função do AWS Lambda a partir de um cluster de banco de dados Aurora MySQL chamando o procedimento `mysql.lambda_async`. Essa abordagem pode ser útil quando você desejar integrar o banco de dados em execução no Aurora MySQL a outros produtos da AWS. Por exemplo, você pode querer enviar uma notificação usando o Amazon Simple Notification Service (Amazon SNS) sempre que uma linha for inserida em uma tabela específica no seu banco de dados.

Sumário

- Considerações da versão do Aurora MySQL
- Trabalhar com o procedimento mysql.lambda_async para invocar uma função do Lambda (defasado)
  - Sintaxe
  - Parâmetros
  - Exemplos

Considerações da versão do Aurora MySQL

A partir do Aurora MySQL versão 2, é possível usar o método da função nativa em vez desses procedimentos armazenados para invocar uma função do Lambda. Para obter mais informações sobre as funções nativas, consulte Como trabalhar com funções nativas para uma função do Lambda.

No Aurora MySQL versão 2, o procedimento armazenado `mysql.lambda_async` não é mais compatível. Em vez disso, é altamente recomendável o uso de funções nativas do Lambda.

No Aurora MySQL versão 3, o procedimento armazenado não está disponível.

Trabalhar com o procedimento mysql.lambda_async para invocar uma função do Lambda (defasado)

O procedimento `mysql.lambda_async` é um procedimento armazenado interno que invoca uma função do Lambda de forma assíncrona. Para usar esse procedimento, o usuário do banco de dados deve ter privilégios `EXECUTE` no procedimento armazenado `mysql.lambda_async`.

Sintaxe

O procedimento `mysql.lambda_async` tem a seguinte sintaxe.

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Parâmetros

O procedimento `mysql.lambda_async` tem os seguintes parâmetros.

lambda_function_ARN

O nome de recurso da Amazon (ARN) da função Lambda a ser invocada.

lambda_function_input

A string de entrada, no formato JSON, para a função Lambda invocada.

Exemplos

Como melhores práticas, recomendamos que você faça chamadas do procedimento `mysql.lambda_async` em um procedimento armazenado que possa ser chamado de diferentes origens, como disparadores ou código de cliente. Essa abordagem pode ajudar a evitar problemas de incompatibilidade de impedância e facilitar a invocação de funções Lambda.

> **Note**
>
> Tenha cuidado ao invocar uma função do AWS Lambda de triggers em tabelas que apresentam alto tráfego de gravação. Os triggers `INSERT`, `UPDATE` e `DELETE` são ativados por linha. Uma workload com muitas gravações em uma tabela com triggers `INSERT`, `UPDATE` ou `DELETE` resulta em um grande número de chamadas para a sua função do AWS Lambda.

Embora as chamadas para o procedimento `mysql.lambda_async` sejam assíncronas, os triggers são síncronos. Uma instrução que resulta em um grande número de ativações de triggers não aguarda a conclusão da chamada para a função do AWS Lambda, mas aguarda a conclusão dos triggers antes de retornar o controle ao cliente.

Example Exemplo: invocar uma função do AWS Lambda para enviar e-mail

O exemplo a seguir cria um procedimento armazenado que você pode chamar no seu código de banco de dados para enviar um e-mail usando uma função Lambda.

AWS Lambda Função do

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Procedimento armazenado

```
DROP PROCEDURE IF EXISTS SES_send_email;
```

```
DELIMITER ;;
  CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                  IN email_to VARCHAR(255),
                                  IN subject VARCHAR(255),
                                  IN body TEXT) LANGUAGE SQL
  BEGIN
    CALL mysql.lambda_async(
         'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
         CONCAT('{"email_to" : "', email_to,
             '", "email_from" : "', email_from,
             '", "email_subject" : "', subject,
             '", "email_body" : "', body, '"}')
     );
  END
  ;;
DELIMITER ;
```

Chamar o procedimento armazenado para invocar a função do AWS Lambda

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Exemplo: invocar uma função do AWS Lambda para publicar um evento a partir de um trigger

O exemplo a seguir cria um procedimento armazenado que publica um evento usando o Amazon SNS. O código chama o procedimento de um trigger quando uma linha é adicionada a uma tabela.

AWS Lambda Função do

```
import boto3

sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Procedimento armazenado

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Tabela

```
CREATE TABLE 'Customer_Feedback' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'customer_name' varchar(255) NOT NULL,
  'customer_feedback' varchar(1024) NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Trigger

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name),
 NEW.customer_feedback INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
;;
DELIMITER ;
```

Inserir uma linha na tabela para disparar a notificação

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs

É possível configurar seu cluster de banco de dados Aurora MySQL para publicar dados de logs gerais, lentos, de auditoria e de erros em um grupo de logs no Amazon CloudWatch Logs. Com o CloudWatch Logs, você pode executar análise em tempo real de dados de log e usar o CloudWatch para criar alarmes e visualizar métricas. Você pode usar o CloudWatch Logs para armazenar seus registros de log em armazenamento resiliente.

Para publicar logs no CloudWatch Logs, os respectivos registros devem estar ativados. Os logs de erro estão ativados por padrão, mas é necessário ativar outros tipos de logs explicitamente. Para obter informações sobre como habilitar logs no MySQL, consulte Selecionar destinos de resultados de logs de consultas gerais e de consultas lentas na documentação do MySQL. Para obter mais informações sobre como habilitar logs de auditoria do Aurora MySQL, consulte Como habilitar a Auditoria avançada.

> **Note**
>
> - Se a exportação de dados de log estiver desativada, o Aurora não excluirá grupos de logs existentes nem fluxos de logs. Se a exportação de dados de log estiver desativada, os dados de log existentes permanecem disponíveis no CloudWatch Logs, dependendo da retenção de log, e você ainda será cobrado por dados de log de auditoria armazenados. Você pode excluir fluxos de log e grupos de logs usando o console do CloudWatch Logs, a AWS CLI ou a API do CloudWatch Logs.
> - Uma maneira alternativa de publicar logs de auditoria no CloudWatch Logs é habilitar a Auditoria avançada, depois, criar um grupo de parâmetros de cluster de banco de dados personalizado e definir o parâmetro `server_audit_logs_upload` como `1`. O padrão para o parâmetro do cluster de banco de dados `server_audit_logs_upload` é `0`. Para ter informações sobre como habilitar a Auditoria avançada, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.
>
>   Se usar esse método alternativo, você deverá tem uma função do IAM para acessar o CloudWatch Logs e definir o parâmetro `aws_default_logs_role` em nível de cluster como o ARN para essa função. Para obter informações sobre como criar a função do , consulte Configurar funções do IAM para acessar outros produtos da AWS. Contudo,

se você tiver a função `AWSServiceRoleForRDS` vinculada ao serviço, ela fornecerá acesso ao CloudWatch Logs e substituirá todas as funções personalizadas. Para obter informações sobre funções vinculadas ao serviço do Amazon RDS, consulte Usar funções vinculadas ao serviço do Amazon Aurora.

- Se você não quiser exportar logs de auditoria para o CloudWatch Logs, certifique-se de que todos os métodos de exportação de logs de auditoria estejam desativados. Estes métodos são o Console de gerenciamento da AWS, a AWS CLI, a API do RDS e o parâmetro `server_audit_logs_upload`.
- O procedimento para clusters de banco de dados do Aurora Serverless v1 é um pouco diferente do procedimento para clusters de banco de dados com instâncias provisionadas ou de banco de dados do Aurora Serverless v2. Os clusters do Aurora Serverless v1 carregam automaticamente todos os logs habilitados por meio dos parâmetros de configuração.

  Portanto, você ativa ou desativa o upload de logs para clusters de banco de dados do Aurora Serverless v1 ativando e desativando diferentes tipos de log no grupo de parâmetros do cluster de banco de dados. Você não modifica as configurações do cluster por meio do Console de gerenciamento da AWS, da AWS CLI ou da API do RDS. Para obter informações sobre como ativar e desativar logs do MySQL para clusters do Aurora Serverless v1, consulte Grupos de parâmetros para Aurora Serverless v1.

Console

Você pode publicar logs do Aurora MySQL para clusters provisionados no CloudWatch Logs com o console.

Para publicar logs do Aurora MySQL a partir do console

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione o cluster de banco de dados Aurora MySQL no qual deseja publicar dados de log.
4. Selecione Modify.
5. Na seção Log exports (Exportações de log), escolha os logs que deseja começar a publicar no CloudWatch Logs.
6. Escolha Continue (Continuar) e depois escolha Modify DB Cluster (Modificar cluster de banco de dados) na página de resumo.

AWS CLI

É possível publicar logs do Aurora MySQL para clusters provisionados com a AWS CLI. Você pode executar o comando modify-db-cluster da AWS CLI com as seguintes opções:

- `--db-cluster-identifier`—O identificador de cluster de banco de dados.
- `--cloudwatch-logs-export-configuration` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Também é possível publicar logs do Aurora MySQL executando um dos seguintes comandos da AWS CLI:

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Execute um destes comandos da AWS CLI com as seguintes opções:

- `--db-cluster-identifier`—O identificador de cluster de banco de dados.
- `--engine` — o mecanismo de banco de dados.
- `--enable-cloudwatch-logs-exports` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Outras opções podem ser exigidas dependendo do comando da AWS CLI que você executa.

Example

O comando a seguir altera um cluster existente do banco de dados Aurora MySQL para publicar arquivos de log no CloudWatch Logs.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Example

O comando a seguir cria um cluster de banco de dados Aurora MySQL para publicar arquivos de log no CloudWatch Logs.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine aurora \
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine aurora ^
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

API do RDS

É possível publicar logs do Aurora MySQL para clusters provisionados com a API do RDS. Para fazer isso, execute a operação ModifyDBCluster com as seguintes opções:

- `DBClusterIdentifier`—O identificador de cluster de banco de dados.
- `CloudwatchLogsExportConfiguration` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Também é possível publicar logs do Aurora MySQL com a API do RDS executando uma das seguintes operações da API do RDS:

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Execute a operação da API do RDS com os seguintes parâmetros:

- `DBClusterIdentifier`—O identificador de cluster de banco de dados.
- `Engine` — o mecanismo de banco de dados.
- `EnableCloudwatchLogsExports` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Outros parâmetros podem ser exigidos dependendo do comando da AWS CLI que você executa.

Monitorar eventos de log no Amazon CloudWatch

Depois de habilitar os eventos de log do Aurora MySQL, é possível monitorá-los no Amazon CloudWatch Logs. Um novo grupo de logs é criado automaticamente para o cluster de banco de dados Aurora com o seguinte prefixo, em que *`cluster-name`* representa o nome do cluster de banco de dados e *`log_type`* representa o tipo de log.

```
/aws/rds/cluster/cluster-name/log_type
```

Por exemplo, se você configurar a função de exportação para incluir o log de consultas lentas em um cluster de banco de dados denominado `mydbcluster`, os dados de consultas lentas serão armazenados no grupo de logs `/aws/rds/cluster/mydbcluster/slowquery`.

Os eventos de todas as instâncias em seu cluster são enviadospara um grupo de log usando diferentes fluxos de log. O comportamento depende de qual das seguintes condições é verdadeira:

- Um grupo de logs com o nome especificado já existe.

  O Aurora usará o grupo de logs existente para exportar dados de log para o cluster. Para criar grupos de log com períodos de retenção de log, filtros de métricas e acesso de clientes predefinidos, você pode usar a configuração automatizada como AWS CloudFormation.

- Um grupo de logs com o nome especificado não existe.

  Quando uma entrada de log correspondente é detectada no arquivo de log da instância, o Aurora MySQL cria um novo grupo de logs no CloudWatch Logs automaticamente. O grupo de logs usa o período de retenção de log padrão de Never Expire (Nunca expira).

  Para alterar o período de retenção do log, use o console do CloudWatch Logs, a AWS CLI ou a API do CloudWatch Logs. Para obter mais informações sobre alteração de períodos de retenção de log no CloudWatch Logs, consulte Alterar a retenção do log de dados no CloudWatch Logs.

Para pesquisar informações nos eventos de log para um cluster de banco de dados, use o console do CloudWatch Logs, a AWS CLI ou a API do CloudWatch Logs. Para obter mais informações sobre como procurar e filtrar dados de log, consulte Procurar e filtrar dados de log.

Modo de laboratório do Amazon Aurora MySQL

> **Important**
>
> O modo de laboratório foi introduzido no Aurora MySQL versão 2 para habilitar a otimização da DDL rápida, o que melhorou a eficiência de certas operações de DDL. No Aurora MySQL versão 3, o modo de laboratório foi removido e a DDL rápida foi substituída pelo recurso do MySQL 8.0 chamado DDL instantânea.

O modo de laboratório do Aurora é usado para habilitar recursos do Aurora que estão disponíveis na versão atual do banco de dados do Aurora, mas que não estão habilitados por padrão. Embora os recursos em modo de laboratório do Aurora não sejam recomendados para uso em clusters de banco de dados de produção, você pode usar o modo de laboratório do Aurora para habilitar esses recursos para clusters de banco de dados nos ambientes de desenvolvimento e teste. Para obter mais informações sobre os recursos do Aurora disponíveis quando o modo de laboratório do Aurora estiver habilitado, consulte Recursos em modo de laboratório do Aurora.

O parâmetro `aurora_lab_mode` é um parâmetro em nível de instância que se encontra no parameter group padrão. O parâmetro é definido como `0` (desabilitado) no parameter group padrão. Para habilitar o modo de laboratório do Aurora, crie um grupo de parâmetros personalizado, configure o parâmetro `aurora_lab_mode` como `1` (habilitado) no grupo de parâmetros personalizado e modifique uma ou mais instâncias de banco de dados em seu cluster do Aurora para usar o grupo de parâmetros personalizado. Depois disso, conecte o endpoint de instância adequado para experimentar os recursos do modo de laboratório. Para obter informações sobre como modificar um parameter group de banco de dados, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora. Para obter informações sobre parameter groups e o Amazon Aurora, consulte Parâmetros de configuração do Aurora MySQL.

Recursos em modo de laboratório do Aurora

No momento, o recurso do Aurora a seguir está disponível quando o modo de laboratório do Aurora está ativado. Você deverá habilitar o modo de laboratório do Aurora para qualquer um desses recursos ser usado.

DDL rápido

Esse recurso permite executar uma operação `ALTER TABLE tbl_name ADD COLUMN col_name column_definition` quase instantaneamente. A operação é concluída sem exigir que a tabela seja copiada e sem causar impacto material em outras instruções DML. Como ela não consome armazenamento temporário para a cópia da tabela, as declarações de DDL são práticas mesmo para tabelas grandes em classes de instância pequenas.

O DDL rápido atualmente tem suporte apenas para adicionar uma coluna anulável, sem um valor padrão, no final de uma tabela. Para obter mais informações sobre o uso desse recurso, consulte Alterar tabelas no Amazon Aurora usando a DDL rápida.

Melhores práticas do Amazon Aurora MySQL

Este tópico inclui informações sobre as melhores práticas e opções para usar ou migrar dados para um cluster de banco de dados Amazon Aurora MySQL. As informações neste tópico resumem e reiteram algumas das diretrizes e procedimentos que você pode encontrar em Como gerenciar um cluster de banco de dados do Amazon Aurora.

Sumário

- Como determinar a qual instância de banco de dados você está conectado
- Práticas recomendadas para a performance e escalabilidade do Aurora MySQL
  - Uso de classes de instância T para desenvolvimento e testes
  - Otimizar consultas de junção indexadas do Aurora MySQL com pré-busca de chave assíncrona
    - Como habilitar a pré-busca de chave assíncrona
    - Como otimizar consultas de pré-busca de chave assíncrona
  - Otimizando grandes consultas de junção do Aurora MySQL com junções hash
    - Permitir hash joins
    - Otimizar consultas para hash joins
  - Como usar o Amazon Aurora para escalar leituras para seu banco de dados MySQL
  - Otimizando as operações de carimbo de data/hora
  - Erros de estouro de ID de índice virtual
- Práticas recomendadas para alta disponibilidade do Aurora MySQL
  - Como utilizar o Amazon Aurora para recuperação de desastres com seus bancos de dados MySQL
  - Migrar do MySQL para o Amazon Aurora MySQL com o tempo de inatividade reduzido
  - Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora MySQL
- Recomendações para recursos do MySQL no Aurora MySQL
  - Utilizar a replicação de vários threads no Aurora MySQL
  - Invocar funções do AWS Lambda com o uso de funções nativas do MySQL
  - Como evitar transações XA com o Amazon Aurora MySQL
  - Como manter chaves externas ativadas durante instruções DML
  - Configurar a frequência com que o buffer de log é liberado

- Minimizar e solucionar problemas de deadlocks do Aurora MySQL
  - Minimizar os deadlocks do InnoDB
  - Monitorar deadlocks do InnoDB
- Avaliação do uso de instâncias de banco de dados para Aurora MySQL com métricas do Amazon CloudWatch

Como determinar a qual instância de banco de dados você está conectado

Para determinar a qual instância de banco de dados dentro de um cluster de banco de dados Aurora MySQL uma conexão está conectada, verifique a variável global `innodb_read_only`, conforme mostrado no exemplo a seguir.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

Se você estiver conectado a uma instância de banco de dados de leitor, a variável `innodb_read_only` será definida como `ON`. Se você estiver conectado a uma instância de banco de dados de gravador, como, por exemplo, uma instância primária em um cluster provisionado, essa configuração será `OFF`.

Essa abordagem pode ser útil se você quiser adicionar lógica ao seu código de aplicação para equilibrar a workload ou para garantir que uma operação de gravação esteja usando a conexão correta.

Práticas recomendadas para a performance e escalabilidade do Aurora MySQL

Você pode aplicar as práticas recomendadas a seguir para melhorar a performance e a escalabilidade dos seus clusters do Aurora MySQL.

Tópicos

- Uso de classes de instância T para desenvolvimento e testes
- Otimizar consultas de junção indexadas do Aurora MySQL com pré-busca de chave assíncrona
- Otimizando grandes consultas de junção do Aurora MySQL com junções hash
- Como usar o Amazon Aurora para escalar leituras para seu banco de dados MySQL
- Otimizando as operações de carimbo de data/hora

- Erros de estouro de ID de índice virtual

Uso de classes de instância T para desenvolvimento e testes

As instâncias do MySQL do Amazon Aurora que usam as classes de instância de banco de dados `db.t2`, `db.t3` ou `db.t4g` são mais adequadas para aplicações que não oferecem suporte a uma workload elevada por um período prolongado. As instâncias T são projetadas para fornecer uma performance de linha de base moderada e capacidade de intermitência para obter uma performance significativamente mais alta, conforme necessário para a sua workload. Elas são destinadas a workloads que não usam a CPU inteira com frequência ou de forma consistente, mas que às vezes precisam de intermitência. Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instância T, consulte Instâncias expansíveis.

Se o seu cluster do Aurora tiver mais que 40 TB, não use classes da instância T. Quando o seu banco de dados tiver um grande volume de dados, a sobrecarga de memória para gerenciar objetos de esquema poderá exceder a capacidade de uma instância T.

Não habilite o esquema de performance do MySQL em instâncias T do Amazon Aurora MySQL. Se o esquema de performance for habilitado, a instância poderá ficar sem memória.

> **Tip**
>
> Se o banco de dados às vezes ficar ocioso, mas outras vezes tiver uma workload substancial, você poderá usar o Aurora Serverless v2 como alternativa às instâncias T. Com o Aurora Serverless v2, você define um intervalo de capacidade e o Aurora reduz ou expande automaticamente o banco de dados, dependendo da workload atual. Para obter mais detalhes sobre uso, consulte Usar o Aurora Serverless v2. Para saber as versões do mecanismo de banco de dados que você pode usar com o Aurora Serverless v2, consulte Requisitos e limitações do Aurora Serverless v2.

Ao usar uma instância T como instância de banco de dados em um cluster de bancos de dados Aurora MySQL, recomendamos o seguinte:

- Use a mesma classe de instância de banco de dados para todas as instâncias em seu cluster de banco de dados. Por exemplo, se você usar `db.t2.medium` para a sua instância de gravador, recomendamos que também use `db.t2.medium` para as suas instâncias de leitor.

- Não ajuste nenhuma configuração relacionada à memória, como, por exemplo, `innodb_buffer_pool_size`. O Aurora usa um conjunto altamente ajustado de valores padrão para buffers de memória em instâncias T. Esses padrões especiais são necessários para que o Aurora seja executado em instâncias com restrições de memória. Se você alterar as configurações relacionadas à memória em uma instância T, o mais provável é que encontre um panorama de memória insuficiente, mesmo que a sua alteração se destine a aumentar tamanhos de buffer.
- Monitore seu saldo de crédito da CPU (`CPUCreditBalance`) para garantir que esteja em um nível sustentável. Ou seja, se os créditos da CPU estão sendo acumulados no mesmo ritmo em que estão sendo usados.

  Quando tiver esgotado os créditos da CPU de uma instância, você verá uma queda imediata na CPU disponível e um aumento na latência de leitura e gravação da instância. Essa situação resulta em uma diminuição elevada no performance geral da instância.

  Se o seu saldo de crédito da CPU não estiver em um nível sustentável, recomendamos que você modifique a sua instância de banco de dados para usar uma das classes de instância de banco de dados R compatíveis (computação de escalabilidade).

  Para ter mais informações sobre as métricas de monitoramento, consulte Visualizar métricas no console do Amazon RDS.
- Monitore o atraso de réplicas (`AuroraReplicaLag`) entre a instância do gravador e as instâncias do leitor.

  Se uma instância de leitor ficar sem créditos de CPU antes da instância de gravador, o consequente atraso poderá gerar reinicializações frequentes na instância de leitor. Esse resultado é comum quando uma aplicação tem uma carga pesada de operações de leitura distribuídas entre as instâncias do leitor, ao mesmo tempo que a instância do gravador tem uma carga mínima de operações de gravação.

  Se você observar um aumento constante no atraso de réplicas, certifique-se de que o seu saldo de crédito da CPU para instâncias do gravador no seu cluster de banco de dados não tenha se esgotado.

  Se o seu saldo de crédito da CPU não estiver em um nível sustentável, recomendamos que você modifique a sua instância de banco de dados para usar uma das classes de instância de banco de dados R compatíveis (computação de escalabilidade).
- Mantenha o número de inserções por transação abaixo de 1 milhão para clusters de banco de dados com log binário habilitado.

Se o grupo de parâmetros do cluster de banco de dados para o seu cluster de banco de dados tiver o parâmetro `binlog_format` configurado com um valor diferente de `OFF`, seu cluster de banco de dados deve enfrentar condições de falta de memória caso o cluster de banco de dados receba transações grandes que contenham mais de 1 milhão de linhas de inserção. Você pode controlar a métrica (`FreeableMemory`) de memória disponível para determinar se o seu cluster de banco de dados está ficando sem memória disponível. Verifique a métrica (`VolumeWriteIOPS`) de operações de gravação para ver se uma instância de gravação está recebendo uma carga pesada de operações de gravação. Se for esse o caso, recomendamos que você atualize sua aplicação para limitar o número de inserções em uma transação para menos de 1 milhão. Como alternativa, você pode modificar a sua instância para usar uma das classes de instância de banco de dados R compatíveis (computação em escala).

Otimizar consultas de junção indexadas do Aurora MySQL com pré-busca de chave assíncrona

O Aurora MySQL pode usar o atributo de pré-busca de chave assíncrona (AKP) para melhorar a performance das consultas de junção de tabelas entre índices. Esse recurso melhora a performance ao antecipar as linhas necessárias para executar consultas nas quais uma consulta JOIN requer o uso do algoritmo de junção de Batched Key Access (BKA - Acesso a chaves em lote) e dos recursos de otimização de Multi-Range Read (MRR - Leitura de vários intervalos). Para obter mais informações sobre BKA e MRR, consulte Block nested-loop and batched key access joins e Multi-range read optimization na documentação do MySQL.

Para usar o recurso AKP, uma consulta deve usar BKA e MRR. Normalmente, essa consulta ocorre quando a cláusula JOIN de uma consulta usa um índice secundário, mas também precisa de algumas colunas do índice primário. Por exemplo, você pode usar a AKP quando uma cláusula JOIN representa uma junção equivalente nos valores de índice entre uma tabela interna grande e uma tabela externa pequena, e o índice é altamente seletivo na tabela maior. A AKP funciona de acordo com o BKA e a MRR para executar uma consulta de índice secundário no primário durante a avaliação da cláusula JOIN. A AKP identifica as linhas necessárias para executar a consulta durante a avaliação da cláusula JOIN. Em seguida, ela usa um thread em segundo plano para carregar de forma assíncrona as páginas contendo as linhas na memória antes de executar a consulta.

O recurso AKP está disponível para Aurora MySQL versão 2.10 e posteriores, e versão 3. Para obter mais informações sobre as versões do Aurora MySQL, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.

Como habilitar a pré-busca de chave assíncrona

Você pode habilitar o recurso de AKP definindo `aurora_use_key_prefetch`, uma variável de servidor MySQL, para on. Por padrão, esse valor é definido como on. Contudo, o recurso AKP não poderá ser habilitado enquanto você não habilitar o algoritmo de junção do BKA e desabilitar a funcionalidade MRR baseada em custo. Para fazer isso, é necessário definir os seguintes valores para `optimizer_switch`, uma variável do servidor MySQL:

- Defina `batched_key_access` como on. Esse valor controla o valor uso do algoritmo de junção do BKA. Por padrão, esse valor é definido como `off`.
- Defina `mrr_cost_based` como `off`. Esse valor controla o uso da funcionalidade MRR baseada no custo. Por padrão, esse valor é definido como on.

No momento, só é possível configurar esses valores no nível da sessão. O exemplo a seguir ilustra como definir esses valores para habilitar a AKP para a sessão atual executando declarações SET.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

Da mesma forma, é possível usar declarações SET para desabilitar a AKP e o algoritmo de junção do BKA e habilitar a funcionalidade MRR baseada no custo novamente para a sessão atual, conforme mostrado no exemplo a seguir.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

Para mais informações sobre os switches otimizadores batched_key_access e mrr_cost_based, consulte Switchable optimizations na documentação do MySQL.

Como otimizar consultas de pré-busca de chave assíncrona

É possível confirmar se uma consulta pode utilizar o recurso AKP. Para fazer isso, use a instrução `EXPLAIN` com a palavra-chave para criar o perfil da consulta antes de executá-la. A declaração `EXPLAIN` fornece informações sobre o plano de execução a ser usado para uma consulta específica.

Na saída da instrução EXPLAIN, a coluna Extra descreve informações adicionais incluídas no plano de execução. Se o recurso AKP se aplica a uma tabela usada na consulta, essa coluna inclui um destes valores:

- Using Key Prefetching
- Using join buffer (Batched Key Access with Key Prefetching)

O exemplo a seguir mostra o uso de EXPLAIN para visualizar o plano de execução de uma consulta que pode utilizar a AKP.

```
mysql> explain select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
    -> order by
    ->     value desc;
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
```

```
| id | select_type | table    | type | possible_keys         | key           | key_len
 | ref                              | rows | filtered | Extra
                              |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where; Using temporary;
 Using filesort                |
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where
                              |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

Para obter mais informações sobre o formato de saída `EXPLAIN`, consulte Extended EXPLAIN output format na documentação do MySQL.

Otimizando grandes consultas de junção do Aurora MySQL com junções hash

Quando você precisa unir uma grande quantidade de dados usando uma junção equivalente, um hash join pode melhorar a performance da consulta. Você pode permitir hash joins para o Aurora MySQL.

Uma coluna de hash join pode ser qualquer expressão complexa. Em uma coluna de hash join, você pode comparar entre tipos de dados nas seguintes formas:

- Você pode comparar tudo na categoria de tipos de dados numéricos precisos, como `int`, `bigint`, `numeric` e `bit`.

- Você pode comparar tudo na categoria de tipos de dados numéricos aproximados, como `float` e `double`.
- Você pode comparar itens entre tipos de string se os tipos de string tiverem o mesmo conjunto de caracteres e agrupamento.
- Você pode comparar itens com tipos de dados de data e hora, se os tipos forem iguais.

> **Note**
>
> Não é possível comparar os tipos de dados em diferentes categorias.

As seguintes restrições se aplicam a hash joins para Aurora MySQL:

- As junções externas da esquerda para a direita não são compatíveis com o Aurora MySQL versão 2, mas são compatíveis com a versão 3.
- Semijunções, como subconsultas, não são compatíveis, a menos que as subconsultas sejam materializadas primeiro.
- Não há compatibilidade para atualizações ou exclusões de várias tabelas.

> **Note**
>
> Há suporte a atualizações ou exclusões de uma única tabela.

- As colunas BLOB e de tipo de dados espaciais não podem ser colunas unidas em um hash join.

Permitir hash joins

Para permitir junções de hash:

- Aurora MySQL versão 2: defina o parâmetro de banco de dados ou o parâmetro de cluster de banco de dados `aurora_disable_hash_join` como `0`. Desativar `aurora_disable_hash_join` define o valor de `optimizer_switch` para `hash_join=on`.
- Aurora MySQL versão 3: defina o parâmetro de servidor MySQL `optimizer_switch` como `block_nested_loop=on`.

Por padrão, as junções de hash estão habilitadas no Aurora MySQL versão 3 e desabilitadas no Aurora MySQL versão 2. O exemplo a seguir ilustra como permitir junções de hash no Aurora MySQL versão 3. É possível enviar a instrução `select @@optimizer_switch` primeiro para ver que outras configurações estão presentes na string de parâmetros `SET`. A ação de atualizar uma configuração no parâmetro `optimizer_switch` não exclui nem modifica as demais configurações.

```
mysql> SET optimizer_switch='block_nested_loop=on';
```

> **Note**
>
> Para o Aurora MySQL versão 3, o suporte para hash joins está disponível em todas as versões secundárias e está habilitado por padrão.
> Para o Aurora MySQL versão 2, o suporte para junções de hash está disponível em todas as versões secundárias. No Aurora MySQL versão 2, o recurso de hash join é sempre controlado pelo valor de `aurora_disable_hash_join`.

Com esta configuração, o otimizador opta por usar uma hash join com base no custo, nas características da consulta e na disponibilidade de recursos. Se a estimativa de custo estiver incorreta, você pode forçar o otimizador a escolher um hash join. Para fazer isso, configure `hash_join_cost_based`, uma variável de servidor MySQL, como `off`. O exemplo a seguir ilustra como forçar o otimizador a escolher um hash join.

```
mysql> SET optimizer_switch='hash_join_cost_based=off';
```

> **Note**
>
> Essa configuração substitui as decisões do otimizador baseado em custos. Embora a configuração possa ser útil para testes e desenvolvimento, recomendamos não usá-la na produção.

Otimizar consultas para hash joins

Para descobrir se uma consulta pode aproveitar uma junção hash, use a instrução `EXPLAIN` para obter um perfil da consulta primeiro. A declaração `EXPLAIN` fornece informações sobre o plano de execução a ser usado para uma consulta específica.

Na saída da instrução EXPLAIN, a coluna Extra descreve informações adicionais incluídas no plano de execução. Se um hash join se aplica às tabelas usadas na consulta, essa coluna inclui valores semelhantes aos seguintes:

- `Using where; Using join buffer (Hash Join Outer table table1_name)`
- `Using where; Using join buffer (Hash Join Inner table table2_name)`

O exemplo a seguir mostra como usar EXPLAIN para visualizar o plano de execução para uma consulta de hash join.

```
mysql> explain SELECT sql_no_cache * FROM hj_small, hj_big, hj_big2
    ->     WHERE hj_small.col1 = hj_big.col1 and hj_big.col1=hj_big2.col1 ORDER BY 1;
+----+-------------+----------+------+---------------+------+---------+------+------
+-------------------------------------------------------------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows |
 Extra                                                              |
+----+-------------+----------+------+---------------+------+---------+------+------
+-------------------------------------------------------------------+
|  1 | SIMPLE      | hj_small | ALL  | NULL          | NULL | NULL    | NULL |    6 |
 Using temporary; Using filesort                                    |
|  1 | SIMPLE      | hj_big   | ALL  | NULL          | NULL | NULL    | NULL |   10 |
 Using where; Using join buffer (Hash Join Outer table hj_big)      |
|  1 | SIMPLE      | hj_big2  | ALL  | NULL          | NULL | NULL    | NULL |   15 |
 Using where; Using join buffer (Hash Join Inner table hj_big2)     |
+----+-------------+----------+------+---------------+------+---------+------+------
+-------------------------------------------------------------------+
3 rows in set (0.04 sec)
```

No resultado, a `Hash Join Inner table` é a tabela usada para criar a tabela hash, e a `Hash Join Outer table` é a tabela usada para testar a tabela hash.

Para ter mais informações sobre o formato de resultado EXPLAIN estendido, consulte Extended EXPLAIN Output Format na documentação do produto do MySQL.

No Aurora MySQL versão 2.08 e posterior, é possível usar dicas SQL para influenciar se uma consulta usa ou não a junção de hash e quais tabelas usar para os lados de compilação e teste da junção. Para obter detalhes, consulte Dicas do Aurora MySQL.

Como usar o Amazon Aurora para escalar leituras para seu banco de dados MySQL

Você pode usar o Amazon Aurora com sua instância de banco de dados MySQL para aproveitar os recursos de escalabilidade de leitura do Amazon Aurora e expandir a workload de leitura de sua instância do banco de dados MySQL. Para usar o Aurora para dimensionar a leitura da instância de banco de dados MySQL, crie um cluster de bancos de dados Aurora MySQL e faça dele uma réplica de leitura da instância do banco de dados MySQL. Em seguida, conecte-se ao cluster do Aurora MySQL para processar as consultas de leitura. O banco de dados de origem pode ser uma instância de banco de dados do RDS para MySQL ou um banco de dados MySQL executado externamente em relação ao Amazon RDS. Para obter mais informações, consulte Escalar leituras para o banco de dados MySQL com o Amazon Aurora.

Otimizando as operações de carimbo de data/hora

Quando o valor da variável do sistema `time_zone` é definido como `SYSTEM`, cada chamada de função do MySQL que requer um cálculo de fuso horário faz uma chamada à biblioteca do sistema. Ao executar instruções SQL que retornam ou alteram esses valores de `TIMESTAMP` em alta simultaneidade, você poderá sofrer maior latência, contenção de bloqueios e uso da CPU. Para obter mais informações, consulte time_zone na documentação do MySQL.

Para evitar esse comportamento, recomendamos que você altere o valor do parâmetro `time_zone` de cluster de banco de dados para `UTC`. Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Embora o parâmetro `time_zone` seja dinâmico (não requer a reinicialização do servidor de banco de dados), o novo valor é usado somente para novas conexões. Para garantir que todas as conexões sejam atualizadas para usar o novo valor de `time_zone`, recomendamos que você recicle as conexões da aplicação depois de atualizar o parâmetro de cluster de banco de dados.

Erros de estouro de ID de índice virtual

O Aurora MySQL limita os valores de ID de índice virtual a 8 bits com o objetivo de evitar um problema causado pelo formato undo no MySQL. Se um índice exceder o limite de ID de índice virtual, seu cluster poderá ficar indisponível. Quando um índice se aproxima do limite de ID de índice virtual ou quando você tenta criar um índice acima desse limite, o RDS pode emitir o código de erro `63955` ou o código de aviso `63955`. Para solucionar um erro de limite de ID de índice virtual, recomendamos recriar o banco de dados com um despejo lógico e restauração.

Para ter mais informações sobre despejo lógico e restauração para o Amazon Aurora MySQL, consulte Migrate very large databases to Amazon Aurora MySQL using MyDumper and MyLoader. Para ter mais informações sobre como acessar logs de erro no Amazon Aurora, consulte Monitorar arquivos de log do Amazon Aurora.

Práticas recomendadas para alta disponibilidade do Aurora MySQL

Você pode aplicar as práticas recomendadas a seguir para melhorar a disponibilidade dos seus clusters do Aurora MySQL.

Tópicos

- Como utilizar o Amazon Aurora para recuperação de desastres com seus bancos de dados MySQL
- Migrar do MySQL para o Amazon Aurora MySQL com o tempo de inatividade reduzido
- Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora MySQL

Como utilizar o Amazon Aurora para recuperação de desastres com seus bancos de dados MySQL

Você pode usar o Amazon Aurora com sua instância de banco de dados MySQL para criar um backup fora do local para recuperação de desastres. Para usar o Aurora para a recuperação de desastres da instância de banco de dados MySQL, crie um cluster de bancos de dados Amazon Aurora e faça dele uma réplica de leitura da instância de banco de dados MySQL. Isso se aplica a uma instância de banco de dados do RDS for MySQL ou a um banco de dados MySQL executado externamente em relação ao Amazon RDS.

> **Important**
>
> Ao configurar a replicação entre uma instância de banco de dados MySQL e um cluster de bancos de dados Amazon Aurora MySQL, você deve monitorar a replicação para garantir que ela permaneça íntegra e repará-la, se necessário.

Para obter instruções sobre como criar um cluster de bancos de dados Amazon Aurora MySQL e torná-lo uma réplica de leitura da instância de banco de dados MySQL, siga o procedimento em Como usar o Amazon Aurora para escalar leituras para seu banco de dados MySQL.

Para ter mais informações sobre modelos de recuperação de desastres, consulte Como escolher a melhor opção de recuperação de desastres para seu cluster do Amazon Aurora MySQL.

Migrar do MySQL para o Amazon Aurora MySQL com o tempo de inatividade reduzido

Ao importar dados de um banco de dados MySQL que suporte uma aplicação on-line para um cluster de bancos de dados Amazon Aurora MySQL, você pode reduzir o tempo de interrupção do serviço durante a migração. Para fazer isso, você pode usar o procedimento descrito em Importing data to an Amazon RDS for MySQL DB instance with reduced downtime no Guia do usuário do Amazon Relational Database Service. Este procedimento pode ser especialmente útil se você está trabalhando com um banco de dados muito grande. É possível usar o procedimento para reduzir o custo da importação, minimizando a quantidade de dados que será passada pela rede para a AWS.

O procedimento lista as etapas para transferir uma cópia dos dados do banco de dados para uma instância do Amazon EC2 e importar os dados para uma nova instância do banco de dados do RDS para MySQL. Como o Amazon Aurora é compatível com o MySQL, você pode usar um cluster de bancos de dados Amazon Aurora para a instância do banco de dados MySQL do Amazon RDS de destino.

Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora MySQL

Se você estiver executando uma workload pesada ou workloads que ultrapassam os recursos alocados de sua instância de banco de dados, você pode esgotar os recursos nos quais está executando sua aplicação e o banco de dados do Aurora. Para obter métricas em sua instância de banco de dados, como utilização de CPU, uso de memória e número de conexões de banco de dados utilizadas, você pode consultar as métricas fornecidas pelo Amazon CloudWatch, Performance Insights e Enhanced Monitoring. Para obter informações sobre como monitorar a instância de banco de dados, consulte Monitorar métricas em um cluster do Amazon Aurora.

Se sua workload esgotar os recursos que você está utilizando, sua instância de banco de dados poderá ficar lenta, ser reiniciada ou até mesmo realizar o failover para outra instância de banco de dados. Para evitar isso, monitore a utilização dos recursos, examine a workload em execução em sua instância de banco de dados e faça otimizações quando necessário. Se as otimizações não melhorarem as métricas da instância e mitigarem a exaustão de recursos, considere aumentar a escala verticalmente de sua instância de banco de dados antes de atingir seus limites. Para obter mais informações sobre as classes de instância de banco de dados disponíveis e suas especificações, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Recomendações para recursos do MySQL no Aurora MySQL

Os recursos a seguir estão disponíveis no Aurora MySQL compatíveis com o MySQL. Contudo, eles apresentam problemas de performance, escalabilidade, estabilidade ou compatibilidade no ambiente do Aurora. Assim, recomendamos que você siga determinadas diretrizes no uso desses recursos. Por exemplo, recomendamos não usar determinados recursos para implantações de produção do Aurora.

Tópicos

- Utilizar a replicação de vários threads no Aurora MySQL
- Invocar funções do AWS Lambda com o uso de funções nativas do MySQL
- Como evitar transações XA com o Amazon Aurora MySQL
- Como manter chaves externas ativadas durante instruções DML
- Configurar a frequência com que o buffer de log é liberado
- Minimizar e solucionar problemas de deadlocks do Aurora MySQL

Utilizar a replicação de vários threads no Aurora MySQL

Com a replicação de logs binários de vários threads, um thread SQL faz a leitura de eventos do log de retransmissão e coloca esses eventos em fila para que os threads de operador SQL sejam aplicados. Os threads de operador SQL são gerenciados por um thread coordenador. Os eventos de log binário são aplicados em paralelo sempre que possível.

A replicação de vários threads não é compatível com o Aurora MySQL versão 3 e o Aurora MySQL versão 2.12.1 e posterior.

Em relação às versões anteriores à 3.04 do Aurora MySQL, o Aurora usa replicação de um único thread por padrão quando um cluster de bancos de dados do Aurora MySQL é usado como uma réplica de leitura para replicação de log binário.

As versões anteriores do Aurora MySQL versão 2 herdaram alguns problemas relacionados à replicação de vários threads do MySQL Community Edition. Para essas versões, recomendamos não usar a replicação de vários threads em ambientes de produção.

Se você usa a replicação de vários threads, recomendamos testá-la completamente.

Para ter mais informações sobre o uso de replicação no Amazon Aurora, consulte Replicação com o Amazon Aurora. Para ter informações sobre a replicação de vários threads no Aurora MySQL, consulte Replicação de logs binários de vários threads.

Invocar funções do AWS Lambda com o uso de funções nativas do MySQL

Recomendamos usar as funções nativas do MySQL `lambda_sync` e `lambda_async` para invocar funções do Lambda.

Se você estiver usando o procedimento `mysql.lambda_async` obsoleto, recomendamos que você faça chamadas do procedimento `mysql.lambda_async` em um procedimento armazenado. Você pode chamar este procedimento armazenado a partir de origens diferentes, como triggers ou código do cliente. Essa abordagem pode ajudar a evitar problemas de incompatibilidade de impedância e facilita para seus programadores de banco de dados invocar funções do Lambda.

Para ter mais informações sobre como invocar funções do Lambda por meio do Amazon Aurora, consulte Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL.

Como evitar transações XA com o Amazon Aurora MySQL

Recomendamos que você não use transações eXtended Architecture (XA) com o Aurora MySQL, porque elas podem gerar longos tempos de recuperação se a XA estiver no estado `PREPARED`. Se você precisa usar transações XA com o Aurora MySQL, siga estas práticas recomendadas:

- Não deixe uma transação XA aberta no estado `PREPARED`.
- Mantenha as transações XA menores o possível.

Para ter mais informações sobre o uso de transações XA com o MySQL, consulte Transações XA na documentação do MySQL.

Como manter chaves externas ativadas durante instruções DML

Recomendamos que você não execute nenhuma instrução de linguagem de definição de dados (DDL) quando a variável `foreign_key_checks` estiver definida como `0` (desativada).

Se você precisar inserir ou atualizar linhas que exigem uma violação transitória das chaves externas, siga estas etapas:

1. Defina `foreign_key_checks` como `0`.
2. Faça as alterações na sua linguagem de manipulação de dados (DML).
3. Verifique se as alterações realizadas não violam restrições de chaves externas.
4. Defina `foreign_key_checks` como `1` (ativada).

Além disso, siga estas práticas recomendadas adicionais para restrições de chaves externas:

- Verifique se suas aplicações cliente não definiram a variável `foreign_key_checks` como `0`, como parte da variável `init_connect`.
- Se a restauração a partir de um backup lógico, como `mysqldump`, apresentar falha ou não for concluída, defina `foreign_key_checks` como `1` antes de iniciar qualquer outra operação na mesma sessão. Um backup lógico define `foreign_key_checks` como `0` ao ser inicializado.

Configurar a frequência com que o buffer de log é liberado

No MySQL Community Edition, para tornar as transações duráveis, o buffer de log do InnoDB deve ser liberado para um armazenamento durável. Use o parâmetro `innodb_flush_log_at_trx_commit` para configurar a frequência com que o buffer de log é liberado para o disco.

Quando você define o parâmetro `innodb_flush_log_at_trx_commit` como o valor padrão de 1, o buffer de log é liberado em cada confirmação de transação. Essa configuração ajuda a manter o banco de dados compatível com ACID. Recomendamos que você mantenha a configuração padrão 1.

Alterar `innodb_flush_log_at_trx_commit` para um valor não padrão pode ajudar a reduzir a latência da linguagem de manipulação de dados (DML), mas sacrifica a durabilidade dos registros de log. Essa falta de durabilidade torna o banco de dados incompatível com ACID. Recomendamos que seus bancos de dados estejam em conformidade com ACID para evitar o risco de perda de dados em caso de uma reinicialização do servidor. Para ter mais informações sobre esse parâmetro, consulte innodb_flush_log_at_trx_commit na documentação do MySQL.

No Aurora MySQL, o processamento de redo log é transferido para a camada de armazenamento, portanto, nenhuma descarga dos arquivos de log ocorre na instância de banco de dados. Quando uma gravação é emitida, os redo logs são enviados da instância de banco de dados do gravador diretamente ao volume do cluster do Aurora. As únicas gravações que atravessam a rede são os registros de redo log. Nenhuma página é gravada na camada do banco de dados.

Por padrão, cada thread que confirma uma transação aguarda a confirmação do volume do cluster do Aurora. Essa confirmação indica que esse registro e todos os registros anteriores de redo log foram gravados e atingiram o quórum. A persistência dos registros de log e a obtenção do quórum tornam a transação durável, seja por meio de confirmação automática ou confirmação explícita. Para ter mais informações sobre a arquitetura de armazenamento do Aurora, consulte Amazon Aurora storage demystified (Armazenamento desmistificado do Amazon Aurora).

O Aurora MySQL não libera logs para arquivos de dados como o MySQL Community Edition. No entanto, você pode usar o parâmetro `innodb_flush_log_at_trx_commit` para reduzir as restrições de durabilidade ao gravar registros de redo log no volume do cluster do Aurora.

Para o Aurora MySQL versão 2:

- `innodb_flush_log_at_trx_commit` = 0 ou 2: o banco de dados não espera pela confirmação de que os registros de log redo sejam gravados no volume do cluster do Aurora.
- `innodb_flush_log_at_trx_commit` = 1: o banco de dados aguarda a confirmação de que os registros de log redo foram gravados no volume do cluster do Aurora.

Para o Aurora MySQL versão 3:

- `innodb_flush_log_at_trx_commit` = 0: o banco de dados não espera pela confirmação de que os registros de log redo sejam gravados no volume do cluster do Aurora.
- `innodb_flush_log_at_trx_commit` = 1 ou 2: o banco de dados aguarda a confirmação de que os registros de log redo foram gravados no volume do cluster do Aurora.

Portanto, para obter o mesmo comportamento não padrão no Aurora MySQL versão 3 que seria obtido com o valor definido como 0 ou 2 no Aurora MySQL versão 2, defina o parâmetro como 0.

Embora essas configurações possam diminuir a latência do DML para o cliente, elas também podem ocasionar perda de dados no caso de failover ou reinicialização. Portanto, recomendamos que você mantenha o parâmetro `innodb_flush_log_at_trx_commit` definido como o valor padrão de 1.

Embora a perda de dados possa ocorrer tanto no MySQL Community Edition quanto no Aurora MySQL, o comportamento difere em cada banco de dados devido às diferentes arquiteturas. Essas diferenças de arquitetura podem ocasionar vários graus de perda de dados. Para garantir que o banco de dados seja compatível com ACID, sempre defina `innodb_flush_log_at_trx_commit` como 1.

> **Note**
>
> No Aurora MySQL versão 3, antes de alterar `innodb_flush_log_at_trx_commit` para um valor diferente de 1, é necessário primeiro alterar o valor de `innodb_trx_commit_allow_data_loss` para 1. Ao fazer isso, você reconhece o risco de perda de dados.

Minimizar e solucionar problemas de deadlocks do Aurora MySQL

Usuários que executam workloads que regularmente enfrentam violações de restrições em índices secundários exclusivos ou chaves estrangeiras, ao modificar registros na mesma página de dados simultaneamente, podem ter maiores deadlocks e tempos de espera de bloqueio. Esses deadlocks e tempos limite são causados por uma correção de bug do MySQL Community Edition.

Essa correção está incluída nas versões 5.7.26 e posteriores do MySQL Community Edition e foi transferida para as versões 2.10.3 e posteriores do Aurora MySQL. A correção é necessária para reforçar a capacidade de serialização por meio da implementação de bloqueios adicionais para esses tipos de operação de linguagem de manipulação de dados (DML) nas alterações feitas nos registros em uma tabela do InnoDB. Esse problema foi descoberto como parte de uma investigação sobre problemas de deadlock introduzidos por uma correção de bug anterior do MySQL Community Edition.

A correção alterou o tratamento interno da reversão parcial de uma atualização de tupla (linha) no mecanismo de armazenamento InnoDB. Operações que geram violações de restrição em chaves estrangeiras ou índices secundários exclusivos causam reversão parcial. Isso inclui, entre outros, as instruções simultâneas `INSERT...ON DUPLICATE KEY UPDATE`, `REPLACE INTO`, e `INSERT IGNORE` (upserts).

Nesse contexto, a reversão parcial não se refere à reversão de transações em nível de aplicação, mas sim a uma reversão interna do InnoDB das alterações em um índice em cluster, quando uma violação de restrição é encontrada. Por exemplo, um valor de chave duplicado é encontrado durante uma operação de upsert.

Em uma operação normal de inserção, o InnoDB cria atomicamente entradas de índice secundárias e em cluster para cada índice. Se o InnoDB detectar um valor duplicado em um índice secundário exclusivo durante uma operação de upsert, a entrada inserida no índice em cluster deverá ser revertida (reversão parcial) e a atualização deverá ser aplicada à linha duplicada existente. Durante essa etapa interna de reversão parcial, o InnoDB deve bloquear cada registro visto como parte da

operação. A correção garante a capacidade de serialização da transação ao introduzir bloqueios adicionais após a reversão parcial.

Minimizar os deadlocks do InnoDB

Você pode adotar as abordagens a seguir para reduzir a frequência de deadlocks em sua instância de banco de dados. Mais exemplos podem ser encontrados na documentação do MySQL.

1. Para reduzir as chances de deadlock, confirme as transações imediatamente depois de fazer um conjunto relacionado de alterações. Você pode fazer isso dividindo transações grandes (atualizações de várias linhas entre confirmações) em transações menores. Se você estiver inserindo linhas em lote, tente reduzir os tamanhos das inserções em lote, especialmente ao usar as operações de upsert mencionadas anteriormente.

   Para reduzir o número de possíveis reversões parciais, você pode tentar algumas das seguintes abordagens:

   a. Substitua as operações de inserção em lote pela inserção de uma linha por vez. Isso pode reduzir o tempo em que os bloqueios são mantidos por transações que podem ter conflitos.

   b. Em vez de usar `REPLACE INTO`, reescreva a instrução SQL como uma transação com várias instruções, como a seguinte:

   ```
   BEGIN;
   DELETE conflicting rows;
   INSERT new rows;
   COMMIT;
   ```

   c. Em vez de usar `INSERT...ON DUPLICATE KEY UPDATE`, reescreva a instrução SQL como uma transação com várias instruções, como a seguinte:

   ```
   BEGIN;
   SELECT rows that conflict on secondary indexes;
   UPDATE conflicting rows;
   INSERT new rows;
   COMMIT;
   ```

2. Evite transações de longa execução, ativas ou inativas, que possam ficar bloqueadas. Isso inclui sessões interativas do cliente MySQL que podem ficar abertas por um longo período com uma transação não confirmada. Ao otimizar os tamanhos das transações ou dos lotes, o impacto pode variar dependendo de vários fatores, como simultaneidade, número de duplicatas e estrutura da tabela. Todas as alterações devem ser implementadas e testadas com base em sua workload.

3. Em algumas situações, deadlocks podem ocorrer quando duas transações tentam acessar os mesmos conjuntos de dados, em uma ou várias tabelas, em ordens diferentes. Para evitar isso, você pode modificar as transações para acessar os dados na mesma ordem, serializando o acesso. Por exemplo, crie uma fila de transações a serem concluídas. Essa abordagem pode ajudar a evitar deadlocks quando ocorrem várias transações simultaneamente.
4. Adicionar índices cuidadosamente escolhidos às suas tabelas pode melhorar a seletividade e reduzir a necessidade de acessar linhas, o que ocasiona menos bloqueios.
5. Se você encontrar um bloqueio de lacunas, poderá modificar o nível de isolamento da transação `READ COMMITTED` para a sessão ou a transação a fim de evitá-lo. Para ter mais informações sobre os níveis de isolamento do InnoDB e seus comportamentos, consulte Níveis de isolamento de transações na documentação do MySQL.

> **Note**
>
> Embora você possa tomar precauções para reduzir a possibilidade de ocorrência de deadlocks, eles são um comportamento esperado do banco de dados e ainda podem ocorrer. As aplicações devem ter a lógica necessária para lidar com deadlocks ao encontrá-los. Por exemplo, implemente a lógica de repetição e recuo na aplicação. É melhor resolver a causa raiz do problema, mas, se ocorrer um deadlock, a aplicação terá a opção de esperar e tentar novamente.

Monitorar deadlocks do InnoDB

Deadlocks podem ocorrer no MySQL quando as transações da aplicação tentam fazer um bloqueio em nível de tabela e linha de uma forma que resulta em espera circular. Um deadlock ocasional do InnoDB não é necessariamente um problema, porque o mecanismo de armazenamento do InnoDB detecta a condição imediatamente e reverte uma das transações de forma automática. Se você se deparar com deadlocks com frequência, recomendamos revisar e modificar sua aplicação para amenizar problemas de performance e evitá-los. Quando a detecção de deadlocks está ativada (o padrão), o InnoDB os detecta automaticamente nas transações e reverte uma ou mais transações para romper o deadlock. O InnoDB tenta selecionar pequenas transações para reverter, caso em que o tamanho de uma transação é determinado pelo número de linhas inseridas, atualizadas ou excluídas.

- Instrução `SHOW ENGINE`: a instrução `SHOW ENGINE INNODB STATUS \G` contém detalhes do deadlock mais recente encontrado no banco de dados desde a última reinicialização.

- Log de erros do MySQL: se você encontrar deadlocks frequentes em que a saída da instrução `SHOW ENGINE` é inadequada, você poderá ativar o parâmetro de cluster de banco de dados innodb_print_all_deadlocks.

  Quando esse parâmetro é ativado, as informações sobre todos os deadlocks nas transações do usuário do InnoDB são registradas no log de erros do Aurora MySQL.

- Métricas do Amazon CloudWatch: também recomendamos que você monitore proativamente os deadlocks usando a métrica do CloudWatch `Deadlocks`. Para ter mais informações, consulte Métricas no nível da instância do Amazon Aurora.

- Amazon CloudWatch Logs: com o CloudWatch Logs, você pode visualizar métricas, analisar dados de log e criar alarmes em tempo real. Para ter mais informações, consulte Monitorar erros no Amazon Aurora MySQL e no Amazon RDS para MySQL usando o Amazon CloudWatch e enviar notificações usando o Amazon SNS.

  Usando o CloudWatch Logs com `innodb_print_all_deadlocks` ativado, você pode configurar alarmes para receber notificação quando o número de deadlocks exceder determinado limite. Para definir um limite, recomendamos que você observe suas tendências e use um valor com base em sua workload normal.

- Performance Insights: ao usar o Performance Insights, você pode monitorar as métricas `innodb_deadlocks` e `innodb_lock_wait_timeout`. Para ter mais informações sobre essas métricas, consulte Contadores não nativos para o Aurora MySQL.

Avaliação do uso de instâncias de banco de dados para Aurora MySQL com métricas do Amazon CloudWatch

Você pode usar as métricas do CloudWatch para monitorar o throughput de sua instância de banco de dados e descobrir se sua classe de instância fornece recursos suficientes para suas aplicações. Para obter informações sobre limites de classe de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora. Encontre as especificações da sua classe de instância de banco de dados para descobrir o desempenho da rede.

Se o uso da sua instância de banco de dados estiver próximo do limite da classe da instância, a performance poderá começar a diminuir. As métricas do CloudWatch podem confirmar essa situação para que você possa planejar o aumento vertical manual da escala para uma classe de instância maior.

Combine os seguintes valores de métricas do CloudWatch para descobrir se você está se aproximando do limite da classe da instância:

- NetworkThroughput: o throughput de rede recebido e transmitido pelos clientes para cada instância no cluster de bancos de dados do Aurora. Esse valor de throughput não inclui o tráfego de rede entre instâncias no cluster de banco de dados e o volume do cluster.
- StorageNetworkThroughput: o throughput da rede recebido e enviado ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora.

Adicione NetworkThroughput a StorageNetworkThroughput para encontrar o throughput da rede recebido e enviado ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora. O limite da classe de instância para sua instância deve ser maior do que a soma dessas duas métricas combinadas.

É possível usar as seguintes métricas para analisar detalhes adicionais do tráfego de rede das aplicações clientes ao enviar e receber:

- NetworkReceiveThroughput: a quantidade de throughput de rede recebida dos clientes por cada instância de banco de dados no cluster de banco de dados do Aurora MySQL. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de banco de dados do e o volume do cluster.
- NetworkTransmitThroughput: a quantidade de throughput de rede enviada aos clientes por cada instância no cluster de bancos de dados do Aurora. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de banco de dados do e o volume do cluster.
- StorageNetworkReceiveThroughput: a quantidade de throughput de rede recebida do subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.
- StorageNetworkTransmitThroughput: a quantidade de throughput de rede enviada ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.

Adicione todas essas métricas para avaliar como o uso da rede se compara ao limite da classe da instância de banco de dados. O limite da classe de instância deve ser maior do que a soma dessas métricas combinadas.

Os limites de rede e a utilização da CPU para armazenamento estão diretamente relacionados. Quando o throughput de rede aumenta, a utilização da CPU também aumenta. O monitoramento do uso da CPU e da rede fornece informações sobre como e por que os recursos estão sendo esgotados.

Para ajudar a minimizar o uso da rede, você pode considerar o seguinte:

- Usar uma classe de instância de banco de dados maior.
- Dividir as solicitações de gravação em lotes para reduzir o total de transações.
- Direcionar a workload somente leitura para uma instância somente leitura.
- Excluir todos os índices não utilizados.

Solucionar problemas de desempenho do banco de dados do Amazon Aurora MySQL

Esse tópico se concentra em alguns problemas comuns de desempenho do banco de dados do Aurora MySQL e em como solucionar ou coletar informações para corrigir esses problemas rapidamente. Dividimos o desempenho do banco de dados em duas categorias:

- Desempenho do servidor: todo o servidor do banco de dados é executado mais lentamente.
- Desempenho da consulta: uma ou mais consultas demoram mais para serem executadas.

Opções de monitoramento da AWS

Recomendamos que você use as opções de monitoramento da AWS a seguir para ajudar na solução de problemas:

- Amazon CloudWatch: o Amazon CloudWatch monitora os recursos da AWS e as aplicações que você executa na AWS em tempo real. Você pode usar o CloudWatch para coletar e monitorar métricas, que são as variáveis mensuráveis que ajudam você a avaliar seus recursos e aplicativos. Consulte mais informações em O que é o Amazon CloudWatch?.

  É possível visualizar todas as métricas e informações de processo do sistema das instâncias de banco de dados no Console de gerenciamento da AWS. É possível configurar seu cluster de banco de dados Aurora MySQL para publicar dados de logs gerais, lentos, de auditoria e de erros em um grupo de logs no Amazon CloudWatch Logs. Isso permite que você visualize tendências, mantenha logs se um host for afetado e crie uma linha de base para o desempenho “normal” a fim de identificar facilmente anomalias ou alterações. Para ter mais informações, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.
- Monitoramento avançado: para habilitar métricas adicionais do Amazon CloudWatch para um banco de dados do Aurora MySQL, ative o monitoramento avançado. Ao criar ou modificar um cluster de banco de dados do Aurora, selecione Habilitar monitoramento avançado. Isso permite que o Aurora publique métricas de desempenho no CloudWatch. Algumas das principais métricas disponíveis incluem uso de CPU, conexões de banco de dados, uso de armazenamento e latência de consultas. Elas podem ajudar a identificar gargalos de desempenho.

  A quantidade de informações transferidas para uma instância de banco de dados é diretamente proporcional à granularidade definida para o monitoramento avançado. Um menor intervalo de monitoramento resulta em relatórios mais frequentes das métricas do sistema operacional e

aumenta seu custo de monitoramento. Para gerenciar custos, defina diferentes detalhamentos para diferentes instâncias nas Contas da AWS. A granularidade padrão na criação de uma instância é de 60 segundos. Para ter mais informações, consulte Custo do monitoramento avançado.

- Performance Insights: é possível visualizar todas as métricas de chamadas do banco de dados. Isso inclui bloqueios de banco de dados, esperas e o número de linhas processadas, todos os quais você pode usar para solucionar problemas. Ao criar ou modificar um cluster de banco de dados do Aurora, selecione Ativar o Performance Insights. Por padrão, o Performance Insights tem um período de retenção de dados de sete dias, mas pode ser personalizado para analisar tendências de desempenho de prazo maior. Para retenção de mais de sete dias, você precisa fazer upgrade para o nível pago. Consulte mais informações em Definição de preço do Performance Insights. É possível definir o período de retenção de dados para cada instância de banco de dados do Aurora separadamente. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

Motivos mais comuns para problemas de performance do banco de dados MySQL

É possível usar as etapas a seguir para solucionar problemas de desempenho no banco de dados do Aurora MySQL. Listamos essas etapas na ordem lógica da investigação, mas elas não são obrigatoriamente lineares. Uma descoberta pode ignorar certas etapas, o que permite uma série de caminhos investigativos.

1. Workload: entenda a workload do banco de dados.
2. Registro em log: revise todos os logs do banco de dados.
3. Conexões a bancos de dados: garanta que as conexões entre as aplicações e o banco de dados sejam confiáveis.
4. Performance de consultas: examine os planos de execução de consultas para ver se eles mudaram. Alterações no código podem fazer com que os planos mudem.

Solucionar problemas em workloads para bancos de dados do Aurora MySQL

A workload do banco de dados pode ser vista como leituras e gravações. Com uma compreensão da workload de banco de dados “normal”, é possível ajustar as consultas e o servidor do banco de dados para atender à demanda à medida que ela muda. Há vários motivos pelos quais o desempenho pode mudar, então a primeira etapa é entender o que mudou.

- Houve um upgrade da versão principal ou secundária?

  Um upgrade da versão principal inclui alterações no código do mecanismo, principalmente no otimizador, que podem alterar o plano de execução da consulta. Ao fazer upgrade das versões do banco de dados, especialmente das principais, é muito importante que você analise a workload do banco de dados e ajuste adequadamente. O ajuste pode envolver a otimização e a reescrita de consultas ou a adição e atualização de configurações de parâmetros, dependendo dos resultados dos testes. Entender o que está causando o impacto permitirá que você comece a se concentrar nessa área específica.

  Consulte mais informações em What is new in MySQL 8.0 e Server and status variables and options added, deprecated, or removed in MySQL 8.0 na documentação do MySQL, e Comparar o Aurora MySQL versão 2 e o Aurora MySQL versão 3.
- Houve um aumento no processamento de dados (contagem de linhas)?
- Há mais consultas sendo executadas simultaneamente?
- Há alterações no esquema ou no banco de dados?
- Houve defeitos ou correções no código?

Sumário

- Métricas de host da instância
  - Uso da CPU
  - Uso de memória
  - Throughput na rede
- Métricas de banco de dados
- Solução de problemas de uso de memória para bancos de dados Aurora MySQL
  - Exemplo 1: Alto uso contínuo de memória
  - Exemplo 2: Picos transitórios de memória

  - Exemplo 3: A memória liberável cai continuamente e não é recuperada
- Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL

Métricas de host da instância

Monitore as métricas do host da instância, como CPU, memória e atividade de rede, para ajudar a entender se houve uma alteração na workload. Há dois conceitos principais para entender as mudanças na workload:

- Utilização: o uso de um dispositivo, como CPU ou disco. Pode ser com base no tempo ou na capacidade.
  - Com base no tempo: a quantidade de tempo em que um recurso está ocupado durante determinado período de observação.
  - Com base na capacidade: a quantidade de throughput que um sistema ou componente pode fornecer, como uma porcentagem de sua capacidade.
- Saturação: o grau em que mais trabalho é exigido de um recurso do que ele pode processar. Quando o uso com base na capacidade atinge 100%, o trabalho extra não pode ser processado e deve ser colocado na fila.

Uso da CPU

É possível usar as seguintes ferramentas para identificar o uso e a saturação da CPU:

- O CloudWatch fornece a métrica `CPUUtilization`. Se isso atingir 100%, a instância estará saturada. No entanto, as métricas do CloudWatch são calculadas em média em mais de um minuto e carecem de granularidade.

  Para obter mais informações sobre métricas do CloudWatch, consulte Métricas no nível da instância do Amazon Aurora.
- O monitoramento aprimorado fornece métricas retornadas pelo comando `top` do sistema operacional. Ele mostra as médias de carga e os seguintes estados de CPU, com granularidade de um segundo:
  - `Idle (%)` = tempo ocioso
  - `IRQ (%)` = interrupções de software
  - `Nice (%)` = bom momento para processos com uma boa prioridade.
  - `Steal (%)` = tempo gasto atendendo outros locatários (relacionado à virtualização)

- `System (%)` = hora do sistema
- `User (%)` = hora do usuário
- `Wait (%)` = espera de E/S

Para ter mais informações sobre as métricas do Monitoramento Avançado, consulte Métricas do SO para Aurora.

Uso de memória

Se o sistema estiver sob pressão de memória e o consumo de recursos estiver atingindo a saturação, você deverá observar um alto grau de erros de digitalização, paginação, troca e falta de memória.

É possível usar as seguintes ferramentas para identificar o uso e a saturação da memória:

O CloudWatch fornece a métrica `FreeableMemory` que mostra quanta memória pode ser recuperada limpando alguns dos caches do sistema operacional e a memória livre atual.

Para obter mais informações sobre métricas do CloudWatch, consulte Métricas no nível da instância do Amazon Aurora.

O monitoramento avançado fornece as seguintes métricas que podem ajudar a identificar problemas de uso de memória:

- `Buffers (KB)`: a quantidade de memória usada para o buffer de solicitações de E/S antes de gravar no dispositivo de armazenamento, em kilobytes.
- `Cached (KB)`: a quantidade de memória utilizada para o armazenamento em cache da E/S baseada no sistema de arquivos.
- `Free (KB)`: a quantidade de memória não atribuída, em kilobytes.
- `Swap`: em cache, livre e total.

Por exemplo, se você perceber que a instância de banco de dados usa memória `Swap`, a quantidade total de memória para sua workload será maior do que a instância tem disponível atualmente. Recomendamos aumentar o tamanho da instância de banco de dados ou ajustar a workload para usar menos memória.

Para ter mais informações sobre as métricas do Monitoramento Avançado, consulte Métricas do SO para Aurora.

Para ter informações mais detalhadas sobre como usar o Esquema de Performance e o esquema `sys` para determinar quais conexões e componentes estão usando memória, consulte Solução de problemas de uso de memória para bancos de dados Aurora MySQL.

Throughput na rede

O CloudWatch fornece as seguintes métricas para throughput total da rede, todas com média de um minuto:

- `NetworkReceiveThroughput`: a quantidade de throughput de rede recebida dos clientes por cada instância no cluster de bancos de dados do Aurora.
- `NetworkTransmitThroughput`: a quantidade de throughput de rede enviada aos clientes por cada instância no cluster de bancos de dados do Aurora.
- `NetworkThroughput`: a quantidade de throughput de rede recebida e transmitida aos clientes por cada instância no cluster de bancos de dados do Aurora.
- `StorageNetworkReceiveThroughput`: a quantidade de throughput de rede recebida do subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.
- `StorageNetworkTransmitThroughput`: a quantidade de throughput de rede enviada ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora.
- `StorageNetworkThroughput`: a quantidade de throughput de rede recebida e enviada ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora.

Para obter mais informações sobre métricas do CloudWatch, consulte Métricas no nível da instância do Amazon Aurora.

O monitoramento avançado fornece os grafos `network` recebidos (RX) e transmitidos (TX), com granularidade de até um segundo.

Para ter mais informações sobre as métricas do Monitoramento Avançado, consulte Métricas do SO para Aurora.

Métricas de banco de dados

Examine as seguintes métricas do CloudWatch para alterações na workload:

- `BlockedTransactions`: o número médio de transações no banco de dados que são bloqueadas por segundo.
- `BufferCacheHitRatio`: a porcentagem de solicitações atendidas pelo cache de buffer.
- `CommitThroughput`: o número médio de operações de confirmação por segundo.
- `DatabaseConnections`: o número de conexões de rede cliente com a instância de banco de dados.
- `Deadlocks`: o número médio de deadlocks no banco de dados por segundo.
- `DMLThroughput`: o número médio de inserções, atualizações e exclusões por segundo.
- `ResultSetCacheHitRatio`: a porcentagem de solicitações atendidas pelo cache de consulta.
- `RollbackSegmentHistoryListLength`: os logs de ações desfeitas que registram transações confirmadas com registros marcados para exclusão.
- `RowLockTime`: o tempo total gasto adquirindo bloqueios de linha para tabelas do InnoDB.
- `SelectThroughput`: o número médio de consultas de seleção por segundo.

Para obter mais informações sobre métricas do CloudWatch, consulte Métricas no nível da instância do Amazon Aurora.

Considere as seguintes perguntas ao examinar a workload:

1. Houve mudanças recentes na classe da instância de banco de dados, por exemplo, redução do tamanho da instância de 8xlarge para 4xlarge ou alteração de db.r5 para db.r6?
2. É possível criar um clone e reproduzir o problema ou isso acontece apenas nessa instância?
3. Há esgotamento dos recursos do servidor, alto esgotamento da CPU ou da memória? Se sim, isso pode significar a necessidade de hardware adicional.
4. Uma ou mais consultas estão demorando mais?
5. As alterações são causadas por um upgrade, especialmente de uma versão principal? Se sim, compare as métricas de antes e depois do upgrade.
6. Há mudanças no número de instâncias de banco de dados do leitor?
7. Você habilitou os logs gerais, de auditoria ou binários? Para ter mais informações, consulte Registro em log de bancos de dados do Aurora MySQL.
8. Você habilitou, desabilitou ou alterou o uso da replicação de logs binários (binlogs)?

9. Há alguma transação de longa duração com um grande número de bloqueios de linha? Examine o tamanho da lista de histórico do InnoDB (HLL) para obter indicações de transações de longa duração.

   Consulte mais informações em O tamanho da lista de histórico do InnoDB aumentou significativamente e na publicação do blog Why is my SELECT query running slowly on my Amazon Aurora MySQL DB cluster?.

   a. Se um grande HLL for causado por uma transação de gravação, isso significa que os logs `UNDO` estão se acumulando (não estão sendo limpos regularmente). Em uma grande transação de gravação, esse acúmulo pode crescer rapidamente. No MySQL, `UNDO` é armazenado no espaço de tabela SYSTEM. O espaço de tabela `SYSTEM` não pode ser reduzido. O log `UNDO` pode fazer com que o espaço de tabela `SYSTEM` cresça para vários GB, ou até mesmo TB. Após a limpeza, libere o espaço alocado fazendo backup lógico (despejo) dos dados e, depois, importe o despejo para uma nova instância de banco de dados.

   b. Se um HLL grande for causado por uma transação de leitura (consulta de longa duração), isso poderá indicar que a consulta está usando uma grande quantidade de espaço temporário. Libere o espaço temporário ao reinicializar. Examine as métricas de banco de dados do Performance Insights para verificar quaisquer alterações na seção `Temp`, como `created_tmp_tables`. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

10. É possível dividir transações de longa duração em transações menores que modificam menos linhas?

11. Há alguma alteração nas transações bloqueadas ou aumento nos deadlocks? Examine as métricas de banco de dados do Performance Insights para verificar quaisquer alterações nas variáveis de status na seção `Locks`, como `innodb_row_lock_time`, `innodb_row_lock_waits` e `innodb_dead_locks`. Use intervalos de um minuto ou de cinco minutos.

12. Há um aumento nos eventos de espera? Examine os eventos de espera e os tipos de espera do Performance Insights usando intervalos de um minuto ou de cinco minutos. Analise os principais eventos de espera e veja se eles estão correlacionados às mudanças na workload ou à contenção do banco de dados. Por exemplo, `buf_pool mutex` indica a contenção do grupo de buffer. Para ter mais informações, consulte Ajustar o Aurora MySQL com eventos de espera.

Solução de problemas de uso de memória para bancos de dados Aurora MySQL

Embora o CloudWatch, o Monitoramento Avançado e o Insights de Performance ofereçam uma boa visão geral do uso da memória em nível de sistema operacional, como a quantidade de memória que o processo do banco de dados está usando, eles não permitem definir quais conexões ou componentes no mecanismo podem estar causando esse uso de memória.

Para solucionar esse problema, é possível usar o Esquema de Performance e o esquema `sys`. No Aurora MySQL versão 3, a instrumentação de memória está habilitada por padrão quando o Esquema de Performance está habilitado. No Aurora MySQL versão 2, somente a instrumentação de memória para uso de memória do Esquema de Performance está habilitada por padrão. Para ter informações sobre tabelas disponíveis no Esquema de Performance para monitorar o uso da memória e habilitar a instrumentação de memória do Esquema de Performance, consulte Memory summary tables na documentação do MySQL. Para ter mais informações sobre o uso do Esquema de Performance com o Insights de Performance, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Embora informações detalhadas estejam disponíveis no Esquema de Performance para monitorar o uso atual da memória, o esquema sys MySQL tem visualizações sobre as tabelas do Esquema de Performance que você pode usar para identificar rapidamente onde a memória está sendo usada.

No esquema `sys`, as visualizações a seguir estão disponíveis para monitorar o uso da memória por conexão, componente e consulta.

Visualizar	Descrição
memory_by_host_by_current_bytes	Fornece informações sobre o uso da memória do mecanismo por host. Pode ser útil para identificar quais servidores de aplicações ou hosts de clientes estão consumindo memória.
memory_by_thread_by_current_bytes	Fornece informações sobre o uso da memória do mecanismo por ID de thread. O ID de thread no MySQL pode ser uma conexão de cliente ou um thread em segundo plano. É possível associar IDs de thread a IDs de conexão MySQL usando a visualização sys.processlist ou a tabela performance_schema.threads.

Visualizar	Descrição
memory_by_user_by_current_bytes	Fornece informações sobre o uso da memória do mecanismo por usuário. Pode ser útil para identificar quais contas de usuário ou clientes estão consumindo memória.
memory_global_by_current_bytes	Fornece informações sobre o uso da memória do mecanismo por componente do mecanismo. Pode ser útil para identificar o uso de memória globalmente pelos buffers ou component es do mecanismo. Por exemplo, é possível que você veja o evento `memory/innodb/buf_buf_pool` para o grupo de buffer InnoDB ou o evento `memory/sql/Prepared_statement::main_mem_root` para declarações preparadas.
memory_global_total	Fornece uma visão geral do uso total de memória monitorado no mecanismo de banco de dados.

No Aurora MySQL versão 3.05 e posterior, também é possível monitorar o uso máximo de memória por resumo de declarações nas tabelas de resumo de declarações do Esquema de Performance. As tabelas de resumo de declarações contêm resumos de declarações normalizados e estatísticas agregadas sobre a execução. A coluna `MAX_TOTAL_MEMORY` pode ajudar a identificar a memória máxima usada pelo resumo da consulta desde a última redefinição das estatísticas ou desde que a instância do banco de dados foi reiniciada. Pode ser útil para identificar consultas específicas que podem estar consumindo muita memória.

> **Note**
>
> O Esquema de Performance e o esquema `sys` mostram o uso atual da memória no servidor e os limites máximos da memória consumida por conexão e componente do mecanismo. Como o Esquema de Performance é mantido na memória, as informações são redefinidas quando a instância de banco de dados é reiniciada. Para manter um histórico ao longo do

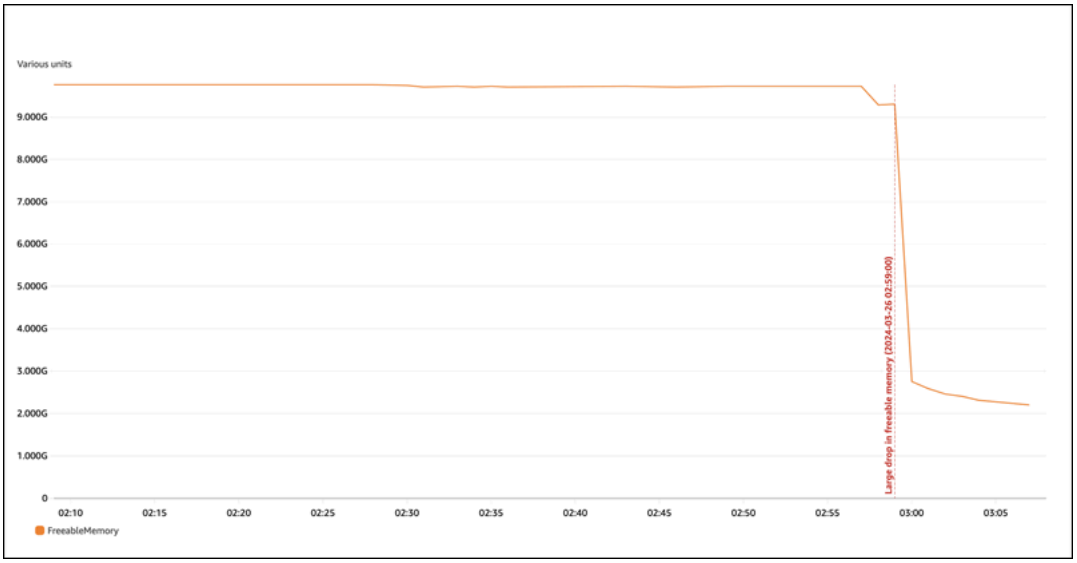
tempo, recomendamos configurar a recuperação e o armazenamento desses dados fora do Esquema de Performance.

Tópicos

- Exemplo 1: Alto uso contínuo de memória
- Exemplo 2: Picos transitórios de memória
- Exemplo 3: A memória liberável cai continuamente e não é recuperada

Exemplo 1: Alto uso contínuo de memória

Examinando globalmente o `FreeableMemory` no CloudWatch, podemos ver que o uso de memória aumentou muito em 26/3/2024, 2:59, Tempo Universal Coordenado.



Ele não exibe o panorama geral. Para determinar qual componente está usando mais memória, é possível fazer login no banco de dados e verificar `sys.memory_global_by_current_bytes`. Essa tabela contém uma lista de eventos de memória monitorados pelo MySQL, além de informações sobre alocação de memória por evento. Cada evento de monitoramento de memória começa com `memory/%`, seguido de outras informações sobre o componente/recurso do mecanismo ao qual o evento está associado.

Por exemplo, `memory/performance_schema/%` corresponde a eventos de memória relacionados ao Esquema de Performance, `memory/innodb/%` corresponde ao InnoDB etc. Para ter informações sobre as convenções de nomenclatura de eventos, consulte Performance Schema instrument naming conventions na documentação do MySQL.

Na consulta a seguir, podemos encontrar o provável culpado com base em `current_alloc`, mas também podemos ver muitos eventos `memory/performance_schema/%`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/sql/Prepared_statement::main_mem_root                                  |
 512817 | 4.91 GiB      | 10.04 KiB         |     512823 | 4.91 GiB   | 10.04 KiB
 |
| memory/performance_schema/prepared_statements_instances                       |
   252 | 488.25 MiB    | 1.94 MiB          |        252 | 488.25 MiB | 1.94 MiB       |
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error            |
  1028 | 52.27 MiB     | 52.06 KiB         |       1028 | 52.27 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     4 | 47.25 MiB     | 11.81 MiB         |          4 | 47.25 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/performance_schema/memory_summary_by_thread_by_event_name              |
     4 | 31.64 MiB     | 7.91 MiB          |          4 | 31.64 MiB  | 7.91 MiB       |
| memory/innodb/memory                                                          |
 15227 | 27.44 MiB     | 1.85 KiB          |      20619 | 33.33 MiB  | 1.66 KiB       |
| memory/sql/String::value                                                      |
 74411 | 21.85 MiB     |  307 bytes        |      76867 | 25.54 MiB  |  348 bytes     |
| memory/sql/TABLE                                                              |
  8381 | 21.03 MiB     | 2.57 KiB          |       8381 | 21.03 MiB  | 2.57 KiB       |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
10 rows in set (0.02 sec)
```

Mencionamos anteriormente que o Esquema de Performance é armazenado na memória, o que significa que ele também é monitorado na instrumentação da memória `performance_schema`.

> **Note**
>
> Se você achar que o Esquema de Performance está usando muita memória e quiser limitar o uso, poderá ajustar os parâmetros do banco de dados com base nos requisitos. Para ter mais informações, consulte The Performance Schema memory-allocation model na documentação do MySQL.

Para facilitar a leitura, é possível executar novamente a mesma consulta, mas excluir os eventos do Esquema de Performance. A saída mostra o seguinte:

- O principal consumidor de memória é `memory/sql/Prepared_statement::main_mem_root`.
- A coluna `current_alloc` informa que o MySQL tem 4,91 GiB atualmente alocados para esse evento.
- A `high_alloc column` informa que 4,91 GiB é o ponto máximo de `current_alloc` desde a última vez que as estatísticas foram redefinidas ou desde a reinicialização do servidor. Isso significa que `memory/sql/Prepared_statement::main_mem_root` está no valor mais alto.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name NOT LIKE
 'memory/performance_schema/%' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root  |        512817 | 4.91 GiB      | 10.04
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
| memory/innodb/hash0hash                       |             4 | 79.07 MiB     | 19.77
 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/memory                          |         17096 | 31.68 MiB     | 1.90
 KiB          |      22498 | 37.60 MiB  | 1.71 KiB       |
| memory/sql/String::value                      |        122277 | 27.94 MiB     |  239
 bytes        |     124699 | 29.47 MiB  |  247 bytes     |
| memory/sql/TABLE                              |          9927 | 24.67 MiB     | 2.55
 KiB          |       9929 | 24.68 MiB  | 2.55 KiB       |
| memory/innodb/lock0lock                       |          8888 | 19.71 MiB     | 2.27
 KiB          |       8888 | 19.71 MiB  | 2.27 KiB       |
```

```
| memory/sql/Prepared_statement::infrastructure |        257623 | 16.24 MiB     |   66
 bytes        |     257631 | 16.24 MiB  |   66 bytes     |
| memory/mysys/KEY_CACHE                        |             3 | 16.00 MiB     | 5.33
 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/innodb/sync0arr                        |             3 | 7.03 MiB      | 2.34
 MiB          |          3 | 7.03 MiB   | 2.34 MiB       |
| memory/sql/THD::main_mem_root                 |           815 | 6.56 MiB      | 8.24
 KiB          |        849 | 7.19 MiB   | 8.67 KiB       |
+-----------------------------------------------+---------------+---------------
+------------------+-----------+-----------+----------------+
10 rows in set (0.06 sec)
```

Pelo nome do evento, podemos dizer que essa memória está sendo usada para declarações preparadas. Se você quiser ver quais conexões estão usando essa memória, poderá conferir memory_by_thread_by_current_bytes.

No exemplo a seguir, cada conexão tem aproximadamente 7 MiB alocados, com limite máximo de 6,29 MiB (`current_max_alloc`). Isso faz sentido, porque o exemplo está usando `sysbench` com 80 tabelas e 800 conexões com declarações preparadas. Se quiser reduzir o uso de memória nessa situação, poderá otimizar o uso de declarações preparadas pela aplicação para diminuir o consumo de memória.

```
mysql> SELECT * FROM sys.memory_by_thread_by_current_bytes;

+-----------+-----------------------------------------+-------------------
+------------------+------------------+------------------+-----------------+
| thread_id | user                                    | current_count_used |
 current_allocated | current_avg_alloc | current_max_alloc | total_allocated |
+-----------+-----------------------------------------+-------------------
+------------------+------------------+------------------+-----------------+
|        46 | rdsadmin@localhost                      |                405 | 8.47 MiB
          | 21.42 KiB         | 8.00 MiB          | 155.86 MiB      |
|        61 | reinvent@10.0.4.4                       |               1749 | 6.72 MiB
          | 3.93 KiB          | 6.29 MiB          | 14.24 MiB       |
|       101 | reinvent@10.0.4.4                       |               1845 | 6.71 MiB
          | 3.72 KiB          | 6.29 MiB          | 14.50 MiB       |
|        55 | reinvent@10.0.4.4                       |               1674 | 6.68 MiB
          | 4.09 KiB          | 6.29 MiB          | 14.13 MiB       |
|        57 | reinvent@10.0.4.4                       |               1416 | 6.66 MiB
          | 4.82 KiB          | 6.29 MiB          | 13.52 MiB       |
|       112 | reinvent@10.0.4.4                       |               1759 | 6.66 MiB
          | 3.88 KiB          | 6.29 MiB          | 14.17 MiB       |
```

```
|        66 | reinvent@10.0.4.4                        |              1428 | 6.64 MiB
          | 4.76 KiB          | 6.29 MiB          | 13.47 MiB       |
|        75 | reinvent@10.0.4.4                        |              1389 | 6.62 MiB
          | 4.88 KiB          | 6.29 MiB          | 13.40 MiB       |
|       116 | reinvent@10.0.4.4                        |              1333 | 6.61 MiB
          | 5.08 KiB          | 6.29 MiB          | 13.21 MiB       |
|        90 | reinvent@10.0.4.4                        |              1448 | 6.59 MiB
          | 4.66 KiB          | 6.29 MiB          | 13.58 MiB       |
|        98 | reinvent@10.0.4.4                        |              1440 | 6.57 MiB
          | 4.67 KiB          | 6.29 MiB          | 13.52 MiB       |
|        94 | reinvent@10.0.4.4                        |              1433 | 6.57 MiB
          | 4.69 KiB          | 6.29 MiB          | 13.49 MiB       |
|        62 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.48 MiB       |
|        87 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.25 MiB       |
|        99 | reinvent@10.0.4.4                        |              1346 | 6.54 MiB
          | 4.98 KiB          | 6.29 MiB          | 13.24 MiB       |
|       105 | reinvent@10.0.4.4                        |              1347 | 6.54 MiB
          | 4.97 KiB          | 6.29 MiB          | 13.34 MiB       |
|        73 | reinvent@10.0.4.4                        |              1335 | 6.54 MiB
          | 5.02 KiB          | 6.29 MiB          | 13.23 MiB       |
|        54 | reinvent@10.0.4.4                        |              1510 | 6.53 MiB
          | 4.43 KiB          | 6.29 MiB          | 13.49 MiB       |
.                                                                                .
.                                                                                .
.                                                                                .
|       812 | reinvent@10.0.4.4                        |              1259 | 6.38 MiB
          | 5.19 KiB          | 6.29 MiB          | 13.05 MiB       |
|       214 | reinvent@10.0.4.4                        |              1279 | 6.38 MiB
          | 5.10 KiB          | 6.29 MiB          | 12.90 MiB       |
|       325 | reinvent@10.0.4.4                        |              1254 | 6.38 MiB
          | 5.21 KiB          | 6.29 MiB          | 12.99 MiB       |
|       705 | reinvent@10.0.4.4                        |              1273 | 6.37 MiB
          | 5.13 KiB          | 6.29 MiB          | 13.03 MiB       |
|       530 | reinvent@10.0.4.4                        |              1268 | 6.37 MiB
          | 5.15 KiB          | 6.29 MiB          | 12.92 MiB       |
|       307 | reinvent@10.0.4.4                        |              1263 | 6.37 MiB
          | 5.17 KiB          | 6.29 MiB          | 12.87 MiB       |
|       738 | reinvent@10.0.4.4                        |              1260 | 6.37 MiB
          | 5.18 KiB          | 6.29 MiB          | 13.00 MiB       |
```

```
|       819 | reinvent@10.0.4.4                          |               1252 | 6.37 MiB
          | 5.21 KiB          | 6.29 MiB          | 13.01 MiB       |
|        31 | innodb/srv_purge_thread                    |              17810 | 3.14 MiB
          |  184 bytes        | 2.40 MiB          | 205.69 MiB      |
|        38 | rdsadmin@localhost                         |                599 | 1.76 MiB
          | 3.01 KiB          | 1.00 MiB          | 25.58 MiB       |
|         1 | sql/main                                   |               3756 | 1.32 MiB
          |  367 bytes        | 355.78 KiB        | 6.19 MiB        |
|       854 | rdsadmin@localhost                         |                 46 | 1.08 MiB
          | 23.98 KiB         | 1.00 MiB          | 5.10 MiB        |
|        30 | innodb/clone_gtid_thread                   |               1596 | 573.14
 KiB        |  367 bytes        | 254.91 KiB        | 970.69 KiB      |
|        40 | rdsadmin@localhost                         |                235 | 245.19
 KiB        | 1.04 KiB          | 128.88 KiB        | 808.64 KiB      |
|       853 | rdsadmin@localhost                         |                 96 | 94.63
 KiB        | 1009 bytes        | 29.73 KiB         | 422.45 KiB      |
|        36 | rdsadmin@localhost                         |                 33 | 36.29
 KiB        | 1.10 KiB          | 16.08 KiB         | 74.15 MiB       |
|        33 | sql/event_scheduler                        |                  3 | 16.27
 KiB        | 5.42 KiB          | 16.04 KiB         | 16.27 KiB       |
|        35 | sql/compress_gtid_table                    |                  8 | 14.20
 KiB        | 1.77 KiB          | 8.05 KiB          | 18.62 KiB       |
|        25 | innodb/fts_optimize_thread                 |                 12 | 1.86 KiB
          |  158 bytes        |  648 bytes        | 1.98 KiB        |
|        23 | innodb/srv_master_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 24.40 KiB       |
|        24 | innodb/dict_stats_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 1.35 KiB        |
|         5 | innodb/io_read_thread                      |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         6 | innodb/io_read_thread                      |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         2 | sql/aws_oscar_log_level_monitor            |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         4 | innodb/io_ibuf_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         7 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         8 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         9 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        10 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
```

```
|        11 | innodb/srv_lra_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        12 | innodb/srv_akp_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        18 | innodb/srv_lock_timeout_thread              |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |  248 bytes      |
|        19 | innodb/srv_error_monitor_thread             |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        20 | innodb/srv_monitor_thread                   |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        21 | innodb/buf_resize_thread                    |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        22 | innodb/btr_search_sys_toggle_thread         |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        32 | innodb/dict_persist_metadata_table_thread |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        34 | sql/signal_handler                          |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
+-----------+---------------------------------------------+--------------------
+-------------------+-------------------+-------------------+-----------------+
831 rows in set (2.48 sec)
```

Conforme mencionado anteriormente, o valor do ID do thread (`thd_id`) aqui pode se referir a threads em segundo plano do servidor ou a conexões do banco de dados. Se quiser associar valores de ID de thread a IDs de conexão de banco de dados, você poderá usar a tabela `performance_schema.threads` ou a visualização `sys.processlist`, em que `conn_id` é o ID de conexão.

```
mysql> SELECT thd_id,conn_id,user,db,command,state,time,last_wait FROM sys.processlist
 WHERE user='reinvent@10.0.4.4';

+--------+---------+------------------+----------+---------+----------------+------
+-----------------------------------------------+
| thd_id | conn_id | user             | db       | command | state          | time |
 last_wait                                     |
+--------+---------+------------------+----------+---------+----------------+------
+-----------------------------------------------+
|    590 |     562 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                        |
|    578 |     550 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                          |
|    579 |     551 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                        |
```

```
|    580 |     552 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    581 |     553 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    582 |     554 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                               |
|    583 |     555 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                               |
|    584 |     556 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    585 |     557 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                             |
|    586 |     558 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    587 |     559 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                             |
.
                                                .
.
                                                .
.
                                                .
|    323 |     295 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                               |
|    324 |     296 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    325 |     297 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                             |
|    326 |     298 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    438 |     410 | reinvent@10.0.4.4 | sysbench | Execute | System lock    |    0 |
 wait/lock/table/sql/handler                        |
|    280 |     252 | reinvent@10.0.4.4 | sysbench | Sleep   | starting       |    0 |
 wait/io/socket/sql/client_connection               |
|     98 |      70 | reinvent@10.0.4.4 | sysbench | Query   | freeing items  |    0 |
 NULL                                               |
+--------+---------+-------------------+----------+---------+----------------+------
+-------------------------------------------------+
804 rows in set (5.51 sec)
```

Agora interrompemos a workload `sysbench`, que encerra as conexões e libera a memória. Conferindo novamente os eventos, podemos confirmar que a memória foi liberada, mas `high_alloc` ainda informa qual é o limite máximo. A coluna `high_alloc` pode ser muito útil na

identificação de pequenos picos no uso da memória, na qual talvez você não consiga identificar imediatamente o uso de `current_alloc`, o qual mostra apenas a memória atualmente alocada.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Se quiser redefinir `high_alloc`, você poderá truncar as tabelas de resumo de memória do `performance_schema`, mas isso redefinirá toda a instrumentação da memória. Para ter mais informações, consulte Performance Schema general table characteristics na documentação do MySQL.

No exemplo a seguir, podemos ver que `high_alloc` é redefinido após o truncamento.

```
mysql> TRUNCATE `performance_schema`.`memory_summary_global_by_event_name`;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

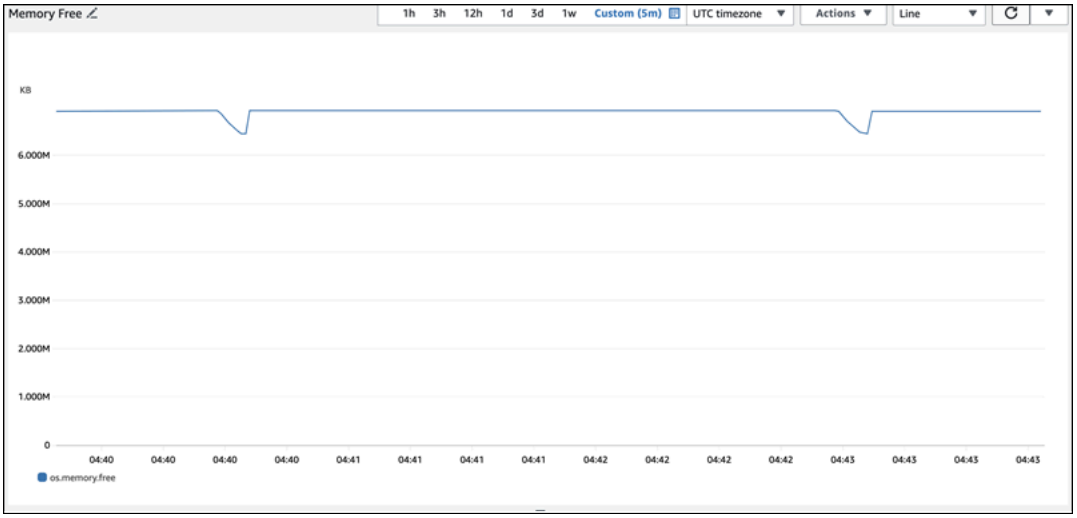
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |         17 | 253.80 KiB | 14.93 KiB      |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Exemplo 2: Picos transitórios de memória

Outra ocorrência comum são pequenos picos no uso da memória em um servidor de banco de dados. Podem ser quedas periódicas na memória liberável que são difíceis de solucionar usando `current_alloc` em `sys.memory_global_by_current_bytes`, pois a memória já foi liberada.

> **Note**
>
> Se as estatísticas do Esquema de Performance tiverem sido redefinidas ou a instância do banco de dados tiver sido reiniciada, essas informações não estarão disponíveis no `sys` nem no `performance_schema`. Para reter essas informações, recomendamos configurar a coleta de métricas externas.

O grafo a seguir da métrica `os.memory.free` no Monitoramento Avançado mostra breves picos de sete segundos no uso da memória. O Monitoramento Avançado permite monitorar em intervalos de até um segundo, o que é perfeito para detectar picos transitórios como esses.



Para ajudar a diagnosticar a causa do uso da memória aqui, podemos usar uma combinação de `high_alloc` nas visualizações resumidas de memória `sys` e tabelas de resumo de declarações do Esquema de Performance para tentar identificar sessões e conexões incorretas.

Como esperado, visto que o uso de memória atualmente não é alto, não podemos ver nenhum problema grave na visualização do esquema `sys` abaixo de `current_alloc`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;
```

```
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+------------+------------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+------------+------------
+----------------+
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/os0event                                                        |
 439372 | 60.34 MiB     |  144 bytes        |     439372 | 60.34 MiB  |  144 bytes
 |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/mysys/KEY_CACHE                                                        |
     3 | 16.00 MiB     | 5.33 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/performance_schema/events_statements_history_long                      |
     1 | 14.34 MiB     | 14.34 MiB         |          1 | 14.34 MiB  | 14.34 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error            |
   257 | 13.07 MiB     | 52.06 KiB         |        257 | 13.07 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     1 | 11.81 MiB     | 11.81 MiB         |          1 | 11.81 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.digest_text         |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.sql_text            |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+------------+------------
+----------------+
10 rows in set (0.01 sec)
```

Expandindo a visualização para ordenar por `high_alloc`, agora podemos ver que o componente `memory/temptable/physical_ram` é um candidato muito bom aqui. No momento de pico, consumiu 515,00 MiB.

Como o próprio nome sugere, `memory/temptable/physical_ram` instrumenta o uso de memória para o mecanismo de armazenamento `TEMP` no MySQL, que foi introduzido no MySQL 8.0. Para ter mais informações sobre como o MySQL usa tabelas temporárias, consulte Internal temporary table use in MySQL na documentação do MySQL.

> **Note**
>
> Estamos usando `sys.x$memory_global_by_current_bytes` neste exemplo.

```
mysql> SELECT event_name, format_bytes(current_alloc) AS "currently allocated",
 sys.format_bytes(high_alloc) AS "high-water mark"
FROM sys.x$memory_global_by_current_bytes ORDER BY high_alloc DESC LIMIT 10;

+-------------------------------------------------------------------------------
+--------------------+-----------------+
| event_name                                                                    |
 currently allocated | high-water mark |
+-------------------------------------------------------------------------------
+--------------------+-----------------+
| memory/temptable/physical_ram                                                 | 4.00
 MiB            | 515.00 MiB      |
| memory/innodb/hash0hash                                                       | 79.07
 MiB           | 79.07 MiB       |
| memory/innodb/os0event                                                        | 63.95
 MiB           | 63.95 MiB       |
| memory/performance_schema/events_statements_summary_by_digest                | 40.28
 MiB           | 40.28 MiB       |
| memory/mysys/KEY_CACHE                                                        | 16.00
 MiB           | 16.00 MiB       |
| memory/performance_schema/events_statements_history_long                      | 14.34
 MiB           | 14.34 MiB       |
| memory/performance_schema/events_errors_summary_by_thread_by_error           | 13.07
 MiB           | 13.07 MiB       |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name | 11.81
 MiB           | 11.81 MiB       |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    | 9.77
 MiB            | 9.77 MiB        |
| memory/performance_schema/events_statements_history_long.sql_text            | 9.77
 MiB            | 9.77 MiB        |
+-------------------------------------------------------------------------------
+--------------------+-----------------+
10 rows in set (0.00 sec)
```

Em Exemplo 1: Alto uso contínuo de memória, conferimos o uso atual da memória para cada conexão com o objetivo de determinar qual delas é responsável pelo uso da memória em questão.

Neste exemplo, a memória já está liberada; portanto, conferir o uso da memória nas conexões atuais não é útil.

Para nos aprofundarmos e encontrarmos as declarações, os usuários e os hosts com problemas, usamos o Esquema de Performance. O Esquema de Performance contém várias tabelas de resumo de declarações que são divididas por dimensões diferentes, como nome do evento, resumo de declarações, host, thread e usuário. Cada visualização possibilitará que você se aprofunde nos locais em que determinadas declarações estão sendo executadas e o que estão fazendo. Esta seção se concentra em `MAX_TOTAL_MEMORY`, mas é possível encontrar mais informações sobre todas as colunas disponíveis na documentação Performance Schema statement summary tables.

```
mysql> SHOW TABLES IN performance_schema LIKE 'events_statements_summary_%';

+------------------------------------------------------------+
| Tables_in_performance_schema (events_statements_summary_%) |
+------------------------------------------------------------+
| events_statements_summary_by_account_by_event_name         |
| events_statements_summary_by_digest                        |
| events_statements_summary_by_host_by_event_name            |
| events_statements_summary_by_program                       |
| events_statements_summary_by_thread_by_event_name          |
| events_statements_summary_by_user_by_event_name            |
| events_statements_summary_global_by_event_name             |
+------------------------------------------------------------+
7 rows in set (0.00 sec)
```

Primeiro, conferimos `events_statements_summary_by_digest` para ver `MAX_TOTAL_MEMORY`.

Com base nesses dados, você pode deduzir o seguinte:

- A consulta com `20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a` resumido parece ser uma boa candidata para esse uso de memória. A `MAX_TOTAL_MEMORY` é 537450710, que corresponde ao limite máximo que vimos no evento `memory/temptable/physical_ram` em `sys.x$memory_global_by_current_bytes`.
- Houve quatro execuções (`COUNT_STAR`); a primeira em 2024-03-26 04:08:34.943256 e a última em 2024-03-26 04:43:06.998310.

```
mysql> SELECT SCHEMA_NAME,DIGEST,COUNT_STAR,MAX_TOTAL_MEMORY,FIRST_SEEN,LAST_SEEN
```

```
FROM performance_schema.events_statements_summary_by_digest ORDER BY MAX_TOTAL_MEMORY
 DESC LIMIT 5;

+-------------+-------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| SCHEMA_NAME | DIGEST                                                                 |
 COUNT_STAR | MAX_TOTAL_MEMORY | FIRST_SEEN                | LAST_SEEN
  |
+-------------+-------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| sysbench    | 20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a |
     4 |        537450710 | 2024-03-26 04:08:34.943256 | 2024-03-26 04:43:06.998310 |
| NULL        | f158282ea0313fefd0a4778f6e9b92fc7d1e839af59ebd8c5eea35e12732c45d |
     4 |          3636413 | 2024-03-26 04:29:32.712348 | 2024-03-26 04:36:26.269329 |
| NULL        | 0046bc5f642c586b8a9afd6ce1ab70612dc5b1fd2408fa8677f370c1b0ca3213 |
     2 |          3459965 | 2024-03-26 04:31:37.674008 | 2024-03-26 04:32:09.410718 |
| NULL        | 8924f01bba3c55324701716c7b50071a60b9ceaf17108c71fd064c20c4ab14db |
     1 |          3290981 | 2024-03-26 04:31:49.751506 | 2024-03-26 04:31:49.751506 |
| NULL        | 90142bbcb50a744fcec03a1aa336b2169761597ea06d85c7f6ab03b5a4e1d841 |
     1 |          3131729 | 2024-03-26 04:15:09.719557 | 2024-03-26 04:15:09.719557 |
+-------------+-------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
5 rows in set (0.00 sec)
```

Agora que conhecemos o resumo de problemas, podemos ver mais detalhes, como o texto da consulta, o usuário que a executou e onde ela foi executada. Com base no texto resumido exibido, podemos ver que essa é uma expressão de tabela comum (CTE) que cria quatro tabelas temporárias e executa quatro verificações de tabela, o que é muito ineficiente.

```
mysql> SELECT
 SCHEMA_NAME,DIGEST_TEXT,QUERY_SAMPLE_TEXT,MAX_TOTAL_MEMORY,SUM_ROWS_SENT,SUM_ROWS_EXAMINED,SUM
FROM performance_schema.events_statements_summary_by_digest
WHERE DIGEST='20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a'\G;

*************************** 1. row ***************************
           SCHEMA_NAME: sysbench
           DIGEST_TEXT: WITH RECURSIVE `cte` ( `n` ) AS ( SELECT ? FROM `sbtest1` UNION
 ALL SELECT `id` + ? FROM `sbtest1` ) SELECT * FROM `cte`
     QUERY_SAMPLE_TEXT: WITH RECURSIVE cte (n) AS (   SELECT 1  from sbtest1 UNION ALL
  SELECT id + 1 FROM sbtest1) SELECT * FROM cte
```

```
      MAX_TOTAL_MEMORY: 537450710
         SUM_ROWS_SENT: 80000000
     SUM_ROWS_EXAMINED: 80000000
SUM_CREATED_TMP_TABLES: 4
     SUM_NO_INDEX_USED: 4
1 row in set (0.01 sec)
```

Para ter mais informações sobre a tabela `events_statements_summary_by_digest` e outras tabelas de resumo de declarações do Esquema de Performance, consulte Statement summary tables na documentação do MySQL.

Também é possível executar uma declaração EXPLAIN ou EXPLAIN ANALYZE para ver mais detalhes.

> **Note**
>
> `EXPLAIN ANALYZE` pode fornecer mais informações do que `EXPLAIN`, mas também executa a consulta; portanto, tenha cuidado.

```
-- EXPLAIN
mysql> EXPLAIN WITH RECURSIVE cte (n) AS (SELECT 1  FROM sbtest1 UNION ALL SELECT id +
 1 FROM sbtest1) SELECT * FROM cte;

+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL   | NULL          | NULL | NULL    |
 NULL | 19221520 |   100.00 | NULL        |
|  2 | DERIVED     | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
|  3 | UNION       | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

-- EXPLAIN format=tree
```

```
mysql> EXPLAIN format=tree WITH RECURSIVE cte (n) AS (SELECT 1 FROM sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
1 row in set (0.00 sec)

-- EXPLAIN ANALYZE
mysql> EXPLAIN ANALYZE WITH RECURSIVE cte (n) AS (SELECT 1 from sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6) (actual
 time=6666..9201 rows=20e+6 loops=1)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6) (actual
 time=6666..6666 rows=20e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0365..2006 rows=10e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0311..2494 rows=10e+6 loops=1)
1 row in set (10.53 sec)
```

Mas quem a executou? Podemos ver no Esquema de Performance que o usuário `destructive_operator` tinha 537450710 de `MAX_TOTAL_MEMORY`, que novamente corresponde aos resultados anteriores.

> **Note**
>
> O Esquema de Performance é armazenado na memória; portanto, não deve ser considerado a única fonte de auditoria. Se você precisar manter um histórico das instruções executadas e dos usuários que as executaram, recomendamos habilitar o Aurora Advanced Auditing. Se você também precisar manter informações sobre o uso da memória, recomendamos configurar o monitoramento para exportar e armazenar esses valores.

```
mysql> SELECT USER,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_user_by_event_name
ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;
```

```
+----------------------+---------------------------+------------+------------------+
| USER                 | EVENT_NAME                | COUNT_STAR | MAX_TOTAL_MEMORY |
+----------------------+---------------------------+------------+------------------+
| destructive_operator | statement/sql/select      |          4 |        537450710 |
| rdsadmin             | statement/sql/select      |       4172 |          3290981 |
| rdsadmin             | statement/sql/show_tables |          2 |          3615821 |
| rdsadmin             | statement/sql/show_fields |          2 |          3459965 |
| rdsadmin             | statement/sql/show_status |         75 |          1914976 |
+----------------------+---------------------------+------------+------------------+
5 rows in set (0.00 sec)

mysql> SELECT HOST,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_host_by_event_name
WHERE HOST != 'localhost' AND COUNT_STAR>0 ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+------------+----------------------+------------+------------------+
| HOST       | EVENT_NAME           | COUNT_STAR | MAX_TOTAL_MEMORY |
+------------+----------------------+------------+------------------+
| 10.0.8.231 | statement/sql/select |          4 |        537450710 |
+------------+----------------------+------------+------------------+
1 row in set (0.00 sec)
```

Exemplo 3: A memória liberável cai continuamente e não é recuperada

O mecanismo de banco de dados InnoDB emprega uma variedade de eventos especializados de rastreamento de memória para diferentes componentes. Esses eventos específicos permitem o rastreamento granular do uso da memória nos principais subsistemas do InnoDB, por exemplo:

- `memory/innodb/buf0buf`: dedicado ao monitoramento das alocações de memória para o grupo de buffers do InnoDB.
- `memory/innodb/ibuf0ibuf`: rastreia especificamente as alterações de memória relacionadas ao buffer de alterações do InnoDB.

Para identificar os principais consumidores de memória, podemos consultar `sys.memory_global_by_current_bytes`:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+---------------------------------------------------------------+---------------+
```

```
| event_name                                                          | current_alloc |
+---------------------------------------------------------------------+---------------+
| memory/innodb/memory                                                | 5.28 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage     | 495.00 MiB    |
| memory/performance_schema/table_shares                              | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                    | 388.95 MiB    |
| memory/innodb/std                                                   | 226.88 MiB    |
| memory/innodb/fil0fil                                               | 198.49 MiB    |
| memory/sql/binlog_io_cache                                          | 128.00 MiB    |
| memory/innodb/mem0mem                                               | 96.82 MiB     |
| memory/innodb/dict0dict                                             | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                          | 88.00 MiB     |
+---------------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

Os resultados mostram que `memory/innodb/memory` é o maior consumidor, usando 5,28 GiB de memória alocada atualmente. Esse evento serve como uma categoria para alocações de memória em vários componentes do InnoDB não associados a eventos de espera mais específicos, como `memory/innodb/buf0buf` mencionado anteriormente.

Tendo estabelecido que os componentes do InnoDB são os principais consumidores de memória, podemos nos aprofundar nas especificidades usando o seguinte comando do MySQL:

```
SHOW ENGINE INNODB STATUS \G;
```

O comando SHOW ENGINE INNODB STATUS fornece um relatório de status abrangente do mecanismo de armazenamento do InnoDB, incluindo estatísticas detalhadas de uso de memória de diferentes componentes do InnoDB. Ele pode ajudar a identificar quais estruturas ou operações específicas do InnoDB estão consumindo mais memória. Para obter mais informações, consulte InnoDB in-memory structures na documentação do MySQL.

Analisando a seção `BUFFER POOL AND MEMORY` do relatório de status do InnoDB, vemos que 5.051.647.748 bytes (4,7 GiB) são alocados ao cache de objetos do dicionário, que representa 89% da memória rastreada por `memory/innodb/memory`.

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5051647748
Buffer pool size 170512
```

```
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

O cache de objetos do dicionário é um cache global compartilhado que armazena objetos de dicionário de dados acessados anteriormente na memória para permitir a reutilização de objetos e melhorar o desempenho. A alta alocação de memória para o cache de objetos do dicionário sugere um grande número de objetos do banco de dados no cache do dicionário de dados.

Agora que sabemos que o cache do dicionário de dados é o principal consumidor, vamos inspecionar o cache do dicionário de dados em busca de tabelas abertas. Para encontrar o número de tabelas no cache de definição de tabela, consulte a variável de status global open_table_definitions.

```
mysql> show global status like 'open_table_definitions';

+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Open_table_definitions | 20000 |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Para obter mais informações, consulte How MySQL opens and closes tables na documentação do MySQL.

É possível limitar o número de definições de tabela no cache do dicionário de dados limitando o parâmetro `table_definition_cache` no grupo de parâmetros do cluster de banco de dados ou da instância de banco de dados. Para o Aurora MySQL, esse valor serve como um limite flexível para o número de tabelas no cache de definição de tabela. O valor padrão depende da classe da instância e é definido da seguinte forma:

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Quando o número de tabelas excede o limite `table_definition_cache`, um mecanismo de menos usados recentemente (LRU) remove as tabelas do cache. No entanto, as tabelas envolvidas em relações de chave externa não são colocadas na lista de LRU, impedindo sua remoção.

Em nosso cenário atual, executamos FLUSH TABLES para limpar o cache de definição de tabela. Essa ação resulta em uma queda significativa na variável de status global Open_table_definitions, de 20.000 para 12, conforme mostrado aqui:

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 12    |
+------------------------+-------+
1 row in set (0.00 sec)
```

Apesar dessa redução, observamos que a alocação de memória para `memory/innodb/memory` permanece alta em 5,18 GiB, e a memória do dicionário alocada também permanece inalterada. Isso fica evidente nos seguintes resultados da consulta:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.18 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5001599639
Buffer pool size 170512
```

```
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Esse uso persistentemente alto de memória pode ser atribuído a tabelas envolvidas em relações de chave externa. Essas tabelas não são colocadas na lista de LRU para remoção, explicando por que a alocação de memória permanece alta mesmo depois de limpar o cache de definição de tabela.

Para resolver esse problema:

1. Revise e otimize seu esquema de banco de dados, particularmente as relações de chave externa.
2. Considere migrar para uma classe de instância de banco de dados maior que tenha mais memória para acomodar seus objetos de dicionário.

Seguindo essas etapas e entendendo os padrões de alocação de memória, é possível gerenciar melhor o uso de memória na instância de banco de dados do Aurora MySQL e evitar possíveis problemas de desempenho devido à pressão de memória.

Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL

O parâmetro de nível de instância `aurora_oom_response` do Aurora MySQL pode habilitar a instância de banco de dados a monitorar a memória de sistema consumida por várias instruções e conexões. Se o sistema estiver com pouca memória, ele poderá executar uma lista de ações para tentar liberar essa memória. Ele faz isso em uma tentativa de evitar uma reinicialização do banco de dados devido a problemas de falta de memória (OOM). O parâmetro no nível da instância usa uma string de ações separadas por vírgula que uma instância de banco de dados realiza quando sua memória está baixa. O parâmetro `aurora_oom_response` é compatível com o Aurora MySQL versões 2 e 3.

Os valores a seguir e as combinações deles podem ser usados para o parâmetro `aurora_oom_response`. Uma string vazia significa que não há ações a serem realizadas e desativa efetivamente o recurso, deixando o banco de dados propenso a reinicializações de OOM.

- `decline`: recusa novas consultas quando a instância de banco de dados tem pouca memória.
- `kill_connect`: fecha as conexões de banco de dados que estão consumindo grande quantidade de memória e encerra as transações atuais e as declarações de linguagem de definição de dados (DDL). Essa resposta não é compatível com o Aurora MySQL versão 2.

Consulte mais informações em KILL statement na documentação do MySQL.

- `kill_query`: encerra as consultas em ordem decrescente de consumo da memória até a memória da instância superar o limite mínimo. As instruções DDL não são encerradas.

  Consulte mais informações em KILL statement na documentação do MySQL.
- `print`: imprime apenas as consultas que consomem uma grande quantidade de memória.
- `tune` – ajusta os caches de tabela internos para liberar parte da memória de volta para o sistema. O Aurora MySQL diminui a memória usada para caches, como `table_open_cache` e `table_definition_cache` em condições de pouca memória. Por fim, o Aurora MySQL define seu uso de memória de volta ao normal quando o sistema deixa de ter pouca memória.

  Consulte mais informações em table_open_cache e table_definition_cache na documentação do MySQL.
- `tune_buffer_pool`: diminui o tamanho do pool de buffer para liberar alguma memória e disponibilizá-la para o servidor de banco de dados processar conexões. Essa resposta é compatível com o Aurora MySQL versão 3.06 e posterior.

  É necessário emparelhar `tune_buffer_pool` com `kill_query` ou `kill_connect` no valor do parâmetro `aurora_oom_response`. Caso contrário, o redimensionamento do pool de buffer não ocorrerá, mesmo quando você incluir `tune_buffer_pool` no valor do parâmetro.

Nas versões do Aurora MySQL anteriores à 3.06, para classes de instância de banco de dados com memória menor ou igual a 4 GiB, quando a instância está sob pressão de memória, as ações padrão incluem `print`, `tune`, `decline` e `kill_query`. Para classes de instância de banco de dados com memória maior que 4 GiB, o valor do parâmetro ficará vazio por padrão (desabilitado).

No Aurora MySQL versão 3.06 e posterior, para classes de instância de banco de dados com memória menor ou igual a 4 GiB, o Aurora MySQL também fecha as conexões que mais consomem memória (`kill_connect`). Em relação a classes de instância de banco de dados com memória maior que 4 GiB, o valor do parâmetro padrão é `print`.

No Aurora MySQL versão 3.09 e posterior, para classes de instância de banco de dados com memória superior a 4 GiB, o valor do parâmetro padrão é `print,decline,kill_connect`.

Se você costuma ter problemas de falta de memória, o uso da memória pode ser monitorado usando tabelas de resumo de memória quando `performance_schema` estiver habilitado.

Em relação a métricas do Amazon CloudWatch relacionadas a OOM, consulte Métricas no nível da instância do Amazon Aurora. Em relação a variáveis de status global relacionadas a OOM, consulte Variáveis de status globais do Aurora MySQL.

Registro em log de bancos de dados do Aurora MySQL

Os logs do Aurora MySQL fornecem informações essenciais sobre a atividade e os erros do banco de dados. Ao habilitar esses logs, é possível identificar e solucionar problemas, entender o desempenho do banco de dados e auditar a atividade do banco de dados. Recomendamos que você habilite esses logs para todas as instâncias de banco de dados do Aurora MySQL para garantir o desempenho e a disponibilidade ideais dos bancos de dados. Os tipos de logs a seguir podem ser habilitados. Cada log contém informações específicas que podem levar à descoberta de impactos no processamento do banco de dados.

- Erro: o Aurora MySQL grava no log de erros apenas na inicialização, no desligamento e quando encontra erros. Uma instância de banco de dados pode passar horas ou dias sem novas entradas gravadas no log de erros. Se você não vir nenhuma entrada recente, é porque o servidor não encontrou nenhum erro que tenha gerado uma entrada de log. O log de erros está habilitado por padrão. Para ter mais informações, consulte Logs de erro do Aurora MySQL.
- Geral: o log geral fornece informações detalhadas sobre a atividade do banco de dados, incluindo todas as instruções SQL executadas pelo mecanismo do banco de dados. Consulte mais informações sobre como habilitar o registro em log geral e definir parâmetros de registro em log em Logs gerais e de consultas lentas do Aurora MySQL e The general query log na documentação do MySQL.

  > **Note**
  >
  > Os logs gerais podem se tornar muito grandes e consumir seu armazenamento. Para ter mais informações, consulte Alternância e retenção de logs do Aurora MySQL.

- Consulta lenta: um log de consulta lenta consiste em instruções SQL que levam mais de long_query_time segundos para serem executadas e exigem que, no mínimo, min_examined_row_limit linhas sejam examinadas. É possível usar o log de consulta lenta para encontrar consultas que demoram muito para serem executadas e, portanto, são candidatas à otimização.

  O valor padrão para `long_query_time` é 10 segundos. Recomendamos que você comece com um valor alto para identificar as consultas mais lentas e, depois, vá ajustando os valores seguintes.

  Também é possível usar parâmetros relacionados, como `log_slow_admin_statements` e `log_queries_not_using_indexes`. Compare `rows_examined` com `rows_returned`. Se

`rows_examined` for muito maior que `rows_returned`, então essas consultas podem estar bloqueadas.

No Aurora MySQL versão 3, é possível habilitar `log_slow_extra` para consultar mais detalhes. Consulte mais informações em Slow query log contents na documentação do MySQL. Também é possível modificar `long_query_time` no nível da sessão para depurar a execução da consulta de forma interativa, o que é bastante útil se `log_slow_extra` estiver habilitado globalmente.

Consulte mais informações sobre como habilitar o registro em log geral e definir parâmetros de registro em log em Logs gerais e de consultas lentas do Aurora MySQL e The general query log na documentação do MySQL.

- Auditoria: o log de auditoria monitora e registra a atividade do banco de dados. O registro em log de auditoria do Aurora MySQL é chamado de Auditoria avançada. Para habilitar a Auditoria avançada, defina determinados parâmetros de cluster de banco de dados. Para ter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.
- Binário: o log binário (binlog) contém eventos que descrevem alterações no banco de dados, como operações de criação de tabelas e alterações nos dados da tabela. Ele também contém eventos para instruções que poderiam ter feito alterações (por exemplo, uma instrução DELETE que não correspondia a nenhuma linha), a menos que o registro em log baseado em linhas seja usado. O log binário também contém informações sobre quanto tempo cada instrução levou para os dados atualizados.

  A execução de um servidor com os logs binários habilitados torna o desempenho um pouco mais baixo. No entanto, os benefícios do log binário em permitir que você configure a replicação e as operações de restauração geralmente superam essa pequena diminuição no desempenho.

  > **Note**
  >
  > O Aurora MySQL não exige registro em log binário para operações de restauração.

  Consulte mais informações sobre como habilitar o registro em log binário e definir o formato do binlog em Configurar o registro em log binário do Aurora MySQL para bancos de dados single-AZ e The binary log na documentação do MySQL.

É possível publicar logs gerais, de erro, de consultas lentas, de consulta e de auditoria no Amazon CloudWatch Logs. Para ter mais informações, consulte Publicação de logs de banco de dados no Amazon CloudWatch Logs.

Outra ferramenta útil para resumir arquivos de log gerais, binários e de consultas lentas é pt-query-digest.

Solucionar problemas de conexão para bancos de dados do Aurora MySQL e

Garantir conectividade confiável entre as aplicações e a instância de banco de dados do RDS é essencial para o bom funcionamento das workloads. No entanto, problemas de conectividade podem surgir como consequência de vários fatores, como configurações de rede, problemas de autenticação ou restrições de recursos. Este guia tem como objetivo fornecer uma abordagem abrangente para solucionar problemas de conectividade com o Aurora MySQL e o .

Sumário

- Identificar problemas de conectividade de banco de dados do Aurora MySQL e
- Coletar dados sobre problemas de conectividade do Aurora MySQL e
- Monitorar conexões de banco de dados do Aurora MySQL e
  - Monitoramento adicional do Aurora MySQL
- Códigos de erro de conectividade do Aurora MySQL e
- Recomendações de ajuste de parâmetros do Aurora MySQL e
- Exemplos de solução de problemas de conexão de banco de dados do Aurora MySQL e
  - Exemplo 1: solução de problemas em tentativas de conexão malsucedidas
  - Exemplo 2: solução de problemas de desconexões anormais do cliente
  - Exemplo 3: solução de problemas em tentativas de conexão malsucedidas do IAM

Identificar problemas de conectividade de banco de dados do Aurora MySQL e

Identificar a categoria específica do problema de conectividade pode ajudar a reduzir as possíveis causas e orientar o processo de solução de problemas. Cada categoria pode exigir abordagens e técnicas diferentes para diagnóstico e solução. Os problemas de conectividade do banco de dados podem ser amplamente classificados nas categorias a seguir.

Erros e exceções de conexão

> Erros e exceções de conexão podem ocorrer por vários motivos, como strings de conexão incorretas, falhas de autenticação, interrupções na rede ou problemas no servidor do banco de dados. As causas podem incluir parâmetros de conexão configurados incorretamente, credenciais inválidas, interrupções na rede, falhas ou reinicializações do servidor de banco de dados. Grupos de segurança configurados incorretamente, configurações de nuvem privada virtual (VPC), listas

de controle de acesso (ACLs) de rede e tabelas de rotas associadas a sub-redes também podem causar problemas de conexão.

Limite de conexão atingido

Este problema ocorre quando o número de conexões simultâneas com o servidor de banco de dados excede o limite máximo permitido. Os servidores de banco de dados geralmente têm um limite máximo de conexão configurável definido pelo parâmetro max_connections nos clusters e nos grupos de parâmetros da instância. Quando um limite de conexão é aplicado, o servidor de banco de dados garante que ele tenha recursos suficientes (por exemplo, memória, CPU e identificadores de arquivos) para lidar com as conexões existentes de maneira eficiente e oferecer uma performance aceitável. As causas podem incluir vazamentos de conexão na aplicação, agrupamento ineficiente de conexões ou um aumento inesperado nas solicitações de conexão.

Tempos limite de conexão

Os tempos limite de conexão ocorrem quando a aplicação cliente não consegue estabelecer uma conexão com o servidor do banco de dados dentro de um período de tempo limite especificado. As causas comuns são problemas de rede, sobrecarga do servidor, regras de firewall e configurações de conexão configuradas incorretamente.

Tempos limite de inatividade da conexão

Conexões ociosas que permanecem inativas por um período prolongado podem ser fechadas automaticamente pelo servidor de banco de dados para conservar recursos. Esse tempo limite geralmente é configurável usando o `wait_timeout` e o `interactive_timeout parameters`, e deve ser ajustado com base nos padrões de uso da conexão da aplicação. As causas podem incluir a lógica da aplicação que deixa as conexões ociosas por longos períodos ou o gerenciamento inadequado da conexão.

Desconexão intermitente das conexões existentes

Esta classe de erros se refere a um cenário em que as conexões estabelecidas entre uma aplicação cliente e o banco de dados são encerradas ou desconectadas inesperadamente em intervalos irregulares, apesar de estarem ativas e em uso. Essas desconexões ocorrem de maneira intermitente, o que significa que acontecem em intervalos irregulares e não consistentes. As causas podem ser as seguintes:

- Problemas do servidor de banco de dados, como reinicializações ou failovers
- Processamento inadequado da conexão com a aplicação

- Problemas de balanceamento de carga e de proxy
- Instabilidade da rede
- Problemas com componentes ou middleware de terceiros envolvidos no caminho da conexão
- Tempos limite de execução da consulta
- Restrições de recursos no lado do servidor ou do cliente

Identificar a causa raiz por meio de monitoramento, registro em log e análise abrangentes é crucial, enquanto a implementação de mecanismos adequados de tratamento de erros, agrupamento de conexões e novas tentativas pode ajudar a mitigar o impacto dessas desconexões intermitentes na funcionalidade e na experiência do usuário da aplicação.

Coletar dados sobre problemas de conectividade do Aurora MySQL e

A coleta de dados abrangentes relacionados aos componentes da aplicação, do banco de dados, da rede e da infraestrutura é essencial para solucionar com eficácia os problemas de conectividade entre uma aplicação e um banco de dados do Aurora MySQL ou . Ao coletar logs, configurações e informações de diagnóstico relevantes, é possível ter insights valiosos que podem ajudar a identificar a causa básica dos problemas de conectividade e orientar você para uma solução adequada.

Logs e configurações de rede, como regras de grupos de segurança, configurações de VPC e tabelas de rotas, são essenciais para identificar possíveis gargalos ou configurações incorretas relacionadas à rede que possam estar impedindo a aplicação de estabelecer uma conexão bem-sucedida com o banco de dados. Ao analisar esses componentes de rede, é possível garantir que as portas necessárias estejam abertas, os endereços IP sejam permitidos e as configurações de roteamento estejam definidas corretamente.

Carimbos de data/hora

Registre os carimbos de data/hora exatos quando os problemas de conectividade ocorrerem. Isso pode ajudar a identificar padrões ou correlacionar os problemas com outros eventos ou atividades.

Logs de mecanismo de banco de dados

Além dos logs gerais de banco de dados, revise os logs do mecanismo de banco de dados (por exemplo, o log de erros do MySQL e o log de consultas lentas) em busca de informações ou erros relevantes que possam estar relacionados aos problemas intermitentes de conectividade. Para obter mais informações, consulte Registro em log de bancos de dados do Aurora MySQL.

Logs da aplicação cliente

Colete logs detalhados das aplicações cliente que se conectam ao banco de dados. Os logs da aplicação fornecem visibilidade das tentativas de conexão, dos erros e de qualquer informação relevante do ponto de vista da aplicação, o que pode revelar problemas relacionados às strings de conexão, às credenciais de autenticação ou ao tratamento da conexão no nível da aplicação.

Os logs do banco de dados, por outro lado, oferecem insights sobre erros do lado do banco de dados, consultas lentas ou eventos que podem estar contribuindo com os problemas de conectividade. Para obter mais informações, consulte Registro em log de bancos de dados do Aurora MySQL.

Variáveis do ambiente do cliente

Verifique se alguma variável de ambiente ou configuração no lado do cliente pode estar afetando a conexão do banco de dados, como configurações de proxy, configurações de SSL/TLS ou qualquer outra variável relevante.

Versões da biblioteca do cliente

Verifique se o cliente está usando as versões mais recentes de todos os drivers, bibliotecas ou frameworks de banco de dados usados para conectividade com o banco de dados. Versões desatualizadas podem ter problemas conhecidos ou problemas de compatibilidade.

Captura de rede do cliente

Realize uma captura de rede no lado do cliente usando uma ferramenta, como o Wireshark ou `tcpdump`, durante os momentos em que ocorrerem problemas de conectividade. Isso pode ajudar a identificar quaisquer problemas ou anomalias relacionados à rede no lado do cliente.

Topologia de rede do cliente

Entenda a topologia de rede do cliente, inclusive quaisquer firewalls, balanceadores de carga ou outros componentes, como RDS Proxy ou Proxy SQL, que estejam estabelecendo conexões com o banco de dados em vez de o cliente estabelecer conexões diretamente.

Configurações do sistema operacional do cliente

Verifique as configurações do sistema operacional do cliente que podem prejudicar a conectividade de rede, como regras de firewall, configurações do adaptador de rede e quaisquer outras configurações relevantes.

Configuração de agrupamento de conexões

Se você estiver usando um mecanismo de agrupamento de conexões na aplicação, revise as definições de configuração e monitore as métricas do grupo (por exemplo, conexões ativas, conexões ociosas e tempos limite de conexão) para garantir que o grupo esteja funcionando corretamente. Revise também as configurações do grupo, como os tamanhos mínimo e máximo e as configurações de validação da conexão, para garantir que estejam configuradas corretamente.

String de conexão

A string de conexão normalmente inclui parâmetros como nome do host ou endpoint, número da porta, nome do banco de dados e credenciais de autenticação. A análise da string de conexão pode ajudar a identificar possíveis configurações incorretas ou configurações inadequadas que possam estar causando problemas de conectividade. Por exemplo, um nome de host ou um número de porta incorreto pode impedir que o cliente acesse a instância do banco de dados. Já credenciais de autenticação inválidas podem causar falhas de autenticação e rejeições de conexão. Além disso, a string de conexão pode revelar problemas relacionados ao agrupamento de conexões, tempos limite ou outras configurações específicas da conexão que possam contribuir com problemas de conectividade. Fornecer a string de conexão completa usada pela aplicação cliente pode ajudar a identificar qualquer configuração incorreta no cliente.

Métricas de banco de dados

Monitore métricas de banco de dados, como uso de CPU, uso de memória e E/S de disco, durante os momentos em que ocorrerem problemas de conectividade. Isso pode ajudar a identificar se a instância de banco de dados está enfrentando problemas de desempenho ou de contenção de recursos.

DB engine version (Versão do mecanismo de banco de dados)

Observe a versão do mecanismo de banco de dados do Aurora MySQL e . A AWS lança regularmente atualizações que abordam problemas conhecidos, vulnerabilidades de segurança e introduzem aprimoramentos de desempenho. Portanto, é altamente recomendável que você atualize para as versões mais recentes disponíveis, pois essas atualizações geralmente incluem correções de erros e melhorias relacionadas especificamente à conectividade, ao desempenho e à estabilidade. Fornecer as informações da versão do banco de dados, junto com os outros detalhes coletados, pode ajudar a Suporte a diagnosticar e resolver problemas de conectividade com eficiência.

Métricas de rede

Colete métricas de rede, como latência, perda de pacotes e throughput, durante os momentos em que ocorrerem problemas de conectividade. Ferramentas como `ping`, `traceroute` e ferramentas de monitoramento de rede podem ajudar a coletar esses dados.

Detalhes da origem e do cliente

Determine os endereços IP dos servidores de aplicações, balanceadores de carga ou quaisquer outros componentes que estejam iniciando as conexões do banco de dados. Isso pode ser um único endereço IP ou um intervalo de endereços IP (notação CIDR). Se a origem for uma instância do Amazon EC2, também é útil analisar o tipo de instância, a zona de disponibilidade, o ID da sub-rede e os grupos de segurança associados à instância, além dos detalhes da interface de rede, como endereço IP privado e endereço IP público.

Ao analisar minuciosamente os dados coletados, é possível identificar configurações incorretas, restrições de recursos, interrupções na rede ou outros problemas subjacentes que estejam causando problemas de conectividade intermitentes ou persistentes. Essas informações permitem que você realize ações específicas, como ajustar configurações, resolver problemas de rede ou lidar com o tratamento de conexões em nível de aplicação.

Monitorar conexões de banco de dados do Aurora MySQL e

Para monitorar e solucionar problemas de conectividade, use as métricas e os recursos a seguir.

métricas do CloudWatch

- `CPUUtilization`: o alto uso da CPU na instância de banco de dados pode levar à execução lenta de consultas, o que pode resultar em tempos limite de conexão ou rejeições.
- `DatabaseConnections`: monitore o número de conexões ativas com a instância de banco de dados. Um número alto de conexões próximo ao máximo definido pode indicar possíveis problemas de conectividade ou esgotamento do grupo de conexões.
- `FreeableMemory`: pouca memória disponível pode causar problemas de desempenho e conectividade devido a restrições de recursos.
- `NetworkReceiveThroughput` e `NetworkTransmitThroughput`: picos ou quedas incomuns no throughput da rede podem indicar problemas de conectividade ou gargalos na rede.

Métricas do Performance Insights

Para solucionar problemas de conectividade no Aurora MySQL e usando o Insights de Performance, analise métricas de banco de dados, como as seguintes:

- Aborted_clients
- Aborted_connects
- Conexões
- max_connections
- Threads_connected
- Threads_created
- Threads_running

Essas métricas podem ajudar na identificação de gargalos de conexão, detecção de problemas de rede ou de autenticação, otimização do agrupamento de conexões e na garantia do gerenciamento eficiente de threads. Para obter mais informações, consulte Contadores do Performance Insights para Aurora MySQL.

Recursos do Insights de Performance

- Carga do banco de dados: visualize a carga do banco de dados ao longo do tempo e correlacione-a com problemas de conectividade ou degradação do desempenho.
- Estatísticas SQL: analise as estatísticas SQL para identificar consultas ou operações de banco de dados ineficientes que possam contribuir com problemas de conectividade.
- Principais consultas: identifique e analise as consultas que consomem mais recursos, o que pode ajudar a identificar possíveis gargalos de desempenho ou consultas de longa duração que possam estar causando problemas de conectividade.

Ao monitorar essas métricas e utilizar o Insights de Performance, você pode ter visibilidade do desempenho da instância do banco de dados, do uso de recursos e dos possíveis gargalos que possam estar causando problemas de conectividade. Por exemplo:

- Um alto nível de `DatabaseConnections` próximo ao limite máximo pode indicar esgotamento do agrupamento de conexões ou tratamento inadequado da conexão, causando problemas de conectividade.
- Alta `CPUUtilization` ou baixa `FreeableMemory` pode indicar restrições de recursos, o que pode causar lentidão na execução de consultas e tempos limite ou rejeições de conexão.

- Analisar as Principais consultas e as Estatísticas SQL pode ajudar a identificar consultas ineficientes ou que consomem muitos recursos que possam estar contribuindo com os problemas de conectividade.

Além disso, o monitoramento do CloudWatch Logs e a configuração de alarmes podem ajudar você a identificar e responder proativamente aos problemas de conectividade antes que eles aumentem.

É importante observar que, embora essas métricas e ferramentas possam fornecer insights valiosos, elas devem ser usadas com outras etapas de solução de problemas. Ao analisar também as configurações de rede, as regras do grupo de segurança e o tratamento de conexões no nível da aplicação, é possível diagnosticar e resolver problemas de conectividade de forma abrangente com instâncias de banco de dados do Aurora MySQL e .

Monitoramento adicional do Aurora MySQL

métricas do CloudWatch

- `AbortedClients`: monitora o número de conexões de cliente que foram fechadas corretamente.
- `AuroraSlowConnectionHandleCount`: monitora o número de operações lentas de controle de conexão, indicando possíveis problemas de conectividade ou gargalos de desempenho.
- `AuroraSlowHandshakeCount`: mede o número de operações lentas de handshake, o que também pode ser um indicador de problemas de conectividade.
- `ConnectionAttempts`: mede o número de tentativas de conexão feitas com a instância de banco de dados do Aurora MySQL.

Variáveis de status globais

`Aurora_external_connection_count`: mostra o número de conexões de banco de dados com a instância de banco de dados, excluindo as conexões de serviço do RDS usadas para verificações de integridade do banco de dados.

Ao monitorar essas métricas e variáveis de status global, é possível obter visibilidade dos padrões de conexão, dos erros e de possíveis gargalos que possam estar causando problemas de conectividade com sua instância do Amazon Aurora MySQL.

Por exemplo, um número alto de `AbortedClients` ou de `AuroraSlowConnectionHandleCount` pode indicar problemas de conectividade.

Além disso, a configuração de alarmes e de notificações do CloudWatch pode ajudar você a identificar e reagir proativamente aos problemas de conectividade antes que eles aumentem e prejudiquem o desempenho da aplicação.

Códigos de erro de conectividade do Aurora MySQL e

Veja a seguir alguns erros comuns de conectividade dos bancos de dados do Aurora MySQL e , além de códigos de erro e de explicações.

Código de erro 1040: muitas conexões

Este erro ocorre quando o cliente tenta estabelecer mais conexões do que o máximo permitido pelo servidor do banco de dados. As possíveis causas incluem:

- Configuração incorreta do agrupamento de conexões: se estiver usando um mecanismo de agrupamento de conexões, garanta que o tamanho máximo do grupo não esteja definido como um valor muito alto e que as conexões estejam sendo liberadas adequadamente de volta ao grupo.
- Configuração da instância do banco de dados: verifique a configuração máxima de conexões permitidas para a instância do banco de dados e ajuste-a, se necessário, definindo o parâmetro `max_connections`.
- Alta simultaneidade: se vários clientes ou aplicações estiverem se conectando ao banco de dados simultaneamente, o limite máximo permitido de conexões poderá ser atingido.

Código de erro 1045: acesso negado para o usuário '...'@'...' (usando senha: SIM/NÃO)

Este erro indica uma falha de autenticação ao tentar se conectar ao banco de dados. As possíveis causas incluem:

- Compatibilidade do plugin de autenticação: verifique se o plugin de autenticação usado pelo cliente é compatível com o mecanismo de autenticação do servidor de banco de dados.
- Nome de usuário ou senha incorretos: verifique se o nome de usuário e a senha corretos estão sendo usados na string de conexão ou no mecanismo de autenticação.
- Permissões do usuário: garanta que o usuário tenha as permissões necessárias para se conectar à instância do banco de dados a partir do host ou da rede especificada.

Código de erro 1049: banco de dados desconhecido '...'

Este erro indica que o cliente está tentando se conectar a um banco de dados que não existe no servidor. As possíveis causas incluem:

- Banco de dados não criado: verifique se o banco de dados especificado foi criado no servidor de banco de dados.
- Nome incorreto do banco de dados: verifique novamente a precisão do nome do banco de dados usado na string de conexão ou na consulta.
- Permissões de usuário: verifique se o usuário tem as permissões necessárias para acessar o banco de dados especificado.

Código de erro 1153: um pacote maior que 'max_allowed_packet' bytes foi recebido

Este erro ocorre quando o cliente tenta enviar ou receber dados que excedem o tamanho máximo de pacote permitido pelo servidor do banco de dados. As possíveis causas incluem:

- Consultas ou conjuntos de resultados grandes: se estiver realizando consultas que envolvam grandes quantidades de dados, o limite de tamanho do pacote pode ser excedido.
- Configurações de tamanho de pacote mal configuradas: verifique a configuração `max_allowed_packet` no servidor do banco de dados e ajuste-a, se necessário.
- Problemas de configuração de rede: garanta que a configuração da rede (por exemplo, tamanho da MTU) permita os tamanhos de pacotes necessários.

Código de erro 1226: o usuário '...' excedeu o recurso 'max_user_connections' (valor atual: ...)

Este erro indica que o usuário excedeu o número máximo de conexões simultâneas permitidas pelo servidor de banco de dados. As possíveis causas incluem:

- Configuração incorreta do agrupamento de conexões: se estiver usando um mecanismo de agrupamento de conexões, garanta que o tamanho máximo do grupo não esteja definido como um valor muito alto para o limite de conexão do usuário.
- Configuração da instância do banco de dados: verifique a configuração `max_user_connections` da instância do banco de dados e ajuste-a, se necessário.
- Alta simultaneidade: se vários clientes ou aplicações estiverem se conectando ao banco de dados simultaneamente usando o mesmo usuário, o limite de conexão específico do usuário poderá ser atingido.

Código de erro 2003: não é possível se conectar ao servidor MySQL em '...' (10061)

Este erro geralmente ocorre quando o cliente não consegue estabelecer uma conexão TCP/IP com o servidor do banco de dados. Ele pode ser causado por vários motivos, incluindo:

- Status da instância do banco de dados: verifique se a instância do banco de dados está no estado `available` e não esteja passando por nenhuma operação de manutenção ou backup.

- Regras de firewall: verifique se algum firewall (do sistema operacional, da rede ou do grupo de segurança) está bloqueando a conexão na porta especificada (geralmente 3306 para MySQL).
- Nome de host ou endpoint incorreto: verifique se o nome do host ou endpoint usado na cadeia de conexão está correto e corresponde à instância do banco de dados.
- Problemas de conectividade de rede: verifique se a máquina cliente pode acessar a instância do banco de dados pela rede. Verifique se há interrupções na rede, problemas de roteamento ou configurações incorretas da VPC ou de sub-rede.

Código de erro 2005: host desconhecido do servidor MySQL '...' (11001)

Este erro ocorre quando o cliente não consegue enviar o nome do host ou o endpoint do servidor de banco de dados para um endereço IP. As possíveis causas incluem:

- Problemas de resolução de DNS: verifique se a máquina cliente pode resolver o nome do host corretamente usando o DNS. Verifique as configurações de DNS, o cache DNS e tente usar o endereço IP em vez do nome do host.
- Nome de host ou endpoint incorreto: verifique novamente a precisão do nome do host ou do endpoint usado na cadeia de conexão.
- Problemas de configuração de rede: garanta que a configuração de rede do cliente (por exemplo, VPC, sub-rede e tabelas de rotas) permita a resolução de DNS e a conectividade com a instância do banco de dados.

Código de erro 2026: erro de conexão SSL

Este erro ocorre quando há um problema com a configuração de SSL/TLS ou com a validação do certificado durante a tentativa de conexão. As possíveis causas incluem:

- Expiração do certificado: verifique se o certificado SSL/TLS usado pelo servidor expirou e precisa ser renovado.
- Problemas de validação do certificado: verifique se o cliente é capaz de validar o certificado SSL/TLS do servidor corretamente e se o certificado é confiável.
- Problemas de configuração de rede: verifique se a configuração de rede permite conexões SSL/TLS e não bloqueia ou interfere no processo de handshake SSL/TLS.
- Incompatibilidade de configuração SSL/TLS: garanta que as configurações de SSL/TLS (por exemplo, conjuntos de cifras e versões de protocolo) no cliente e no servidor sejam compatíveis.

Ao entender as explicações detalhadas e as possíveis causas de cada código de erro, é possível solucionar melhor os problemas de conectividade ao trabalhar com bancos de dados do Aurora MySQL e .

Recomendações de ajuste de parâmetros do Aurora MySQL e

Conexões máximas

O ajuste desses parâmetros pode ajudar a evitar problemas de conexão causados por atingir o limite máximo permitido de conexões. Esses valores devem ser definidos adequadamente com base nos requisitos de simultaneidade e nas restrições de recursos da sua aplicação.

- `max_connections`: este parâmetro especifica o número máximo de conexões simultâneas permitidas com a instância de banco de dados.
- `max_user_connections`: este parâmetro pode ser especificado durante a criação e modificação do usuário e define o número máximo de conexões simultâneas permitidas para uma conta de usuário específica.

Tamanho do buffer de rede

Aumentar esses valores pode melhorar o desempenho da rede, especialmente para workloads que envolvem grandes transferências de dados ou conjuntos de resultados. No entanto, é preciso ter cuidado, pois tamanhos maiores de buffer podem consumir mais memória.

- `net_buffer_length`: este parâmetro define o tamanho inicial da conexão do cliente e dos buffers de resultados, equilibrando o uso da memória com o desempenho da consulta.
- `max_allowed_packet`: este parâmetro especifica o tamanho máximo de um único pacote de rede que pode ser enviado ou recebido pela instância de banco de dados.

Compressão de rede (lado do cliente)

Habilitar a compressão de rede pode reduzir o uso da largura de banda da rede, mas pode aumentar a sobrecarga da CPU nos lados do cliente e do servidor.

- `compress`: este parâmetro habilita ou desabilita a compressão de rede para a comunicação cliente/servidor.
- `compress_protocol`: este parâmetro especifica o protocolo de compressão a ser usado para comunicação de rede.

Ajuste de desempenho da rede

Ajustar esses tempos limite pode auxiliar a gerenciar conexões ociosas e evitar o esgotamento de recursos. No entanto, é preciso ter cuidado, pois valores baixos podem causar encerramentos prematuros de conexão.

- `interactive_timeout`: este parâmetro especifica o número de segundos que o servidor espera por atividade em uma conexão interativa antes de fechá-la.
- `wait_timeout`: este parâmetro determina o número de segundos que o servidor espera por atividade em uma conexão não interativa antes de fechá-la.

Configurações de tempo limite de rede

Ajustar esses tempos limite pode ajudar a resolver problemas relacionados a conexões lentas ou sem resposta. No entanto, tenha cuidado para não definir valores muito baixos, pois isso pode causar falhas de conexão prematuras.

- `net_read_timeout`: este parâmetro especifica o número de segundos de espera por mais dados de uma conexão antes de encerrar a operação de leitura.
- `net_write_timeout`: este parâmetro determina o número de segundos de espera para que um bloco seja gravado em uma conexão antes de encerrar a operação de gravação.

Exemplos de solução de problemas de conexão de banco de dados do Aurora MySQL e

Os exemplos a seguir mostram como identificar e solucionar problemas de conexão de banco de dados do Aurora MySQL e .

Exemplo 1: solução de problemas em tentativas de conexão malsucedidas

As tentativas de conexão podem falhar por vários motivos, incluindo falhas de autenticação, falhas de handshake SSL/TLS, `max_connections` com limite atingido e restrições de recursos na instância de banco de dados.

É possível rastrear o número de conexões com falha com o Insights de Performance ou usando o comando a seguir.

```
mysql> show global status like 'aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
```

```
| Aborted_connects | 7     |
+------------------+-------+
1 row in set (0.00 sec)
```

Se o número de `Aborted_connects` aumentar com o tempo, a aplicação pode estar tendo problemas intermitentes de conectividade.

Você pode usar a Auditoria avançada do Aurora para registrar as conexões e desconexões das conexões do cliente. Você pode fazer isso definindo os seguintes parâmetros no grupo de parâmetros do cluster de banco de dados:

- `server_audit_logging` = `1`
- `server_audit_events` = `CONNECT`

Veja a seguir um extrato dos logs de auditoria de uma falha de login.

```
1728498527380921,auora-mysql-node1,user_1,172.31.49.222,147189,0,FAILED_CONNECT,,,1045
1728498527380940,auora-mysql-node1,user_1,172.31.49.222,147189,0,DISCONNECT,,,0
```

Em que:

- `1728498527380921`: o carimbo de data/hora da época em que ocorreu falha no login
- `aurora-mysql-node1`: o identificador da instância do nó do cluster do Aurora MySQL no qual houve falha de conexão
- `user_1`: o nome do usuário do banco de dados para o qual houve falha de login
- `172.31.49.222`: o endereço IP privado do cliente pelo qual a conexão foi estabelecida
- `147189`: o ID de conexão do login com falha
- `FAILED_CONNECT`: indica que a conexão falhou.
- `1045`: o código de retorno. Um valor diferente de zero indica um erro. Nesse caso, `1045` corresponde ao acesso negado.

Para obter mais informações, consulte Server error codes e Client error codes na documentação do MySQL.

Você também pode examinar os logs de erro do Aurora MySQL em busca de quaisquer mensagens de erro relacionadas, por exemplo:

```
2024-10-09T19:26:59.310443Z 220 [Note] [MY-010926] [Server] Access denied for user
 'user_1'@'172.31.49.222' (using password: YES) (sql_authentication.cc:1502)
```

Exemplo 2: solução de problemas de desconexões anormais do cliente

É possível rastrear o número de desconexões anormais de clientes com o Insights de Performance ou usando o comando a seguir.

```
mysql> show global status like 'aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 9     |
+-----------------+-------+
1 row in set (0.01 sec)
```

Se o número de `Aborted_clients` aumentar ao longo do tempo, a aplicação não está fechando as conexões com o banco de dados corretamente. Se as conexões não forem fechadas adequadamente, isso pode levar a vazamentos de recursos e possíveis problemas de desempenho. Conexões abertas desnecessariamente podem consumir recursos do sistema, como memória e descritores de arquivos, o que pode eventualmente fazer com que a aplicação ou o servidor pare de responder ou reinicie.

Você pode usar a consulta a seguir para identificar contas que não estão fechando conexões corretamente. Ela recupera o nome da conta do usuário, o host do qual o usuário está se conectando, o número de conexões abertas e a porcentagem de conexões não fechadas.

```
SELECT
    ess.user,
    ess.host,
    (a.total_connections - a.current_connections) - ess.count_star AS not_closed,
    (((a.total_connections - a.current_connections) - ess.count_star) * 100) /
 (a.total_connections - a.current_connections) AS pct_not_closed
FROM
    performance_schema.events_statements_summary_by_account_by_event_name AS ess
    JOIN performance_schema.accounts AS a ON (ess.user = a.user AND ess.host = a.host)
WHERE
    ess.event_name = 'statement/com/quit'
    AND (a.total_connections - a.current_connections) > ess.count_star;

+----------+--------------+------------+----------------+
```

```
| user     | host          | not_closed | pct_not_closed |
+----------+---------------+------------+----------------+
| user1    | 172.31.49.222 |          1 |        33.3333 |
| user1    | 172.31.93.250 |       1024 |        12.1021 |
| user2    | 172.31.93.250 |         10 |        12.8551 |
+----------+---------------+------------+----------------+
3 rows in set (0.00 sec)
```

Depois de identificar as contas de usuário e os hosts dos quais as conexões não estão sendo fechadas, é possível verificar o código que não está fechando as conexões adequadamente.

Por exemplo, com o conector MySQL em Python, use o método `close()` do objeto de conexão para fechar conexões. Veja a seguir um exemplo de função que estabelece uma conexão com um banco de dados, executa uma consulta e fecha a conexão:

```
import mysql.connector

def execute_query(query):
    # Establish a connection to the database
    connection = mysql.connector.connect(
        host="your_host",
        user="your_username",
        password="your_password",
        database="your_database"
    )

    try:
        # Create a cursor object
        cursor = connection.cursor()

        # Execute the query
        cursor.execute(query)

        # Fetch and process the results
        results = cursor.fetchall()
        for row in results:
            print(row)

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
```

Neste exemplo, o método `connection.close()` é chamado no bloco `finally` para garantir que a conexão seja fechada, independentemente de ocorrer ou não uma exceção.

Exemplo 3: solução de problemas em tentativas de conexão malsucedidas do IAM

A conectividade com os usuários do AWS Identity and Access Management (IAM) pode falhar por vários motivos, como:

- Configuração incorreta da política do IAM
- Credenciais de segurança expiradas
- Problemas de conectividade de rede
- Incompatibilidade de permissões de banco de dados

Para solucionar esses erros de autenticação, habilite o recurso de exportação de logs `iam-db-auth-error` em seu Amazon Relational Database Service (RDS) ou banco de dados Aurora. Isso permitirá que você visualize mensagens de erro de autenticação detalhadas no grupo de logs do CloudWatch para seu cluster do Amazon RDS ou do Amazon Aurora.

Depois de habilitados, você pode revisar esses logs para identificar e resolver a causa específica das falhas de autenticação do IAM.

Por exemplo:

```
2025-09-22T12:02:30,806 [ERROR] Failed to authorize the connection request for
 user 'user_1' due to an internal IAM DB Auth error. (Status Code: 500, Error Code:
 InternalError)
```

e

```
2025-09-22T12:02:51,954 [ERROR] Failed to authenticate the connection request for user
 'user_2' because the provided token is malformed or otherwise invalid. (Status Code:
 400, Error Code: InvalidToken)
```

Para receber orientação sobre solução de problemas, consulte o guia de solução de problemas do Aurora para autenticação de banco de dados do IAM.

Solucionar problemas de performance em bancos de dados do Aurora MySQL

O MySQL fornece controle do otimizador de consultas por meio de variáveis do sistema que afetam a forma como os planos de consulta são avaliados, otimizações selecionáveis, dicas de otimizador e índice e o modelo de custo do otimizador. Esses pontos de dados podem ser úteis não apenas para comparar diferentes ambientes do MySQL, mas também para comparar planos de execução de consultas anteriores com planos de execução atuais e para entender a execução geral de uma consulta do MySQL a qualquer momento.

O desempenho da consulta depende de muitos fatores, incluindo o plano de execução, o esquema e o tamanho da tabela, estatísticas, recursos, índices e configuração de parâmetros. O ajuste de consultas requer a identificação de gargalos e a otimização do caminho de execução.

- Encontre o plano de execução da consulta e verifique se ela está usando índices apropriados. É possível otimizar a consulta usando `EXPLAIN` e revisando os detalhes de cada plano.
- O Aurora MySQL versão 3 (compatível com o MySQL 8.0 Community Edition) usa uma instrução `EXPLAIN ANALYZE`. A instrução `EXPLAIN ANALYZE` é uma ferramenta de criação de perfil que mostra onde o MySQL gasta tempo na consulta e por quê. Com `EXPLAIN ANALYZE`, o Aurora MySQL planeja, prepara e executa a consulta enquanto conta as linhas e mede o tempo gasto em vários pontos do plano de execução. Quando a consulta é concluída, `EXPLAIN ANALYZE` imprime o plano e suas medições em vez do resultado da consulta.
- Mantenha as estatísticas do esquema atualizadas usando a instrução `ANALYZE`. Às vezes, o otimizador de consultas pode escolher planos de execução inadequados devido a estatísticas desatualizadas. Isso pode levar a um desempenho insatisfatório de uma consulta devido às estimativas de cardinalidade imprecisas das tabelas e dos índices. A coluna `last_update` da tabela innodb_table_stats mostra a última vez que as estatísticas do esquema foram atualizadas, o que é um bom indicador de “desatualização”.
- Outros problemas podem ocorrer, como distorção da distribuição de dados, que não são levados em consideração para a cardinalidade da tabela. Consulte mais informações em Estimating ANALYZE TABLE complexity for InnoDB tables e Histogram statistics in MySQL na documentação do MySQL.

Noções básicas do tempo gasto com consultas

Veja a seguir formas de determinar o tempo gasto pelas consultas:

- Criação de perfil
- Esquema de desempenho
- Otimizador de consultas

Criação de perfil

Por padrão, a criação de perfil está desabilitada. Habilite a criação de perfil, depois execute a consulta lenta e revise seu perfil.

```
SET profiling = 1;
Run your query.
SHOW PROFILE;
```

1. Identifique o estágio em que a maior parte do tempo é gasto. De acordo com General thread states na documentação do MySQL, ler e processar linhas de uma instrução `SELECT` geralmente é o estado de execução mais longa durante a vida útil de uma determinada consulta. É possível usar a instrução `EXPLAIN` para entender como o MySQL executa essa consulta.
2. Analise o log de consultas lentas para avaliar `rows_examined` e `rows_sent` a fim de garantir que a workload seja semelhante em cada ambiente. Para obter mais informações, consulte Registro em log de bancos de dados do Aurora MySQL.
3. Execute o comando a seguir para tabelas que fazem parte da consulta identificada:

   ```
   SHOW TABLE STATUS\G;
   ```

4. Capture as seguintes saídas antes e depois de executar a consulta em cada ambiente:

   ```
   SHOW GLOBAL STATUS;
   ```

5. Execute os comandos a seguir em cada ambiente para ver se há alguma outra consulta/sessão influenciando o desempenho dessa consulta de amostra.

   ```
   SHOW FULL PROCESSLIST;

   SHOW ENGINE INNODB STATUS\G;
   ```

   Às vezes, quando os recursos no servidor estão ocupados, isso afeta todas as outras operações no servidor, incluindo consultas. Também é possível capturar informações

periodicamente quando as consultas são executadas ou configurar um trabalho `cron` para capturar informações em intervalos úteis.

Performance Schema

O Performance Schema fornece informações úteis sobre o desempenho do tempo de execução do servidor, embora tenha um impacto mínimo sobre esse desempenho. Isso é diferente de `information_schema`, que fornece informações de esquema sobre a instância de banco de dados. Para obter mais informações, consulte Visão geral do Performance Schema para o Insights de Performance no Aurora MySQL.

Rastreamento do otimizador de consulta

Para entender por que um plano de consulta específico foi escolhido para execução, é possível configurar `optimizer_trace` para acessar o otimizador de consultas do MySQL.

Execute um rastreamento do otimizador para mostrar informações abrangentes sobre todos os caminhos disponíveis para o otimizador e sua escolha.

```
SET SESSION OPTIMIZER_TRACE="enabled=on";
SET optimizer_trace_offset=-5, optimizer_trace_limit=5;

-- Run your query.
SELECT * FROM table WHERE x = 1 AND y = 'A';

-- After the query completes:
SELECT * FROM information_schema.OPTIMIZER_TRACE;
SET SESSION OPTIMIZER_TRACE="enabled=off";
```

Revisar as configurações do otimizador de consultas

O Aurora MySQL versão 3 (compatível com MySQL 8.0 Community Edition) tem muitas alterações relacionadas ao otimizador em comparação com o Aurora MySQL versão 2 (compatível com o MySQL 5.7 Community Edition). Se você tiver alguns valores personalizados para `optimizer_switch`, recomendamos que analise as diferenças nos padrões e defina os valores de `optimizer_switch` que funcionem melhor para sua workload. Também recomendamos que você teste as opções disponíveis para o Aurora MySQL versão 3 para examinar o desempenho das consultas.

> **Note**
>
> O Aurora MySQL versão 3 usa o valor padrão da comunidade de 20 para o parâmetro innodb_stats_persistent_sample_pages.

É possível usar o seguinte comando para mostrar os valores de `optimizer_switch`:

```
SELECT @@optimizer_switch\G;
```

A tabela a seguir mostra os valores padrão de `optimizer_switch` do Aurora MySQL versões 2 e 3.

Configuração	Aurora MySQL versão 2	Aurora MySQL versão 3
`batched_key_access`	desligar	desligar
`block_nested_loop`	ativado	ativado
`condition_fanout_filter`	ativado	ativado
`derived_condition_pushdown`	–	ativado
`derived_merge`	ativado	ativado
`duplicateweedout`	ativado	ativado
`engine_condition_pushdown`	ativado	ativado
`firstmatch`	ativado	ativado
`hash_join`	desligar	ativado
`hash_join_cost_based`	ativado	–
`hypergraph_optimizer`	–	desligar
`index_condition_pushdown`	ativado	ativado
`index_merge`	ativado	ativado

Configuração	Aurora MySQL versão 2	Aurora MySQL versão 3
index_merge_intersection	ativado	ativado
index_merge_sort_union	ativado	ativado
index_merge_union	ativado	ativado
loosescan	ativado	ativado
materialization	ativado	ativado
mrr	ativado	ativado
mrr_cost_based	ativado	ativado
prefer_ordering_index	ativado	ativado
semijoin	ativado	ativado
skip_scan	–	ativado
subquery_materialization_cost_based	ativado	ativado
subquery_to_derived	–	desligar
use_index_extensions	ativado	ativado
use_invisible_indexes	–	desligar

Consulte mais informações em Switchable optimizations (MySQL 5.7) e Switchable optimizations (MySQL 8.0) na documentação do MySQL.

Referência do Amazon Aurora MySQL

Esta referência inclui informações sobre parâmetros do Aurora MySQL, variáveis de status e extensões SQL gerais ou diferenças do mecanismo de banco de dados MySQL da comunidade.

Tópicos

- Parâmetros de configuração do Aurora MySQL
- Variáveis de status globais do Aurora MySQL
- Eventos de espera do Aurora MySQL
- Estados de threads do Aurora MySQL
- Níveis de isolamento do Aurora MySQL
- Dicas do Aurora MySQL
- Referência de procedimentos armazenados do Aurora MySQL
- Tabelas information_schema específicas do Aurora MySQL

Parâmetros de configuração do Aurora MySQL

Você gerencia o cluster de bancos de dados Amazon Aurora MySQL da mesma maneira que gerencia outras instâncias de banco de dados do Amazon RDS, usando parâmetros em um grupo de parâmetros de banco de dados. O Amazon Aurora é diferente de outros mecanismos de banco de dados porque você tem um cluster de banco de dados contendo várias instâncias de banco de dados. Como resultado, alguns dos parâmetros que você usa para gerenciar seu cluster de bancos de dados Aurora MySQL aplicam-se a todo o cluster. Outros parâmetros aplicam-se apenas a uma instância de banco de dados particular no cluster de banco de dados.

Para gerenciar parâmetros no nível do cluster, use grupos de parâmetros de cluster de banco de dados. Para gerenciar parâmetros no nível da instância, use grupos de parâmetros de banco de dados. Cada instância de banco de dados em um cluster de bancos de dados Aurora MySQL é compatível com o mecanismo de banco de dados MySQL. No entanto, aplique alguns parâmetros do mecanismo de banco de dados do MySQL no nível do cluster e gerencie esses parâmetros usando grupos de parâmetros do cluster de banco de dados. Você não consegue encontrar parâmetros em nível de cluster no grupo de parâmetros de banco de dados para uma instância em um cluster de bancos de dados Aurora. Uma lista de parâmetros em nível de cluster é exibida mais adiante neste tópico.

Você pode gerenciar parâmetros no nível do cluster e no nível da instância usando o Console de gerenciamento da AWS, a AWS CLI, ou a API do Amazon RDS. Use comandos separados para gerenciar parâmetros no nível do cluster e parâmetros no nível da instância. Por exemplo, você pode usar o comando da CLI modify-db-cluster-parameter-group para gerenciar parâmetros em nível do cluster em um grupo de parâmetros de cluster de banco de dados. É possível usar o comando da CLI modify-db-parameter-group para gerenciar parâmetros em nível de instância em um grupo de parâmetros de banco de dados para uma instância de banco de dados em um cluster de banco de dados.

Você pode visualizar parâmetros em nível de cluster e em nível de instância no console ou usando a CLI ou a API do RDS. Por exemplo, você pode usar o comando describe-db-cluster-parameters da AWS CLI para visualizar parâmetros em nível de cluster em um grupo de parâmetros de cluster de banco de dados. É possível usar o comando da CLI describe-db-parameters para gerenciar parâmetros em nível de instância em um grupo de parâmetros de banco de dados para uma instância de banco de dados em um cluster de banco de dados.

> **Note**
>
> Cada grupo de parâmetros padrão contém os valores padrão para todos os parâmetros no grupo. Se o parâmetro tiver “engine default” (padrão do mecanismo) para esse valor, consulte a documentação específica da versão do MySQL ou PostgreSQL quanto o valor padrão real.
> Salvo indicação em contrário, os parâmetros listados nas tabelas a seguir são válidos para as versões 2 e 3 do Aurora MySQL.

Para ter mais informações sobre os grupos de parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora. Para conhecer as regras e restrições para clusters Aurora Serverless v1, consulte Grupos de parâmetros para Aurora Serverless v1.

Tópicos

- Parâmetros no nível do cluster
- Parâmetros no nível da instância
- Parâmetros do MySQL não se aplicam ao Aurora MySQL

Parâmetros no nível do cluster

A tabela a seguir mostra todos os parâmetros que se aplicam a todo o cluster de bancos de dados Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`aurora_binlog_read_buffer_size`	Sim	Só afeta clusters que usam replicação de log binário (binlog). Para obter informações sobre replicação de log binário, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário). Retirado do Aurora MySQL versão 3.
`aurora_binlog_replication_max_yield_seconds`	Sim	Só afeta clusters que usam replicação de log binário (binlog). Para obter informações sobre replicação de log binário, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).
`aurora_binlog_replication_sec_index_parallel_workers`	Sim	Define o número total de threads paralelos disponíveis para aplicar alterações de índice secundário ao replicar transações para tabelas grandes com mais de um índice secundário. O parâmetro é definido como `0` (desabilitado) por padrão. Esse parâmetro está disponível no Aurora MySQL versão 306 e posteriores. Para ter mais informações, consulte Otimizar a replicação de logs binários para Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`aurora_binlog_use_large_read_buffer`	Sim	Só afeta clusters que usam replicação de log binário (binlog). Para obter informações sobre replicação de log binário, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário). Retirado do Aurora MySQL versão 3.
`aurora_disable_hash_join`	Sim	Defina esse parâmetro como `ON` para desativar a otimização de junções de hash no Aurora MySQL versão 2.09 ou posterior. Não há suporte para a versão 3. Para ter mais informações, consulte Consulta paralela do Amazon Aurora MySQL.
`aurora_enable_replica_log_compression`	Sim	Para ter mais informações, consulte Considerações sobre performance da replicação do Amazon Aurora MySQL. Não se aplica a clusters que fazem parte de um banco de dados global Aurora. Retirado do Aurora MySQL versão 3.
`aurora_enable_repl_bin_log_filtering`	Sim	Para ter mais informações, consulte Considerações sobre performance da replicação do Amazon Aurora MySQL. Não se aplica a clusters que fazem parte de um banco de dados global Aurora. Retirado do Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`aurora_enable_staggered_replica_restart`	Sim	Essa configuração está disponível no Aurora MySQL versão 3, mas não é usada.
`aurora_enable_zdr`	Sim	Essa configuração é ativada por padrão no Aurora MySQL 2.10 e posterior. Para ter mais informações, consulte Zero-downtime restart (ZDR – Reinício com tempo de inatividade zero) para Amazon Aurora MySQL.
`aurora_enhanced_binlog`	Sim	Defina o valor desse parâmetro como 1 para ativar o log binário avançado no Aurora MySQL versão 3.03.1 e posterior. Para ter mais informações, consulte Configurar o log binário avançado para Aurora MySQL.
`aurora_jemalloc_background_thread`	Sim	Use esse parâmetro para permitir que um thread em segundo plano execute operações de manutenção de memória. Os valores permitidos são `0` (desabilitado) e 1 (habilitado). O valor padrão é `0`. Esse parâmetro é aplicável ao Aurora MySQL versão 3.05 e posterior.

Nome do parâmetro	Permite modificação	Observações
`aurora_jemalloc_dirty_decay_ms`	Sim	Use esse parâmetro para reter a memória liberada por um período especificado (em milissegundos). A retenção da memória permite uma reutilização mais rápida. Os valores permitidos são `0`–`18446744073709551615`. O valor padrão (`0`) retorna toda a memória para o sistema operacional como memória liberável. Esse parâmetro é aplicável ao Aurora MySQL versão 3.05 e posterior.
`aurora_jemalloc_tcache_enabled`	Sim	Use esse parâmetro para atender a pequenas solicitações de memória (até 32 KiB) em um cache local de thread, ignorando as arenas de memória. Os valores permitidos são `0` (desabilitado) e 1 (habilitado). O valor padrão é 1. Esse parâmetro é aplicável ao Aurora MySQL versão 3.05 e posterior.
`aurora_load_from_s3_role`	Sim	Para ter mais informações, consulte Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3. No momento indisponível no Aurora MySQL versão 3. Usar `aws_default_s3_role`.

Nome do parâmetro	Permite modificação	Observações
`aurora_mask_password_hashes_type`	Sim	Essa configuração é ativada por padrão no Aurora MySQL 2.11 e posterior. Use essa configuração para mascarar os hashes de senha do Aurora MySQL nos logs de consultas lentas e de auditoria. Os valores permitidos são `0` e 1 (padrão). Quando definido como 1, as senhas são registradas como `<secret>`. Quando definido como `0`, as senhas são registradas como valores de hash (#).
`aurora_select_into_s3_role`	Sim	Para ter mais informações, consulte Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3. No momento indisponível no Aurora MySQL versão 3. Usar `aws_default_s3_role` .

Nome do parâmetro	Permite modificação	Observações
`authentication_kerberos_caseins_cmp`	Sim	Controla a comparação de nomes de usuário sem distinção entre maiúsculas e minúsculas para o plug-in `authentication_kerberos` . Defina-o como `true` para comparação sem distinção entre maiúsculas e minúsculas. Por padrão, é usada a comparação com distinção entre maiúsculas e minúsculas (`false`). Para ter mais informações, consulte Usar a autenticação Kerberos para Aurora MySQL. Esse parâmetro está disponível no Aurora MySQL versão 3.03 e posteriores.
`auto_increment_increment`	Sim	
`auto_increment_offset`	Sim	
`aws_default_lambda_role`	Sim	Para ter mais informações, consulte Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`aws_default_s3_role`	Sim	Usado ao invocar a instrução `LOAD DATA FROM S3`, `LOAD XML FROM S3` ou `SELECT INTO OUTFILE S3` do seu cluster de banco de dados. No Aurora MySQL versão 2, o perfil do IAM especificado nesse parâmetro será usado se não for especificado um perfil do IAM para `aurora_load_from_s3_role` ou `aurora_select_into_s3_role` para a instrução apropriada. No Aurora MySQL versão 3, o perfil do IAM especificado para esse parâmetro sempre é usado. Para ter mais informações, consulte Associar uma função do IAM a um cluster de banco de dados do Amazon Aurora MySQL.
`binlog_backup`	Sim	Defina o valor desse parâmetro como 0 para ativar o log binário avançado no Aurora MySQL versão 3.03.1 e posterior. É possível desativar esse parâmetro somente ao usar o log binário avançado. Para ter mais informações, consulte Configurar o log binário avançado para Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`binlog_checksum`	Sim	A API do RDS e AWS CLI reportam um valor de `None` se esse parâmetro não estiver definido. Nesse caso, Aurora MySQL usa o valor padrão do mecanismo, que é `CRC32`. Isso é diferente da configuração explícita de `NONE`, que desativa a soma de verificação.
`binlog-do-db`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog_format`	Sim	Para ter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).
`binlog_group_commit_sync_delay`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog_group_commit_sync_no_delay_count`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog-ignore-db`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`binlog_replication_globaldb`	Sim	Defina o valor desse parâmetro como 0 para ativar o log binário avançado no Aurora MySQL versão 3.03.1 e posterior. É possível desativar esse parâmetro somente ao usar o log binário avançado. Para ter mais informações, consulte Configurar o log binário avançado para Aurora MySQL.
`binlog_row_image`	Não	
`binlog_row_metadata`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog_row_value_options`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog_rows_query_log_events`	Sim	
`binlog_transaction_compression`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`binlog_transaction_compression_level_zstd`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
binlog_transaction_dependency_history_size	Sim	Esse parâmetro define um limite superior para o número de hashes de linha que são mantidos na memória e usados para pesquisar a transação que modificou pela última vez uma linha específica. Depois que esse número de hashes for atingido, o histórico será eliminado. Esse parâmetro se aplica ao Aurora MySQL versão 2.12 e posterior e versão 3.
binlog_transaction_dependency_tracking	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
character-set-client-handshake	Sim	
character_set_client	Sim	
character_set_connection	Sim	
character_set_database	Sim	
character_set_filesystem	Sim	
character_set_results	Sim	
character_set_server	Sim	
collation_connection	Sim	
collation_server	Sim	
completion_type	Sim	

Nome do parâmetro	Permite modificação	Observações
`default_storage_engine`	Não	Os clusters do Aurora MySQL usam o mecanismo de armazenamento InnoDB para todos os dados.
`enforce_gtid_consistency`	Às vezes	Modificável no Aurora MySQL versão 2 e posterior.
`event_scheduler`	Sim	Indica o status do programador de eventos. Modificável somente no nível do cluster no Aurora MySQL versão 3.
`gtid-mode`	Às vezes	Modificável no Aurora MySQL versão 2 e posterior.
`information_schema_stats_expiry`	Sim	O número de segundos após os quais o servidor de banco de dados MySQL busca dados do mecanismo de armazenamento e os substitui no cache. Os valores permitidos são `0`–`31536000`. Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`init_connect`	Sim	O comando a ser executado pelo servidor para cada cliente conectado . Use aspas duplas (“) nas configurações para evitar falhas de conexão, por exemplo: `SET optimizer_switch="hash_join =off"` No Aurora MySQL versão 3, esse parâmetro não se aplica aos usuários que têm o privilégio `CONNECTION_ADMIN` . Isso inclui o usuário principal do Aurora. Para ter mais informações, consulte Modelo de privilégios baseados em funções.
`innodb_adaptive_hash_index`	Sim	É possível modificar esse parâmetro no nível do cluster de banco de dados no Aurora MySQL versões 2 e 3. O Adaptive Hash Index não é compatível com instâncias de banco de dados de leitor.

Nome do parâmetro	Permite modificação	Observações
innodb_aurora_instant_alter_column_allowed	Sim	Controla se o algoritmo INSTANT pode ser usado para operações ALTER COLUMN em nível global. Os valores permitidos são os seguintes: • 0: o algoritmo INSTANT não é permitido para operações ALTER COLUMN (OFF). Reverte para outros algoritmos. • 1: o algoritmo INSTANT não é permitido para operações ALTER COLUMN (ON). Este é o valor padrão. Para obter mais informações, consulte Operações de coluna na documentação do MySQL. Esse parâmetro é aplicável ao Aurora MySQL versão 3.05 e posterior.
innodb_autoinc_lock_mode	Sim	
innodb_checksums	Não	Retirado do Aurora MySQL versão 3.
innodb_cmp_per_index_enabled	Sim	
innodb_commit_concurrency	Sim	
innodb_data_home_dir	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.

Nome do parâmetro	Permite modificação	Observações
`innodb_deadlock_detect`	Sim	Esta opção é usada para desabilitar a detecção de deadlock no Aurora MySQL versão 2.11 e posterior e versão 3. Em sistemas de alta simultaneidade, a detecção de deadlock pode causar uma desaceleração quando vários segmentos aguardam o mesmo bloqueio. Consulte a documentação do MySQL para obter mais informações sobre esse parâmetro.
`innodb_default_row_format`	Sim	Esse parâmetro define o formato de linha padrão para tabelas do InnoDB (incluindo tabelas temporárias do InnoDB criadas pelo usuário). Aplica-se ao Aurora MySQL versões 2 e 3. Os valores podem ser `DYNAMIC`, `COMPACT` ou `REDUNDANT`.
`innodb_file_per_table`	Sim	Esse parâmetro afeta como o armazenamento de tabela é organizado. Para ter mais informações, consulte Escalabilidade de armazenamento.

Nome do parâmetro	Permite modificação	Observações
`innodb_flush_log_at_trx_commit`	Sim	É altamente recomendável usar o valor padrão de 1. No Aurora MySQL versão 3, antes de definir esse parâmetro como um valor diferente de 1, é necessário definir o valor de `innodb_trx_commit_allow_data_loss` como 1. Para ter mais informações, consulte Configurar a frequência com que o buffer de log é liberado.
`innodb_ft_max_token_size`	Sim	
`innodb_ft_min_token_size`	Sim	
`innodb_ft_num_word_optimize`	Sim	
`innodb_ft_sort_pll_degree`	Sim	
`innodb_online_alter_log_max_size`	Sim	
`innodb_optimize_fulltext_only`	Sim	
`innodb_page_size`	Não	

Nome do parâmetro	Permite modificação	Observações
innodb_print_all_deadlocks	Sim	Quando ativado, registra informações sobre todos os deadlocks do InnoDB no log de erros do Aurora MySQL. Para ter mais informações, consulte Minimizar e solucionar problemas de deadlocks do Aurora MySQL.
innodb_purge_batch_size	Sim	
innodb_purge_threads	Sim	
innodb_rollback_on_timeout	Sim	
innodb_rollback_segments	Sim	
innodb_spin_wait_delay	Sim	
innodb_strict_mode	Sim	
innodb_support_xa	Sim	Retirado do Aurora MySQL versão 3.
innodb_sync_array_size	Sim	
innodb_sync_spin_loops	Sim	
innodb_stats_include_delete_marked	Sim	Quando esse parâmetro está habilitado, o InnoDB inclui registros marcados com exclusão ao calcular estatísticas persistentes do otimizador. Esse parâmetro se aplica ao Aurora MySQL versão 2.12 e posterior e versão 3.
innodb_table_locks	Sim	

Nome do parâmetro	Permite modificação	Observações
`innodb_trx_commit_allow_data_loss`	Sim	No Aurora MySQL versão 3, defina o valor desse parâmetro como 1 para que você possa alterar o valor de `innodb_flush_log_at_trx_commit` . O valor padrão de `innodb_trx_commit_allow_data_loss` é `0`. Para ter mais informações, consulte Configurar a frequência com que o buffer de log é liberado.
`innodb_undo_directory`	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
`internal_tmp_disk_storage_engine`	Sim	Controla qual mecanismo de armazenamento na memória é usado para tabelas temporárias internas. Os valores permitidos são `INNODB` e `MYISAM`. Este parâmetro é aplicável ao Aurora MySQL versão 2.

Nome do parâmetro	Permite modificação	Observações
`internal_tmp_mem_storage_engine`	Sim	Controla qual mecanismo de armazenamento na memória é usado para tabelas temporárias internas. Os valores permitidos são `MEMORY` e `TempTable` . Este parâmetro é aplicável ao Aurora MySQL versão 3.
`key_buffer_size`	Sim	Cache de chaves para tabelas MyISAM. Para ter mais informações, consulte keycache->cache_lock mutex.
`lc_time_names`	Sim	
`log_error_suppression_list`	Sim	Especifica uma lista de códigos de erros que não estão registrados no log de erros do MySQL. Isso permite que você ignore certas condições de erro não críticas para ajudar a manter seus logs de erros limpos. Para ter mais informações, consulte log_error_suppression_list na documentação do MySQL. Esse parâmetro é aplicável ao Aurora MySQL versão 3.03 e posterior.

Nome do parâmetro	Permite modificação	Observações
`low_priority_updates`	Sim	As operações `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` aguardam até que não haja nenhuma operação `SELECT` pendente. Esse parâmetro só afeta mecanismos de armazenamento que usam apenas bloqueio em nível de tabela (MyISAM, MEMORY, MERGE). Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`lower_case_table_names`	Sim (Aurora MySQL versão 2) Somente no momento da criação do cluster (Aurora MySQL versão 3)	No Aurora MySQL versão 2.10 e versões 2.x posteriores, certifique-se de reinicializar todas as instâncias de leitor depois de alterar essa configuração e de reinicializar a instância de gravador. Para obter detalhes, consulte Reinicializar um cluster do Aurora com disponibilidade de leitura. No Aurora MySQL versão 3, o valor desse parâmetro é definido permanentemente no momento da criação do cluster. Se você utilizar um valor não padrão para essa opção, configure o grupo de parâmetros personalizado do Aurora MySQL versão 3 antes do upgrade e especifique o grupo de parâmetros durante a operação de restauração do snapshot que cria o cluster da versão 3. Com um banco de dados Aurora global baseado no Aurora MySQL, você não poderá executar uma atualização no local do Aurora MySQL versão 2 para a versão 3 se o parâmetro `lower_case_table_names` estiver ativado. Para ter mais informações sobre os métodos que você pode usar, consulte Atualizações de versão principal.
`master-info-repository`	Sim	Retirado do Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`master_verify_checksum`	Sim	Aurora MySQL versão 2. Use `source_verify_checksum` no Aurora MySQL versão 3.
`max_delayed_threads`	Sim	Define o número máximo de threads para lidar com instruções `INSERT DELAYED`. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`max_error_count`	Sim	O número máximo de mensagens de erro, advertência e observação que devem ser armazenadas para exibição. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`max_execution_time`	Sim	O tempo limite para executar declarações `SELECT`, em milissegundos. O valor pode ser de `0` a `18446744073709551615`. Quando definido como `0`, não há tempo limite. Para ter mais informações, consulte max_execution_time na documentação do MySQL.
`min_examined_row_limit`	Sim	Use esse parâmetro para evitar que consultas que examinam menos do que o número especificado de linhas sejam registradas em log. Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`partial_revokes`	Não	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`preload_buffer_size`	Sim	O tamanho do buffer que é alocado ao pré-carregar índices. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`query_cache_type`	Sim	Retirado do Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`read_only`	Sim	Quando esse parâmetro é ativado, o servidor não permite atualizações, exceto aquelas executadas por threads de réplica. Para o Aurora MySQL versão 2, os valores válidos são os seguintes: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` : `ON` para réplicas de leitura. Este é o valor padrão. • `{TrueIfClusterReplica}` : `ON` para clusters de réplicas, como réplicas de leitura entre regiões, clusters secundários em um banco de dados global do Aurora e implantações azul/verde. Para o Aurora MySQL versão 3, os valores válidos são os seguintes: • `0`: `OFF`. Esse é o valor padrão. • `1` – `ON` • `{TrueIfClusterReplica}` : `ON` para clusters de réplicas, como réplicas de leitura entre regiões, clusters secundários em um banco de dados global do Aurora e implantações azul/verde.

Nome do parâmetro	Permite modificação	Observações
		No Aurora MySQL versão 3, esse parâmetro não se aplica aos usuários que têm o privilégio `CONNECTION_ADMIN` . Isso inclui o usuário principal do Aurora. Para ter mais informações, consulte Modelo de privilégios baseados em funções.
`relay-log-space-limit`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replica_parallel_type`	Sim	Esse parâmetro permite a execução paralela na réplica de todos os threads não confirmados que já estão na fase de preparação, sem violar a consistência. Aplica-se ao Aurora MySQL versão 3. No Aurora MySQL versão 3.03* e anteriores, o valor padrão é DATABASE. No Aurora MySQL versão 3.04 e posteriores, o valor padrão é LOGICAL_CLOCK.
`replica_preserve_commit_order`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replica_transaction_retries`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`replica_type_conversions`	Sim	Esse parâmetro determina as conversões de tipo usadas nas réplicas. Os valores permitidos são: `ALL_LOSSY` , `ALL_NON_LOSSY` , `ALL_SIGNED` e `ALL_UNSIGNED` . Para obter mais informações, consulte Replication with differing table definitions on source and replica na documentação do MySQL. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-do-db`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-do-table`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-ignore-db`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-ignore-table`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-wild-do-table`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`replicate-wild-ignore-table`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`require_secure_transport`	Sim	Este parâmetro é aplicável ao Aurora MySQL versões 2 e 3. Para ter mais informações, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`rpl_read_size`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`server_audit_cw_upload`	Não	Esse parâmetro foi descontinuado no Aurora MySQL. Usar `server_audit_logs_upload` . Para ter mais informações, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.
`server_audit_events`	Sim	Para ter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.
`server_audit_excl_users`	Sim	Para ter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.
`server_audit_incl_users`	Sim	Para ter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.
`server_audit_logging`	Sim	Para obter instruções sobre como fazer upload dos logs no Amazon CloudWatch Logs, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.

Nome do parâmetro	Permite modificação	Observações
`server_audit_logs_upload`	Sim	É possível publicar logs de auditoria no CloudWatch Logs habilitando a Auditoria avançada e definindo esse parâmetro como 1. O padrão do parâmetro `server_audit_logs_upload` é `0`. Para ter mais informações, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.
`server_id`	Não	
`skip-character-set-client-handshake`	Sim	
`skip_name_resolve`	Não	
`slave-skip-errors`	Sim	Só se aplica a clusters do Aurora MySQL versão 2, com compatibilidade com o MySQL 5.7.
`source_verify_checksum`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`sync_frm`	Sim	Retirado do Aurora MySQL versão 3.
`thread_cache_size`	Sim	O número de threads para armazenar em cache. Esse parâmetro se aplica ao Aurora MySQL versões 2 e 3.

Nome do parâmetro	Permite modificação	Observações
`time_zone`	Sim	Por padrão, o fuso horário de um cluster de bancos de dados do Aurora é o Tempo Universal Coordenado (UTC). Você pode definir o fuso horário de instâncias em seu cluster de banco de dados como o fuso horário local de seu aplicativo. Para ter mais informações, consulte Fuso horário local para os clusters de bancos de dados Amazon Aurora.
`tls_version`	Sim	Para ter mais informações, consulte Versões do TLS para o Aurora MySQL.

Parâmetros no nível da instância

A tabela a seguir mostra todos os parâmetros que se aplicam uma instância de banco de dados específica no cluster de bancos de dados Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`activate_all_roles_on_login`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
`allow-suspicious-udfs`	Não	
`aurora_disable_hash_join`	Sim	Defina esse parâmetro como `ON` para desativar a otimização de junções de hash no Aurora MySQL versão 2.09 ou posterior. Não há suporte para a versão 3. Para ter mais informações, consulte

Nome do parâmetro	Permite modificação	Observações
		Consulta paralela do Amazon Aurora MySQL.
`aurora_lab_mode`	Sim	Para ter mais informações, consulte Modo de laboratório do Amazon Aurora MySQL. Retirado do Aurora MySQL versão 3.
`aurora_oom_response`	Sim	Este parâmetro é compatível com o Aurora MySQL versões 2 e 3. Para ter mais informações, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL.
`aurora_parallel_query`	Sim	Defina como `ON` para ativar a consulta paralela no Aurora MySQL versão 2.09 ou posterior. O parâmetro `aurora_pq` antigo não é usado nessas versões. Para ter mais informações, consulte Consulta paralela do Amazon Aurora MySQL.
`aurora_pq`	Sim	Defina como `OFF` para desativar a consulta paralela para instâncias de banco de dados específicas nas versões anteriores à 2.09 do Aurora MySQL. Na versão 2.09 ou posterior, ative e desative a consulta paralela com `aurora_parallel_query`. Para ter mais informações, consulte Consulta paralela do Amazon Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`aurora_read_replica_read_committed`	Sim	Habilita o nível de isolamento `READ COMMITTED` para réplicas do Aurora e altera o comportamento de isolamento com o intuito de reduzir o atraso de eliminação durante consultas de longa execução. Habilite essa configuração somente se você entende as alterações de comportamento e como elas afetam os resultados da consulta. Por exemplo, essa configuração usa um isolamento menos rigoroso que o padrão do MySQL. Quando habilitada, as consultas de longa execução podem ver mais de uma cópia da mesma linha, pois o Aurora reorganiza os dados da tabela enquanto a consulta está em execução. Para ter mais informações, consulte Níveis de isolamento do Aurora MySQL.
`aurora_tmptable_enable_per_table_limit`	Sim	Determina se o parâmetro `tmp_table_size` controla o tamanho máximo das tabelas temporárias na memória criadas pelo mecanismo de armazenamento `TempTable` no Aurora MySQL versão 3.04 e posterior. Para ter mais informações, consulte Limitar o tamanho de tabelas temporárias internas na memória.

Nome do parâmetro	Permite modificação	Observações
aurora_use_vector_instructions	Sim	Quando esse parâmetro é ativado, o Aurora MySQL usa instruções otimizadas de processamento vetorial fornecidas por CPUs modernas para melhorar a performance em workloads que fazem uso intenso de E/S. Essa configuração é ativada por padrão no Aurora MySQL versão 3.05 e posterior.
autocommit	Sim	
automatic_sp_privileges	Sim	
back_log	Sim	
basedir	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
binlog_cache_size	Sim	
binlog_max_flush_queue_time	Sim	
binlog_order_commits	Sim	
binlog_stmt_cache_size	Sim	
binlog_transaction_compression	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
binlog_transaction_compression_level_zstd	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
bulk_insert_buffer_size	Sim	
concurrent_insert	Sim	
connect_timeout	Sim	
core-file	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
datadir	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
default_authentication_plugin	Não	Este parâmetro é aplicável ao Aurora MySQL versão 3.
default_time_zone	Não	
default_tmp_storage_engine	Sim	O mecanismo de armazenamento padrão para tabelas temporárias.
default_week_format	Sim	
delay_key_write	Sim	
delayed_insert_limit	Sim	
delayed_insert_timeout	Sim	
delayed_queue_size	Sim	

Nome do parâmetro	Permite modificação	Observações
div_precision_increment	Sim	
end_markers_in_json	Sim	
eq_range_index_dive_limit	Sim	
event_scheduler	Às vezes	Indica o status do programador de eventos. Modificável somente no nível do cluster no Aurora MySQL versão 3.
explicit_defaults_for_timestamp	Sim	
flush	Não	
flush_time	Sim	
ft_boolean_syntax	Não	
ft_max_word_len	Sim	
ft_min_word_len	Sim	
ft_query_expansion_limit	Sim	
ft_stopword_file	Sim	
general_log	Sim	Para obter instruções sobre como fazer upload dos logs no CloudWatch Logs, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.

Nome do parâmetro	Permite modificação	Observações
`general_log_file`	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
`group_concat_max_len`	Sim	
`host_cache_size`	Sim	
`init_connect`	Sim	O comando a ser executado pelo servidor para cada cliente conectado . Use aspas duplas (“) nas configurações para evitar falhas de conexão, por exemplo: `SET optimizer_switch="hash_join =off"` No Aurora MySQL versão 3, esse parâmetro não se aplica aos usuários que têm o privilégio `CONNECTION_ADMIN` , incluindo o usuário principal do Aurora. Para ter mais informações, consulte Modelo de privilégios baseados em funções.

Nome do parâmetro	Permite modificação	Observações
`innodb_adaptive_hash_index`	Sim	É possível modificar esse parâmetro no nível da instância de banco de dados no Aurora MySQL versão 2. É modificável somente no nível do cluster de banco de dados no Aurora MySQL versão 3. O Adaptive Hash Index não é compatível com instâncias de banco de dados de leitor.
`innodb_adaptive_max_sleep_delay`	Sim	Modificar este parâmetro não tem efeito, porque `innodb_thread_concurrency` é sempre 0 para o Aurora.
`innodb_aurora_max_partitions_for_range`	Sim	Em alguns casos em que estatísticas persistentes não estão disponíveis, você pode usar esse parâmetro para melhorar a performance das estimativas de contagem de linhas em tabelas particionadas. Você pode configurá-lo com um valor de 0 a 8.192, em que o valor determina o número de partições a serem verificadas durante a estimativa de contagem de linhas. O valor padrão é 0, que estima o uso de todas as partições, em consistência com o comportamento padrão do MySQL. Esse parâmetro está disponível para o Aurora MySQL versão 3.03.1 e posteriores.

Nome do parâmetro	Permite modificação	Observações
innodb_autoextend_increment	Sim	
innodb_buffer_pool_dump_at_shutdown	Não	
innodb_buffer_pool_dump_now	Não	
innodb_buffer_pool_filename	Não	
innodb_buffer_pool_load_abort	Não	
innodb_buffer_pool_load_at_startup	Não	
innodb_buffer_pool_load_now	Não	
innodb_buffer_pool_size	Sim	O valor padrão é representado por uma fórmula. Para obter detalhes sobre como o valor DBInstanceClassMemory na fórmula é calculado, consulte Variáveis de fórmulas de parâmetros de banco de dados.
innodb_change_buffer_max_size	Não	O Aurora MySQL não usa o buffer de mudança do InnoDB.
innodb_compression_failure_threshold_pct	Sim	
innodb_compression_level	Sim	

Nome do parâmetro	Permite modificação	Observações
innodb_compression_pad_pct_max	Sim	
innodb_concurrency_tickets	Sim	Modificar este parâmetro não tem efeito, porque innodb_thread_concurrency é sempre 0 para Aurora.
innodb_deadlock_detect	Sim	Esta opção é usada para desabilitar a detecção de deadlock no Aurora MySQL versão 2.11 e posterior e versão 3. Em sistemas de alta simultaneidade, a detecção de deadlock pode causar uma desaceleração quando vários segmentos aguardam o mesmo bloqueio. Consulte a documentação do MySQL para obter mais informações sobre esse parâmetro.
innodb_file_format	Sim	Retirado do Aurora MySQL versão 3.
innodb_flushing_avg_loops	Não	
innodb_force_load_corrupted	Não	
innodb_ft_aux_table	Sim	
innodb_ft_cache_size	Sim	
innodb_ft_enable_stopword	Sim	
innodb_ft_server_stopword_table	Sim	

Nome do parâmetro	Permite modificação	Observações
innodb_ft_user_stopword_table	Sim	
innodb_large_prefix	Sim	Retirado do Aurora MySQL versão 3.
innodb_lock_wait_timeout	Sim	
innodb_log_compressed_pages	Não	
innodb_lru_scan_depth	Sim	
innodb_max_purge_lag	Sim	
innodb_max_purge_lag_delay	Sim	
innodb_monitor_disable	Sim	
innodb_monitor_enable	Sim	
innodb_monitor_reset	Sim	
innodb_monitor_reset_all	Sim	
innodb_old_blocks_pct	Sim	
innodb_old_blocks_time	Sim	
innodb_open_files	Sim	
innodb_print_all_deadlocks	Sim	Quando ativado, registra informações sobre todos os deadlocks do InnoDB no log de erros do Aurora MySQL. Para ter mais informações, consulte Minimizar e solucionar problemas de deadlocks do Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
innodb_random_read_ahead	Sim	
innodb_read_ahead_threshold	Sim	
innodb_read_io_threads	Não	
innodb_read_only	Não	O Aurora MySQL gerencia os estados somente leitura e de leitura/gravação de instâncias de banco de dados com base no tipo de cluster. Por exemplo, um cluster provisionado tem uma instância de banco de dados de leitura/gravação (a instância primária) e qualquer outra instância no cluster é somente leitura (as réplicas do Aurora).
innodb_replication_delay	Sim	
innodb_sort_buffer_size	Sim	
innodb_stats_auto_recalc	Sim	
innodb_stats_method	Sim	
innodb_stats_on_metadata	Sim	
innodb_stats_persistent	Sim	
innodb_stats_persistent_sample_pages	Sim	
innodb_stats_transient_sample_pages	Sim	
innodb_thread_concurrency	Não	

Nome do parâmetro	Permite modificação	Observações
innodb_thread_sleep_delay	Sim	Modificar este parâmetro não tem efeito, porque innodb_thread_concurrency é sempre 0 para o Aurora.
interactive_timeout	Sim	O Aurora avalia o valor mínimo de interactive_timeout e wait_timeout . Em seguida, ele utiliza esse mínimo como o tempo limite para encerrar todas as sessões inativas, interativas e não interativas.
internal_tmp_disk_storage_engine	Sim	Controla qual mecanismo de armazenamento na memória é usado para tabelas temporárias internas. Os valores permitidos são INNODB e MYISAM. Este parâmetro é aplicável ao Aurora MySQL versão 2.
internal_tmp_mem_storage_engine	Sim	Controla qual mecanismo de armazenamento na memória é usado para tabelas temporárias internas. Os valores permitidos são MEMORY e TempTable . Este parâmetro é aplicável ao Aurora MySQL versão 3.
join_buffer_size	Sim	
keep_files_on_create	Sim	

Nome do parâmetro	Permite modificação	Observações
`key_buffer_size`	Sim	Cache de chaves para tabelas MyISAM. Para ter mais informações, consulte keycache->cache_lock mutex.
`key_cache_age_threshold`	Sim	
`key_cache_block_size`	Sim	
`key_cache_division_limit`	Sim	
`local_infile`	Sim	
`lock_wait_timeout`	Sim	
`log-bin`	Não	Definir `binlog_format` como `STATEMENT` , `MIXED` ou `ROW` define automaticamente `log-bin` como `ON`. Definir `binlog_format` como `OFF` define automaticamente `log-bin` como `OFF`. Para ter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).
`log_bin_trust_function_creators`	Sim	
`log_bin_use_v1_row_events`	Sim	Retirado do Aurora MySQL versão 3.
`log_error`	Não	

Nome do parâmetro	Permite modificação	Observações
log_error_suppression_list	Sim	Especifica uma lista de códigos de erros que não estão registrados no log de erros do MySQL. Isso permite que você ignore certas condições de erro não críticas para ajudar a manter seus logs de erros limpos. Para ter mais informações, consulte log_error_suppression_list na documentação do MySQL. Esse parâmetro é aplicável ao Aurora MySQL versão 3.03 e posterior.
log_output	Sim	
log_queries_not_using_indexes	Sim	
log_slave_updates	Não	Aurora MySQL versão 2. Use log_replica_updates no Aurora MySQL versão 3.
log_replica_updates	Não	Aurora MySQL versão 3
log_throttle_queries_not_using_indexes	Sim	
log_warnings	Sim	Retirado do Aurora MySQL versão 3.
long_query_time	Sim	

Nome do parâmetro	Permite modificação	Observações
`low_priority_updates`	Sim	As operações `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` aguardam até que não haja nenhuma operação `SELECT` pendente. Esse parâmetro só afeta mecanismos de armazenamento que usam apenas bloqueio em nível de tabela (MyISAM, MEMORY, MERGE). Este parâmetro é aplicável ao Aurora MySQL versão 3.
`max_allowed_packet`	Sim	
`max_binlog_cache_size`	Sim	
`max_binlog_size`	Não	
`max_binlog_stmt_cache_size`	Sim	
`max_connect_errors`	Sim	
`max_connections`	Sim	O valor padrão é representado por uma fórmula. Para obter detalhes sobre como o valor `DBInstanceClassMemory` na fórmula é calculado, consulte Variáveis de fórmulas de parâmetros de banco de dados. Para obter os valores padrão, dependendo da classe da instância, consulte Número máximo de conexões com uma instância de bancos de dados Aurora MySQL.

Nome do parâmetro	Permite modificação	Observações
`max_delayed_threads`	Sim	Define o número máximo de threads para lidar com instruções `INSERT DELAYED`. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`max_error_count`	Sim	O número máximo de mensagens de erro, advertência e observação que devem ser armazenadas para exibição. Este parâmetro é aplicável ao Aurora MySQL versão 3.
`max_execution_time`	Sim	O tempo limite para executar declarações `SELECT`, em milissegundos. O valor pode ser de `0` a `18446744073709551615`. Quando definido como `0`, não há tempo limite. Para ter mais informações, consulte max_execution_time na documentação do MySQL.
`max_heap_table_size`	Sim	
`max_insert_delayed_threads`	Sim	
`max_join_size`	Sim	
`max_length_for_sort_data`	Sim	Retirado do Aurora MySQL versão 3.
`max_prepared_stmt_count`	Sim	
`max_seeks_for_key`	Sim	

Nome do parâmetro	Permite modificação	Observações
max_sort_length	Sim	
max_sp_recursion_depth	Sim	
max_tmp_tables	Sim	Retirado do Aurora MySQL versão 3.
max_user_connections	Sim	
max_write_lock_count	Sim	
metadata_locks_cache_size	Sim	Retirado do Aurora MySQL versão 3.
min_examined_row_limit	Sim	Use esse parâmetro para evitar que consultas que examinam menos do que o número especificado de linhas sejam registradas em log. Este parâmetro é aplicável ao Aurora MySQL versão 3.
myisam_data_pointer_size	Sim	
myisam_max_sort_file_size	Sim	
myisam_mmap_size	Sim	
myisam_sort_buffer_size	Sim	
myisam_stats_method	Sim	
myisam_use_mmap	Sim	
net_buffer_length	Sim	
net_read_timeout	Sim	
net_retry_count	Sim	

Nome do parâmetro	Permite modificação	Observações
net_write_timeout	Sim	
old-style-user-limits	Sim	
old_passwords	Sim	Retirado do Aurora MySQL versão 3.
optimizer_prune_level	Sim	
optimizer_search_depth	Sim	
optimizer_switch	Sim	Para obter informações sobre os recursos do Aurora MySQL que usam essa alternância, consulte Melhores práticas do Amazon Aurora MySQL.
optimizer_trace	Sim	
optimizer_trace_features	Sim	
optimizer_trace_limit	Sim	
optimizer_trace_max_mem_size	Sim	
optimizer_trace_offset	Sim	
performance-schema-consumer-events-waits-current	Sim	
performance-schema-instrument	Sim	
performance_schema	Sim	
performance_schema_accounts_size	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_consumer_global_instrumentation	Sim	
performance_schema_consumer_thread_instrumentation	Sim	
performance_schema_consumer_events_stages_current	Sim	
performance_schema_consumer_events_stages_history	Sim	
performance_schema_consumer_events_stages_history_long	Sim	
performance_schema_consumer_events_statements_current	Sim	
performance_schema_consumer_events_statements_history	Sim	
performance_schema_consumer_events_statements_history_long	Sim	
performance_schema_consumer_events_waits_history	Sim	
performance_schema_consumer_events_waits_history_long	Sim	
performance_schema_consumer_statements_digest	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_digests_size	Sim	
performance_schema_events_stages_history_long_size	Sim	
performance_schema_events_stages_history_size	Sim	
performance_schema_events_statements_history_long_size	Sim	
performance_schema_events_statements_history_size	Sim	
performance_schema_events_transactions_history_long_size	Sim	
performance_schema_events_transactions_history_size	Sim	
performance_schema_events_waits_history_long_size	Sim	
performance_schema_events_waits_history_size	Sim	
performance_schema_hosts_size	Sim	
performance_schema_max_cond_classes	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_max_cond_instances	Sim	
performance_schema_max_digest_length	Sim	
performance_schema_max_file_classes	Sim	
performance_schema_max_file_handles	Sim	
performance_schema_max_file_instances	Sim	
performance_schema_max_index_stat	Sim	
performance_schema_max_memory_classes	Sim	
performance_schema_max_metadata_locks	Sim	
performance_schema_max_mutex_classes	Sim	
performance_schema_max_mutex_instances	Sim	
performance_schema_max_prepared_statements_instances	Sim	
performance_schema_max_program_instances	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_max_rwlock_classes	Sim	
performance_schema_max_rwlock_instances	Sim	
performance_schema_max_socket_classes	Sim	
performance_schema_max_socket_instances	Sim	
performance_schema_max_sql_text_length	Sim	
performance_schema_max_stage_classes	Sim	
performance_schema_max_statement_classes	Sim	
performance_schema_max_statement_stack	Sim	
performance_schema_max_table_handles	Sim	
performance_schema_max_table_instances	Sim	
performance_schema_max_table_lock_stat	Sim	
performance_schema_max_thread_classes	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_max_thread_instances	Sim	
performance_schema_session_connect_attrs_size	Sim	
performance_schema_setup_actors_size	Sim	
performance_schema_setup_objects_size	Sim	

Nome do parâmetro	Permite modificação	Observações
performance_schema_show_processlist	Sim	Esse parâmetro determina qual implementação SHOW PROCESSLIST usar: • A implementação padrão itera em todos os segmentos ativos de dentro do gerenciador de segmentos enquanto mantém um mutex global. Isso pode causar lentidão de performance, principalmente em sistemas ocupados. • A implementação SHOW PROCESSLIST alternativa é baseada na tabela Esquema de performance processlist . Essa implementação consulta dados de segmento ativos dessa tabela em vez do gerenciador de segmentos e não exige um mutex. Esse parâmetro se aplica ao Aurora MySQL versão 2.12 e posterior e versão 3.
performance_schema_users_size	Sim	
pid_file	Não	
plugin_dir	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.

Nome do parâmetro	Permite modificação	Observações
port	Não	O Aurora MySQL gerencia as propriedades de conexão e impõe configurações consistentes para todas as instâncias de banco de dados em um cluster.
preload_buffer_size	Sim	O tamanho do buffer que é alocado ao pré-carregar índices. Este parâmetro é aplicável ao Aurora MySQL versão 3.
profiling_history_size	Sim	
query_alloc_block_size	Sim	
query_cache_limit	Sim	Retirado do Aurora MySQL versão 3.
query_cache_min_res_unit	Sim	Retirado do Aurora MySQL versão 3.
query_cache_size	Sim	O valor padrão é representado por uma fórmula. Para obter detalhes sobre como o valor DBInstanceClassMemory na fórmula é calculado, consulte Variáveis de fórmulas de parâmetros de banco de dados. Retirado do Aurora MySQL versão 3.
query_cache_type	Sim	Retirado do Aurora MySQL versão 3.
query_cache_wlock_invalidate	Sim	Retirado do Aurora MySQL versão 3.
query_prealloc_size	Sim	
range_alloc_block_size	Sim	

Nome do parâmetro	Permite modificação	Observações
`read_buffer_size`	Sim	

Nome do parâmetro	Permite modificação	Observações
`read_only`	Sim	Quando esse parâmetro é ativado, o servidor não permite atualizações, exceto aquelas executadas por threads de réplica. Para o Aurora MySQL versão 2, os valores válidos são os seguintes: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` : `ON` para réplicas de leitura. Este é o valor padrão. • `{TrueIfClusterReplica}` : `ON` para instâncias em clusters de réplicas, como réplicas de leitura entre regiões, clusters secundários em um banco de dados global do Aurora e implantações azul/verde. Recomendamos que você use o grupo de parâmetros do cluster de banco de dados na versão 2 do Aurora MySQL para garantir que o parâmetro `read_only` seja aplicado a novas instâncias do gravador em caso de failover. **Note** As instâncias do leitor são sempre somente leitura,

Nome do parâmetro	Permite modificação	Observações
		porque o Aurora MySQL define `innodb_read_only` como 1 em todos os leitores. Portanto, `read_only` é redundante em instâncias do leitor. Retirado no nível da instância do Aurora MySQL versão 3.
`read_rnd_buffer_size`	Sim	
`relay-log`	Não	
`relay_log_info_repository`	Sim	Retirado do Aurora MySQL versão 3.
`relay_log_recovery`	Não	
`replica_checkpoint_group`	Sim	Aurora MySQL versão 3
`replica_checkpoint_period`	Sim	Aurora MySQL versão 3
`replica_parallel_workers`	Sim	Aurora MySQL versão 3
`replica_pending_jobs_size_max`	Sim	Aurora MySQL versão 3
`replica_skip_errors`	Sim	Aurora MySQL versão 3
`replica_sql_verify_checksum`	Sim	Aurora MySQL versão 3
`safe-user-create`	Sim	

Nome do parâmetro	Permite modificação	Observações
`secure_auth`	Sim	Este parâmetro está sempre ativado no Aurora MySQL versão 2. Se tentar desativá-lo, um erro será gerado. Retirado do Aurora MySQL versão 3.
`secure_file_priv`	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
`show_create_table_verbosity`	Sim	Habilitar essa variável faz com que SHOW_CREATE_TABLE exiba o `ROW_FORMAT` independentemente de ser o formato padrão. Esse parâmetro se aplica ao Aurora MySQL versão 2.12 e posterior e versão 3.
`skip-slave-start`	Não	
`skip_external_locking`	Não	
`skip_show_database`	Sim	
`slave_checkpoint_group`	Sim	Aurora MySQL versão 2. Use `replica_checkpoint_group` no Aurora MySQL versão 3.
`slave_checkpoint_period`	Sim	Aurora MySQL versão 2. Use `replica_checkpoint_period` no Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`slave_parallel_workers`	Sim	Aurora MySQL versão 2. Use `replica_parallel_workers` no Aurora MySQL versão 3.
`slave_pending_jobs_size_max`	Sim	Aurora MySQL versão 2. Use `replica_pending_jobs_size_max` no Aurora MySQL versão 3.
`slave_sql_verify_checksum`	Sim	Aurora MySQL versão 2. Use `replica_sql_verify_checksum` no Aurora MySQL versão 3.
`slow_launch_time`	Sim	
`slow_query_log`	Sim	Para obter instruções sobre como fazer upload dos logs no CloudWatch Logs, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.
`slow_query_log_file`	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
`socket`	Não	
`sort_buffer_size`	Sim	
`sql_mode`	Sim	
`sql_select_limit`	Sim	
`stored_program_cache`	Sim	

Nome do parâmetro	Permite modificação	Observações
sync_binlog	Não	
sync_master_info	Sim	
sync_source_info	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3.
sync_relay_log	Sim	Retirado do Aurora MySQL versão 3.
sync_relay_log_info	Sim	
sysdate-is-now	Sim	
table_cache_element_entry_ttl	Não	
table_definition_cache	Sim	O valor padrão é representado por uma fórmula. Para obter detalhes sobre como o valor DBInstanceClassMemory na fórmula é calculado, consulte Variáveis de fórmulas de parâmetros de banco de dados.
table_open_cache	Sim	O valor padrão é representado por uma fórmula. Para obter detalhes sobre como o valor DBInstanceClassMemory na fórmula é calculado, consulte Variáveis de fórmulas de parâmetros de banco de dados.
table_open_cache_instances	Sim	
temp-pool	Sim	Retirado do Aurora MySQL versão 3.

Nome do parâmetro	Permite modificação	Observações
`temptable_max_mmap`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3. Para obter detalhes, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3.
`temptable_max_ram`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3. Para obter detalhes, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3.
`temptable_use_mmap`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3. Para obter detalhes, consulte Novo comportamento de tabela temporária no Aurora MySQL versão 3.
`thread_cache_size`	Sim	O número de threads para armazenar em cache. Esse parâmetro se aplica ao Aurora MySQL versões 2 e 3.
`thread_handling`	Não	
`thread_stack`	Sim	
`timed_mutexes`	Sim	

Nome do parâmetro	Permite modificação	Observações
`tmp_table_size`	Sim	Define o tamanho máximo das tabelas temporárias na memória criadas pelo mecanismo de armazenamento `MEMORY` no Aurora MySQL versão 3. No Aurora MySQL versão 3.04 e posterior, define o tamanho máximo das tabelas temporárias na memória criadas pelo mecanismo de armazenamento `TempTable` quando `aurora_tmptable_enable_per_table_limit` está `ON`. Para ter mais informações, consulte Limitar o tamanho de tabelas temporárias internas na memória.
`tmpdir`	Não	O Aurora MySQL utiliza instâncias gerenciadas em que não é possível acessar o sistema de arquivos diretamente.
`transaction_alloc_block_size`	Sim	
`transaction_isolation`	Sim	Este parâmetro é aplicável ao Aurora MySQL versão 3. Ele substitui `tx_isolation` .
`transaction_prealloc_size`	Sim	

Nome do parâmetro	Permite modificação	Observações
`tx_isolation`	Sim	Retirado do Aurora MySQL versão 3. Ele é substituído por `transaction_isolation` .
`updatable_views_with_limit`	Sim	
`validate-password`	Não	
`validate_password_dictionary_file`	Não	
`validate_password_length`	Não	
`validate_password_mixed_case_count`	Não	
`validate_password_number_count`	Não	
`validate_password_policy`	Não	
`validate_password_special_char_count`	Não	
`wait_timeout`	Sim	O Aurora avalia o valor mínimo de `interactive_timeout` e `wait_timeout` . Em seguida, ele utiliza esse mínimo como o tempo limite para encerrar todas as sessões inativas, interativas e não interativas.

Parâmetros do MySQL não se aplicam ao Aurora MySQL

Por causa das diferenças de arquitetura entre o Aurora MySQL e o MySQL, alguns parâmetros do MySQL não se aplicam ao Aurora MySQL.

Os seguintes parâmetros do MySQL não se aplicam ao Aurora MySQL. Essa lista não é exaustiva.

- `activate_all_roles_on_login`: este parâmetro não se aplica ao Aurora MySQL versão 2. Ele está disponível no Aurora MySQL versão 3.
- `big_tables`
- `bind_address`
- `character_sets_dir`
- `innodb_adaptive_flushing`
- `innodb_adaptive_flushing_lwm`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
- `innodb_change_buffering`
- `innodb_checksum_algorithm`
- `innodb_data_file_path`
- `innodb_dedicated_server`
- `innodb_doublewrite`
- `innodb_flush_log_at_timeout`: esse parâmetro não se aplica ao Aurora MySQL. Para ter mais informações, consulte Configurar a frequência com que o buffer de log é liberado.
- `innodb_flush_method`
- `innodb_flush_neighbors`
- `innodb_io_capacity`
- `innodb_io_capacity_max`
- `innodb_log_buffer_size`
- `innodb_log_file_size`
- `innodb_log_files_in_group`
- `innodb_log_spin_cpu_abs_lwm`
- `innodb_log_spin_cpu_pct_hwm`
- `innodb_log_writer_threads`
- `innodb_max_dirty_pages_pct`
- `innodb_numa_interleave`
- `innodb_page_size`

- innodb_redo_log_capacity
- innodb_redo_log_encrypt
- innodb_undo_log_encrypt
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- innodb_use_native_aio
- innodb_write_io_threads

Variáveis de status globais do Aurora MySQL

O Aurora MySQL inclui variáveis de status do MySQL da comunidade e variáveis exclusivas do Aurora. É possível examinar essas variáveis para saber o que está acontecendo no mecanismo de banco de dados. Para ter mais informações sobre as variáveis de status no MySQL da comunidade, consulte Server Status Variables na documentação do MySQL 8.0 da comunidade.

Você pode encontrar os valores atuais das variáveis de status globais do Aurora MySQL usando uma declaração como a seguinte:

```
show global status like '%aurora%';
```

> **Note**
>
> As variáveis de status global são limpas quando o mecanismo de banco de dados é reiniciado.

A tabela a seguir descreve as variáveis de status globais que o Aurora MySQL usa.

Name (Nome)	Descrição
AuroraDb_commits	O número total de confirmações desde a última reinicialização.
AuroraDb_commit_latency	A latência agregada de confirmação desde a última reinicialização.

Name (Nome)	Descrição
AuroraDb_ddl_stmt_duration	A latência agregada de DDL desde a última reinicialização.
AuroraDb_select_stmt_duration	A latência agregada de declaraçã o SELECT desde a última reinicialização.
AuroraDb_insert_stmt_duration	A latência agregada de declaraçã o INSERT desde a última reinicialização.
AuroraDb_update_stmt_duration	A latência agregada de declaraçã o UPDATE desde a última reinicialização.
AuroraDb_delete_stmt_duration	A latência agregada de declaraçã o DELETE desde a última reinicialização.
Aurora_binlog_io_cache_allocated	O número de bytes alocados para o cache de E/S do binlog.
Aurora_binlog_io_cache_read _requests	O número de solicitações de leitura feitas no cache de E/S do binlog.
Aurora_binlog_io_cache_reads	O número de solicitações de leitura que foram atendidas do cache de E/S do binlog.
Aurora_enhanced_binlog	Indica se o binlog aprimorado está habilitad o ou desabilitado para essa instância de banco de dados. Para obter mais informaçõ es, consulte Configurar o log binário avançado para Aurora MySQL.
Aurora_external_connection_count	O número de conexões de banco de dados com a instância de banco de dados, excluindo as conexões de serviço do RDS usadas para verificações de integridade do banco de dados.

Name (Nome)	Descrição
`Aurora_fast_insert_cache_hits`	Um contador que é incrementado quando o cursor em cache é recuperado e verificado com êxito. Para receber mais informações sobre o cache de inserção rápida, consulte Melhorias de performance do Amazon Aurora MySQL.
`Aurora_fast_insert_cache_misses`	Um contador que é incrementado quando o cursor em cache não é mais válido e o Aurora realiza um percurso de índice normal. Para receber mais informações sobre o cache de inserção rápida, consulte Melhorias de performance do Amazon Aurora MySQL.
`Aurora_fts_cache_memory_used`	A quantidade de memória em bytes que o sistema de pesquisa de texto completo do InnoDB está usando. Essa variável aplica-se ao Aurora MySQL versão 3.07 e posterior.
`Aurora_fwd_master_dml_stmt_count`	Número total de declarações DML encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 2.
`Aurora_fwd_master_dml_stmt_duration`	A duração total das declarações DML encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 2.
`Aurora_fwd_master_errors_rpc_timeout`	O número de vezes que uma conexão encaminhada não foi estabelecida no gravador.
`Aurora_fwd_master_errors_session_limit`	O número de consultas encaminhadas que são rejeitadas devido a `session full` no gravador.

Name (Nome)	Descrição
`Aurora_fwd_master_errors_session_timeout`	O número de vezes que uma sessão de encaminhamento é encerrada devido ao tempo limite do gravador.
`Aurora_fwd_master_open_sessions`	O número de sessões encaminhadas na instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 2.
`Aurora_fwd_master_select_stmt_count`	O número total de declarações `SELECT` encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 2.
`Aurora_fwd_master_select_stmt_duration`	A duração total das declarações `SELECT` encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 2.
`Aurora_fwd_writer_dml_stmt_count`	Número total de declarações DML encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 3.
`Aurora_fwd_writer_dml_stmt_duration`	A duração total das declarações DML encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 3.
`Aurora_fwd_writer_errors_rpc_timeout`	O número de vezes que uma conexão encaminhada não foi estabelecida no gravador.
`Aurora_fwd_writer_errors_session_limit`	O número de consultas encaminhadas que são rejeitadas devido a `session full` no gravador.

Name (Nome)	Descrição
`Aurora_fwd_writer_errors_session_timeout`	O número de vezes que uma sessão de encaminhamento é encerrada devido ao tempo limite do gravador.
`Aurora_fwd_writer_open_sessions`	O número de sessões encaminhadas na instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 3.
`Aurora_fwd_writer_select_stmt_count`	O número total de declarações `SELECT` encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 3.
`Aurora_fwd_writer_select_stmt_duration`	A duração total das declarações `SELECT` encaminhadas para essa instância de banco de dados do gravador. Esta variável é aplicável ao Aurora MySQL versão 3.
`Aurora_lockmgr_buffer_pool_memory_used`	A quantidade de memória de grupo de buffer em bytes que o gerenciador de bloqueio do Aurora MySQL está usando.
`Aurora_lockmgr_memory_used`	A quantidade de memória em bytes que o gerenciador de bloqueio do Aurora MySQL está usando.
`Aurora_ml_actual_request_cnt`	A contagem agregada de solicitações que o Aurora MySQL faz nos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados. Para obter mais informações, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.

Name (Nome)	Descrição
`Aurora_ml_actual_response_cnt`	A contagem agregada de respostas que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados. Para obter mais informaçõ es, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.
`Aurora_ml_cache_hit_cnt`	A contagem agregada de acertos do cache interno que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados. Para obter mais informações, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.
`Aurora_ml_logical_request_cnt`	O número de solicitações lógicas que a instância de banco de dados avaliou para serem enviadas aos serviços de machine learning do Aurora desde a última redefiniç ão de status. Dependendo se o lote foi ou não usado, esse valor pode ser maior que `Aurora_ml_actual_request_cnt` . Para obter mais informações, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.
`Aurora_ml_logical_response_cnt`	A contagem agregada de respostas que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados. Para obter mais informaçõ es, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.

Name (Nome)	Descrição
`Aurora_ml_retry_request_cnt`	O número de novas solicitações que a instância de banco de dados enviou aos serviços de machine learning do Aurora desde a última redefinição de status. Para obter mais informações, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.
`Aurora_ml_single_request_cnt`	A contagem agregada de funções do machine learning do Aurora que são avaliadas pelo modo que não seja em lote entre todas as consultas executadas por usuários da instância de banco de dados. Para obter mais informaçõ es, consulte Usar o machine learning do Amazon Aurora com o Aurora MySQL.
`aurora_oom_avoidance_recove ry_state`	Indica se a recuperação para evitar falta de memória (OOM) do Aurora está no estado `ACTIVE` ou `INACTIVE` para essa instância de banco de dados. Essa variável aplica-se ao Aurora MySQL versão 3.06.0 e posterior.
`aurora_oom_reserved_mem_enter_kb`	Representa o limite para entrar no estado `RESERVED` no mecanismo de tratamento de OOM do Aurora. Quando a memória disponível no servidor fica abaixo desse limite, `aurora_oom_status` muda para `RESERVED`, indicando que o servidor está se aproximando de um nível crítico de uso da memória. Essa variável aplica-se ao Aurora MySQL versão 3.06.0 e posterior.

Name (Nome)	Descrição
`aurora_oom_reserved_mem_exit_kb`	Representa o limite para sair do estado `RESERVED` no mecanismo de tratamento de OOM do Aurora. Quando a memória disponível no servidor ultrapassa esse limite, o `aurora_oom_status` reverte para `NORMAL`, indicando que o servidor retornou a um estado mais estável com recursos de memória suficientes. Essa variável aplica-se ao Aurora MySQL versão 3.06.0 e posterior.
`aurora_oom_status`	Representa o status atual de OOM dessa instância de banco de dados. Quando o valor é `NORMAL`, indica que há recursos de memória suficientes. Se o valor mudar para `RESERVED`, isso indica que o servidor tem pouca memória disponível. As ações são realizadas com base na configuração dos parâmetros `aurora_oom_response`. Para obter mais informações, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL. Essa variável aplica-se ao Aurora MySQL versão 3.06.0 e posterior.
`Aurora_pq_bytes_returned`	O número de bytes de estruturas de dados de tupla transmitidos para o nó de cabeçalho durante as consultas paralelas. Divida por 16.384 para comparar com `Aurora_pq_pages_pushed_down`.

Name (Nome)	Descrição
`Aurora_pq_max_concurrent_requests`	O número máximo de sessões de consulta paralela que podem ser executadas simultaneamente nesta instância de bancos de dados Aurora. Esse é um número fixo, que depende da classe da instância de banco de dados da AWS.
`Aurora_pq_pages_pushed_down`	O número de páginas de dados (cada uma com um tamanho fixo de 16 KiB) em que a consulta paralela evitou uma transmissão de rede para o nó de cabeçalho.
`Aurora_pq_request_attempted`	O número de sessões de consultas paralelas solicitadas. Esse valor pode representar mais de uma sessão por consulta, dependendo dos elementos SQL, como subconsultas e junções.
`Aurora_pq_request_executed`	O número de sessões de consultas paralelas executadas com êxito.
`Aurora_pq_request_failed`	O número de sessões de consultas paralelas que retornaram um erro para o cliente. Em alguns casos, uma solicitação por uma consulta paralela pode falhar, por exemplo, devido a um problema na camada de armazenamento. Nesses casos, a parte da consulta que falhou é reprocessada usando um mecanismo de consulta não paralelo. Se a consulta reprocessada também falhar, será retornado um erro para o cliente e o contador será incrementado.

Name (Nome)	Descrição
`Aurora_pq_request_in_progress`	O número de sessões de consultas paralelas em andamento no momento. Esse número se aplica à instância de bancos de dados Aurora em particular à qual você está conectado, e não a todo o cluster de bancos de dados Aurora. Para saber se uma instância de banco de dados está próxima do limite de simultane idade, compare este valor a `Aurora_pq _max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	O número de vezes em que a consulta paralela não foi escolhida para atender uma consulta. Este valor é a soma de vários outros contadore s mais granulares. Uma instrução `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chose n_below_min_rows`	O número de vezes em que a consulta paralela não foi escolhida devido ao número de linhas na tabela. Uma declaração `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chose n_column_bit`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela devido a um tipo de dados sem suporte na lista de colunas projetadas.

Name (Nome)	Descrição
`Aurora_pq_request_not_chosen_column_geometry`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem colunas com o tipo de dados `GEOMETRY`. Para saber mais sobre as versões do Aurora MySQL que removem essa limitação, consulte Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3.
`Aurora_pq_request_not_chosen_column_lob`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela inclui colunas com um tipo de dados `LOB` ou colunas `VARCHAR` que são armazenadas externamente devido ao comprimento declarado. Para saber mais sobre as versões do Aurora MySQL que removem essa limitação, consulte Fazer upgrade de clusters de consulta paralela para o Aurora MySQL versão 3.
`Aurora_pq_request_not_chosen_column_virtual`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela contém uma coluna virtual.
`Aurora_pq_request_not_chosen_custom_charset`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem colunas com um conjunto de caracteres personalizado.

Name (Nome)	Descrição
`Aurora_pq_request_not_chosen_fast_ddl`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela está sendo alterada atualmente por uma instrução `ALTER` DDL rápida.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	O número de vezes em que a consulta paralela não foi escolhida, ainda que menos de 95 por cento dos dados da tabela já estivessem no grupo de buffers, porque não havia uma quantidade suficiente de dados da tabela fora do buffer que fizesse a consulta paralela valer a pena.
`Aurora_pq_request_not_chosen_full_text_index`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela tem índices de texto completo.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	O número de vezes em que a consulta paralela não foi escolhida porque uma alta porcentag em de dados da tabela (no momento, maior do que 95 por cento) já se encontrava no grupo de buffers. Nesses casos, o otimizador determina que a leitura dos dados a partir do grupo de buffers é mais eficiente. Uma instrução `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_index_hint`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta inclui uma dica de índice.

Name (Nome)	Descrição
`Aurora_pq_request_not_chosen_innodb_table_format`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a tabela usa um formato de linha InnoDB que não é compatível. A consulta paralela do Aurora só se aplica aos formatos de linha `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	O número de solicitações de consultas paralelas que usaram o caminho de processamento de consultas não paralelas, devido à execução da consulta ter sido iniciada dentro de uma transação de execução demorada. Uma instrução `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_no_where_clause`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta não inclui nenhuma cláusula `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta usa uma verificação de intervalo em um índice.
`Aurora_pq_request_not_chosen_row_length_too_long`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque o comprimento total combinado de todas as colunas é muito longo.

Name (Nome)	Descrição
`Aurora_pq_request_not_chosen_small_table`	O número de vezes em que a consulta paralela não foi escolhida devido ao tamanho total da tabela, o que é determinado pelo número de linhas e pelo comprimento médio da linha. Uma instrução `EXPLAIN` pode incrementar esse contador mesmo que a consulta não seja realmente executada.
`Aurora_pq_request_not_chosen_temporary_table`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta se refere a tabelas temporárias que usam os tipos de tabela `MyISAM` ou `memory` que não têm suporte.
`Aurora_pq_request_not_chosen_tx_isolation`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta usa um nível de isolamento de transação sem suporte. Em instâncias de banco de dados do leitor, a consulta paralela se aplica somente aos níveis de isolamento `REPEATABLE READ` e `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	O número de solicitações de consulta paralela que usam o caminho de processamento de consulta não paralela porque a consulta faz parte de uma instrução `UPDATE` ou `DELETE`.

Name (Nome)	Descrição
`Aurora_pq_request_not_chosen_unsupported_access`	O número de solicitações de consultas paralelas que usam o caminho de processamento de consultas não paralelas porque a cláusula `WHERE` não atende aos critérios de consulta paralela. Esse resultado pode ocorrer se a consulta não exigir uma varredura de dados intensiva ou se a consulta for uma instrução `DELETE` ou `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	O número de solicitações de consulta paralelas que usam o caminho de processamento de consulta não paralelo porque o cluster de banco de dados do Aurora MySQL não está usando uma configuração de armazenamento de cluster compatível do Aurora. Para obter mais informações, consulte Limitações. Esse parâmetro é aplicável ao Aurora MySQL versão 3.04 e posterior.
`Aurora_pq_request_throttled`	O número de vezes em que a consulta paralela não foi escolhida devido ao número máximo de consultas paralelas simultâneas em execução em uma determinada instância de bancos de dados Aurora.
`Aurora_repl_bytes_received`	Número de bytes replicados em uma instância de banco de dados do leitor do Aurora MySQL desde a última reinicialização. Para obter mais informações, consulte Replicação com o Amazon Aurora MySQL.

Name (Nome)	Descrição
`Aurora_reserved_mem_exceeded_incidents`	O número de vezes, desde a última reinicialização, que o engenheiro excedeu os limites de memória reservada. Se `aurora_oom_response` estiver configurado, esse limite definirá quando as atividades para evitar falta de memória (OOM) serão acionadas. Para receber mais informações sobre a resposta OOM do Aurora MySQL, consulte Solucionar problemas de falta de memória em bancos de dados do Aurora MySQL.
`aurora_temptable_max_ram_allocation`	A quantidade máxima de memória, em bytes, usada em qualquer momento por tabelas temporárias internas desde a última reinicialização.
`aurora_temptable_ram_allocation`	A quantidade atual de memória, em bytes, usada pelas tabelas temporárias internas.
`Aurora_in_memory_relaylog_status`	O status atual do recurso de log de retransmissão na memória. O valor pode ser ENABLED ou DISABLED.
`Aurora_in_memory_relaylog_disabled_reason`	Mostra o motivo do status atual do recurso de log de retransmissão na memória; se o recurso estiver desabilitado, será exibida uma mensagem explicando por que o recurso está desabilitado.

Name (Nome)	Descrição
`Aurora_in_memory_relaylog_fallback_count`	Mostra o número total de fallbacks do recurso de log de retransmissão na memória para o modo de log de retransmissão persistente (legado). O fallback pode ser causado por um único evento maior que o tamanho do cache (no momento, 128 MB) ou por uma nova tentativa de transação que excede o limite de repetição de transação de réplica replica_transaction_retries.
`Aurora_in_memory_relaylog_recovery_count`	Mostra o número total de recuperações de log de retransmissão na memória executadas automaticamente. Essa contagem inclui o número total de fallbacks e o número de retornos do modo automático para o modo de log de retransmissão na memória após os fallbacks temporários.
`Aurora_thread_pool_thread_count`	O número atual de segmentos no grupo de segmentos do Aurora. Para receber mais informações sobre o grupo de segmentos no Aurora MySQL, consulte Pool de threads.
`Aurora_tmz_version`	Indica a versão atual das informações de fuso horário usadas pelo cluster de banco de dados. Os valores seguem o formato Internet Assigned Numbers Authority (IANA): `YYYYsuffix` ; por exemplo, `2022a` e `2023c`. Esse parâmetro se aplica ao Aurora MySQL versão 2.12 e posterior e versão 3.04 e posterior.

Name (Nome)	Descrição
`Aurora_zdr_oom_threshold`	Representa o limite de memória, em quilobytes (KB), para que uma instância de banco de dados do Aurora inicie uma reinicialização com tempo de inatividade zero (ZDR) para se recuperar de possíveis problemas relacionados à memória.
`server_aurora_das_running`	Indica se Database Activity Streams (DAS) está habilitado ou desabilitado nessa instância de banco de dados. Para obter mais informações, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.

Variáveis de status do MySQL não se aplicam ao Aurora MySQL

Por causa das diferenças de arquitetura entre o Aurora MySQL e o MySQL, algumas variáveis de status do MySQL não se aplicam ao Aurora MySQL.

As variáveis de status do MySQL a seguir não são aplicáveis ao Aurora MySQL. Essa lista não é exaustiva.

- `innodb_buffer_pool_bytes_dirty`
- `innodb_buffer_pool_pages_dirty`
- `innodb_buffer_pool_pages_flushed`

O Aurora MySQL versão 3 remove as seguintes variáveis de status presentes no Aurora MySQL versão 2:

- `AuroraDb_lockmgr_bitmaps0_in_use`
- `AuroraDb_lockmgr_bitmaps1_in_use`
- `AuroraDb_lockmgr_bitmaps_mem_used`
- `AuroraDb_thread_deadlocks`
- `available_alter_table_log_entries`
- `Aurora_lockmgr_memory_used`

- Aurora_missing_history_on_replica_incidents
- Aurora_new_lock_manager_lock_release_cnt
- Aurora_new_lock_manager_lock_release_total_duration_micro
- Aurora_new_lock_manager_lock_timeout_cnt
- Aurora_total_op_memory
- Aurora_total_op_temp_space
- Aurora_used_alter_table_log_entries
- Aurora_using_new_lock_manager
- Aurora_volume_bytes_allocated
- Aurora_volume_bytes_left_extent
- Aurora_volume_bytes_left_total
- Com_alter_db_upgrade
- Compression
- External_threads_connected
- Innodb_available_undo_logs
- Last_query_cost
- Last_query_partial_plans
- Slave_heartbeat_period
- Slave_last_heartbeat
- Slave_received_heartbeats
- Slave_retried_transactions
- Slave_running
- Time_since_zero_connections

Estas variáveis de status do MySQL estão disponíveis no Aurora MySQL versão 2, mas não no Aurora MySQL versão 3:

- Innodb_redo_log_enabled
- Innodb_undo_tablespaces_total
- Innodb_undo_tablespaces_implicit

- `Innodb_undo_tablespaces_explicit`
- `Innodb_undo_tablespaces_active`

Eventos de espera do Aurora MySQL

Os seguintes são alguns eventos de espera comuns para o Aurora MySQL.

> **Note**
>
> Para ter informações sobre como ajustar a performance do Aurora MySQL usando eventos de espera, consulte Ajustar o Aurora MySQL com eventos de espera.
> Para obter informações sobre as convenções de nomenclatura usadas em eventos de espera do MySQL, consulte Performance Schema instrument naming conventions na documentação do MySQL.

cpu

O número de conexões ativas que estão prontas para execução é consistentemente maior que o número de vCPUs. Para obter mais informações, consulte cpu.

io/aurora_redo_log_flush

Uma sessão está mantendo dados no armazenamento do Aurora. Normalmente, esse evento de espera é para uma operação de E/S de gravação no Aurora MySQL. Para obter mais informações, consulte io/aurora_redo_log_flush.

io/aurora_respond_to_client

O processamento da consulta foi concluído e os resultados estão sendo retornados ao cliente da aplicação para as seguintes versões do Aurora MySQL: 2.10.2 e versões 2.10 posteriores, 2.09.3 e versões 2.09 posteriores, e 2.07.7 e versões 2.07 posteriores. Compare a largura de banda da rede da classe de instância de banco de dados com o tamanho do conjunto de resultados que está sendo retornado. Além disso, confira os tempos de resposta no lado do cliente. Se o cliente não responder e não conseguir processar os pacotes TCP, poderão ocorrer descartes de pacotes e retransmissões TCP. Essa situação afeta negativamente a largura de banda da rede. Nas versões inferiores às versões 2.10.2, 2.09.3 e 2.07.7, o evento de espera inclui incorretamente o tempo ocioso. Para aprender a ajustar seu banco de dados quando essa espera for proeminente, consulte io/aurora_respond_to_client.

io/file/csv/data

Threads estão gravando em tabelas no formato de valores separados por vírgula (CSV). Verifique o uso de sua tabela CSV. Uma causa típica desse evento é a definição de `log_output` em uma tabela.

io/file/sql/binlog

Um thread está aguardando um arquivo de log binário (binlog) que está sendo gravado em disco.

io/redo_log_flush

Uma sessão está mantendo dados no armazenamento do Aurora. Normalmente, esse evento de espera é para uma operação de E/S de gravação no Aurora MySQL. Para obter mais informações, consulte io/redo_log_flush.

io/socket/sql/client_connection

O programa `mysqld` está ocupado com a criação de threads para lidar com novas conexões de clientes de entrada. Para obter mais informações, consulte io/socket/sql/client_connection.

io/table/sql/handler

O mecanismo está aguardando acesso a uma tabela. Esse evento ocorre não importa se os dados estão armazenados em cache no grupo de buffer ou acessados no disco. Para obter mais informações, consulte io/table/sql/handler.

lock/table/sql/handler

Este evento de espera é um manipulador de eventos de espera de bloqueio de tabela. Para ter mais informações sobre eventos "átomo" e "molécula" no Performance Schema, consulte o tópico sobre Eventos "átomo" e "molécula" no Performance Schema, na documentação do MySQL.

synch/cond/innodb/row_lock_wait

Várias instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas de banco de dados simultaneamente. Para obter mais informações, consulte synch/cond/innodb/row_lock_wait.

synch/cond/innodb/row_lock_wait_cond

Várias instruções DML estão acessando as mesmas linhas de banco de dados simultaneamente. Para obter mais informações, consulte synch/cond/innodb/row_lock_wait_cond.

synch/cond/sql/MDL_context::COND_wait_status

Threads estão aguardando em um bloqueio de metadados de tabela. O mecanismo utiliza esse tipo de bloqueio para gerenciar o acesso simultâneo a um esquema de banco de dados e garantir a consistência dos dados. Para ter mais informações, consulte Optimizing locking operations na documentação do MySQL. Para aprender a ajustar seu banco de dados quando esse evento for proeminente, consulte synch/cond/sql/MDL_context::COND_wait_status.

synch/cond/sql/MYSQL_BIN_LOG::COND_done

Você habilitou o registro em log binário. Pode haver uma alta taxa de transferência de confirmações, alto número de transações fazendo confirmações ou réplicas lendo binlogs. Considere utilizar instruções em várias linhas ou instruções de empacotamento em uma única transação. No Aurora, use bancos de dados globais em vez da replicação de logs binários ou use os parâmetros `aurora_binlog_*`.

synch/mutex/innodb/aurora_lock_thread_slot_futex

Várias instruções DML estão acessando as mesmas linhas de banco de dados simultaneamente. Para obter mais informações, consulte synch/mutex/innodb/aurora_lock_thread_slot_futex.

synch/mutex/innodb/buf_pool_mutex

O grupo de buffer não é suficientemente grande para armazenar o conjunto de dados de trabalho. Ou a workload acessa páginas de uma tabela específica, resultando na contenção no grupo de buffer. Para obter mais informações, consulte synch/mutex/innodb/buf_pool_mutex.

synch/mutex/innodb/fil_system_mutex

O processo está esperando ter acesso ao cache de memória do espaço de tabelas. Para obter mais informações, consulte synch/mutex/innodb/fil_system_mutex.

synch/mutex/innodb/trx_sys_mutex

Operações estão verificando, atualizando, excluindo ou adicionando IDs de transações no InnoDB de maneira consistente ou controlada. Essas operações exigem uma chamada mutex `trx_sys` que é rastreada pela instrumentação do Performance Schema. Elas incluem o gerenciamento do sistema de transações quando o banco de dados é iniciado ou encerrado, reversões, limpezas de ações desfazer, acesso para leitura de linhas e cargas de grupos de buffer. A alta carga do banco de dados com um alto número de transações resulta no surgimento frequente desse evento de espera. Para obter mais informações, consulte synch/mutex/innodb/trx_sys_mutex.

synch/mutex/mysys/KEY_CACHE::cache_lock

O mutex `keycache->cache_lock` controla o acesso ao cache de chaves para tabelas MyISAM. Embora o Aurora MySQL não permita o uso de tabelas MyISAM para armazenar dados persistentes, elas são usadas para armazenar tabelas temporárias internas. Considere conferir os contadores de status `created_tmp_disk_tables` ou `created_tmp_tables`, porque em determinadas situações, as tabelas temporárias são gravadas no disco quando não cabem mais na memória.

synch/mutex/sql/FILE_AS_TABLE::LOCK_offsets

O mecanismo obtém esse mutex quando abre ou cria um arquivo de metadados de tabela. Quando esse evento de espera ocorre excessivamente, significa que o número de tabelas criadas ou abertas aumentou.

synch/mutex/sql/FILE_AS_TABLE::LOCK_shim_lists

O mecanismo obtém esse mutex ao executar operações como `reset_size`,`detach_contents` ou `add_contents` na estrutura interna que controla tabelas abertas. O mutex sincroniza o acesso ao conteúdo das listas. Quando esse evento de espera ocorre com alta frequência, indica uma súbita mudança no conjunto de tabelas anteriormente acessadas. O mecanismo precisa acessar novas tabelas ou descartar o contexto relacionado a essas tabelas.

synch/mutex/sql/LOCK_open

O número de tabelas que estão sendo abertas pelas suas sessões excede o tamanho do cache de definição de tabelas ou do cache de abertura de tabelas. Aumente o tamanho desses caches. Para ter mais informações, consulte Como o MySQL abre e fecha tabelas.

synch/mutex/sql/LOCK_table_cache

O número de tabelas que estão sendo abertas pelas suas sessões excede o tamanho do cache de definição de tabelas ou do cache de abertura de tabelas. Aumente o tamanho desses caches. Para ter mais informações, consulte Como o MySQL abre e fecha tabelas.

synch/mutex/sql/LOG

Neste evento de espera, há threads aguardando um bloqueio de log. Por exemplo, um thread pode aguardar um bloqueio para gravar no arquivo de log de consultas lentas.

synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit

Neste evento de espera, há um thread aguardando para adquirir um bloqueio com a intenção de confirmar no log binário. A contenção do log binário pode ocorrer em bancos de dados com uma taxa de alteração muito alta. Dependendo da versão do MySQL, há alguns bloqueios usados

para proteger a consistência e durabilidade do log binário. No RDS para MySQL, os logs binários são usados para a replicação e para o processo de backup automatizado. No Aurora MySQL, os logs binários não são necessários para a replicação ou backups nativos. Por padrão, eles estão desabilitados, mas podem ser habilitados e usados para replicação externa e captura de dados de alterações. Para ter mais informações, consulte The binary log na documentação do MySQL.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_dump_thread_metrics_collection

Se o registro em log binário estiver habilitado, o mecanismo obterá esse mutex ao imprimir métricas de threads de despejo ativo no log de erros do mecanismo e no mapa de operações interno.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_inactive_binlogs_map

Se o registro em log binário estiver habilitado, o mecanismo obterá esse mutex ao adicionar, excluir ou pesquisar na lista de arquivos binlog atrás do mais recente.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_io_cache

Se o registro em log binário estiver habilitado, o mecanismo obterá esse mutex durante operações de cache de E/S para binlog do Aurora: alocar, redimensionar, liberar, gravar, ler, limpar e acessar informações do cache. Se esse evento ocorrer com frequência, significa que o mecanismo está acessando o cache em que os eventos de log binário são armazenados. Para reduzir tempos de espera, reduza confirmações. Tente agrupar várias instruções em uma só transação.

synch/mutex/sql/MYSQL_BIN_LOG::LOCK_log

Você habilitou o registro em log binário. Pode haver uma alta taxa de transferência de confirmações, muitas transações fazendo confirmações ou réplicas lendo binlogs. Considere utilizar instruções em várias linhas ou instruções de empacotamento em uma única transação. No Aurora, use bancos de dados globais em vez da replicação de logs binários ou use os parâmetros `aurora_binlog_*`.

synch/mutex/sql/SERVER_THREAD::LOCK_sync

O mutex `SERVER_THREAD::LOCK_sync` é obtido durante o agendamento, o processamento ou a inicialização de threads para gravações em arquivos. A ocorrência excessiva desse evento de espera indica aumento na atividade de gravação no banco de dados.

synch/mutex/sql/TABLESPACES:lock

O mecanismo obtém o mutex `TABLESPACES:lock` durante as operações de espaço de tabela a seguir: criar, excluir, truncar e estender. A ocorrência excessiva desse evento de espera indica

alta frequência de operações de espaço de tabela. Um exemplo é carregar muitos dados no banco de dados.

synch/rwlock/innodb/dict

Neste evento de espera, há threads aguardando um rwlock mantido no dicionário de dados InnoDB.

synch/rwlock/innodb/dict_operation_lock

Neste evento de espera, há threads aguardando bloqueios nas operações do dicionário de dados InnoDB.

synch/rwlock/innodb/dict sys RW lock

É acionado ao mesmo tempo um alto número de instruções de linguagem de controle de dados (DCLs) simultâneas no código de linguagem de definição de dados (DDLs). Reduza a dependência da aplicação por DDLs durante suas atividades regulares.

synch/rwlock/innodb/index_tree_rw_lock

Muitas instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas de objetos de banco de dados simultaneamente. Tente utilizar instruções com várias linhas. Além disso, disperse a workload por objetos de banco de dados diferentes. Por exemplo, implemente o particionamento.

synch/sxlock/innodb/dict_operation_lock

É acionado ao mesmo tempo um alto número de instruções de linguagem de controle de dados (DCLs) simultâneas no código de linguagem de definição de dados (DDLs). Reduza a dependência da aplicação por DDLs durante suas atividades regulares.

synch/sxlock/innodb/dict_sys_lock

É acionado ao mesmo tempo um alto número de instruções de linguagem de controle de dados (DCLs) simultâneas no código de linguagem de definição de dados (DDLs). Reduza a dependência da aplicação por DDLs durante suas atividades regulares.

synch/sxlock/innodb/hash_table_locks

A sessão não foi capaz de encontrar páginas no grupo de buffer. O mecanismo precisa ler um arquivo ou modificar a lista com utilização menos recente (LRU) para o grupo de buffer. Considere aumentar o tamanho do cache do buffer e melhorar caminhos de acesso para consultas relevantes.

synch/sxlock/innodb/index_tree_rw_lock

Muitas instruções de linguagem de manipulação de dados (DML) estão acessando as mesmas linhas de objetos de banco de dados simultaneamente. Tente utilizar instruções com várias linhas. Além disso, disperse a workload por objetos de banco de dados diferentes. Por exemplo, implemente o particionamento.

synch/mutex/innodb/temp_pool_manager_mutex

Esse evento de espera ocorre quando uma sessão está aguardando para adquirir um mutex para o grupo de tablespaces temporários de sessão.

Para ter mais informações sobre como solucionar problemas de eventos de espera de sincronização, consulte Why is my MySQL DB instance showing a high number of active sessions waiting on SYNCH wait events in Performance Insights? (Por que minha instância de banco de dados do MySQL está mostrando um grande número de sessões ativas aguardando eventos de espera SYNCH no Performance Insights?).

Estados de threads do Aurora MySQL

Os seguintes são alguns estados de thread comuns do Aurora MySQL.

checking permissions

O thread está verificando se o servidor tem os privilégios necessários para executar a instrução.

checking query cache for query

O servidor está verificando se a consulta atual está presente no cache de consulta.

cleaned up

Este é o estado final de uma conexão cujo trabalho está concluído, mas que não foi encerrada pelo cliente. A melhor solução é encerrar a conexão explicitamente no código. Outra alternativa é definir um valor mais baixo para `wait_timeout` no grupo de parâmetros.

closing tables

O thread está liberando os dados de tabelas alterados no disco e encerrando as tabelas utilizadas. Se esta não for uma operação rápida, verifique as métricas de consumo de largura de banda de rede em relação à largura de banda de rede da classe de instância. Verifique

também se os valores dos parâmetros para `table_open_cache` e `table_definition_cache` permitem que tabelas suficientes sejam abertas ao mesmo tempo, para que o mecanismo não precise abrir e fechar tabelas com frequência. Esses parâmetros influenciam o consumo de memória na instância.

converting HEAP to MyISAM

A consulta está convertendo uma tabela temporária de "na memória" para "no disco". A conversão é necessária porque as tabelas temporárias criadas pelo MySQL nas etapas intermediárias do processamento de consultas se tornaram grandes demais para a memória. Verifique os valores de `tmp_table_size` e `max_heap_table_size`. Em versões posteriores, o nome desse estado de thread é `converting HEAP to ondisk`.

converting HEAP to ondisk

O thread está convertendo uma tabela temporária interna de uma tabela "na memória" para uma tabela "no disco".

copy to tmp table

O thread está processando uma instrução `ALTER TABLE`. Esse estado ocorre após a criação da tabela com a nova estrutura, mas antes que as linhas sejam copiadas para ela. Para um thread nesse estado, é possível utilizar o Performance Schema para obter informações sobre o progresso da operação de cópia.

creating sort index

O Aurora MySQL está fazendo uma classificação porque não consegue utilizar um índice existente para satisfazer a cláusula `ORDER BY` ou `GROUP BY` de uma consulta. Para obter mais informações, consulte criar índice de classificação.

creating table

O thread está criando uma tabela permanente ou temporária.

delayed commit ok done

Uma confirmação assíncrona no Aurora MySQL recebeu um reconhecimento e está concluída.

delayed commit ok initiated

O thread do Aurora MySQL iniciou o processo de confirmação assíncrona, mas está aguardando reconhecimento. Em geral, esse é o tempo de confirmação autêntico de uma transação.

delayed send ok done

Um thread de operador do Aurora MySQL que está vinculado a uma conexão pode ser liberado enquanto uma resposta é enviada ao cliente. O thread pode iniciar outro trabalho. O estado `delayed send ok` indica que o reconhecimento assíncrono para o cliente foi concluído.

delayed send ok initiated

Um thread de operador do Aurora MySQL enviou uma resposta assíncrona a um cliente e está livre para trabalhar com outras conexões. A transação iniciou um processo de confirmação assíncrona que ainda não foi reconhecido.

executing

O thread iniciou a execução de uma instrução.

freeing items

O thread executou um comando. Algumas liberações de itens realizadas nesse estado envolvem o cache de consulta. Em geral, esse estado é seguido por uma limpeza.

init

Esse estado ocorre antes da inicialização de instruções `ALTER TABLE`,`DELETE`,`INSERT`,`SELECT` ou `UPDATE`. As ações nesse estado incluem liberar o log binário ou o log do InnoDB e a limpeza do cache de consulta.

A origem enviou todos os logs binários à réplica; aguardando mais atualizações.

O nó primário terminou sua parte da replicação. O thread está aguardando mais consultas serem executadas para poder gravar no log binário (binlog).

opening tables

O thread está tentando abrir uma tabela. Essa operação é rápida, a menos que uma instrução `ALTER TABLE` ou `LOCK TABLE` precise ser encerrada ou exceda o valor de `table_open_cache`.

optimizing

O servidor está fazendo otimizações iniciais para uma consulta.

preparing

Esse estado ocorre durante a otimização de consultas.

query end

Esse estado ocorre após o processamento de uma consulta, mas antes do estado de liberação de itens.

removing duplicates

O Aurora MySQL não conseguiu otimizar uma operação `DISTINCT` no estágio inicial de uma consulta. O Aurora MySQL precisa remover todas as linhas duplicadas antes de enviar o resultado ao cliente.

searching rows for update

O thread está localizando todas as linhas correspondentes antes de as atualizar. Este estágio será necessário se o `UPDATE` estiver alterando o índice usado pelo mecanismo para localizar as linhas.

sending binlog event to slave

O thread fez a leitura de um evento do log binário e o está enviando para a réplica.

sending cached result to client

O servidor está obtendo o resultado de uma consulta do cache de consultas e o enviando ao cliente.

sending data

O thread está fazendo a leitura e o processamento de linhas para uma instrução `SELECT`, mas ainda não começou a enviar dados ao cliente. O processo está identificando quais páginas contêm os resultados necessários para atender à consulta. Para obter mais informações, consulte enviar dados.

sending to client

O servidor está gravando um pacote para o cliente. Em versões anteriores do MySQL, esse evento de espera se chamava `writing to net`.

starting

Este é o primeiro estágio no início da execução da instrução.

estatísticas

O servidor está calculando estatísticas para desenvolver um plano de execução de consultas. Se um thread estiver nesse estado por um longo tempo, o servidor provavelmente está associado ao disco durante a realização de outros trabalhos.

storing result in query cache

O servidor está armazenando o resultado de uma consulta no cache de consultas.

system lock

O thread chamou `mysql_lock_tables`, mas o estado desse thread não foi atualizado desde a chamada. Esse estado geral ocorre por diversos motivos.

atualizar

O thread está se preparando para iniciar a atualização da tabela.

atualizar

O thread está procurando linhas e as está atualizando.

user lock

O thread emitiu uma chamada `GET_LOCK`. O thread solicitou um bloqueio consultivo e está aguardando por ele ou está planejando solicitá-lo.

waiting for more updates

O nó primário terminou sua parte da replicação. O thread está aguardando mais consultas serem executadas para poder gravar no log binário (binlog).

waiting for schema metadata lock

Uma espera por um bloqueio de metadados.

waiting for stored function metadata lock

Uma espera por um bloqueio de metadados.

waiting for stored procedure metadata lock

Uma espera por um bloqueio de metadados.

waiting for table flush

O thread está executando `FLUSH TABLES` e aguardando todos os threads fecharem suas tabelas. Ou, o thread recebeu uma notificação de que a estrutura subjacente de uma tabela foi modificada e, portanto, precisa reabrir essa tabela para obter a nova estrutura. Para reabrir a tabela, o thread deve aguardar até que todos os outros threads s tenham fechado. Essa notificação ocorre quando outro thread utilizou uma das seguintes instruções na tabela: `FLUSH TABLES`, `ALTER TABLE`, `RENAME TABLE`, `REPAIR TABLE`, `ANALYZE TABLE` ou `OPTIMIZE TABLE`.

waiting for table level lock

Uma sessão está mantendo um bloqueio em uma tabela enquanto outra sessão tenta obter esse mesmo bloqueio na mesma tabela.

waiting for table metadata lock

O Aurora MySQL utiliza o bloqueio de metadados para gerenciar o acesso simultâneo a objetos de banco de dados e garantir a consistência dos dados. Nesse evento de espera, uma sessão está mantendo um bloqueio de metadados em uma tabela enquanto outra sessão tenta obter esse mesmo bloqueio na mesma tabela. Quando o Performance Schema está habilitado, esse estado de thread é indicado como o evento de espera `synch/cond/sql/MDL_context::COND_wait_status`.

writing to net

O servidor está gravando um pacote na rede. Em versões posteriores do MySQL, esse evento de espera se chama `Sending to client`.

Níveis de isolamento do Aurora MySQL

Descubra como as instâncias de banco de dados de um cluster do Aurora MySQL implementam a propriedade de isolamento do banco de dados. Este tópico explica como o comportamento padrão do Aurora MySQL se equilibra entre consistência rigorosa e alta performance. Você pode usar essas informações para decidir quando alterar as configurações padrão com base nas características da sua workload.

Níveis de isolamento disponíveis para instâncias do gravador

Você pode usar os níveis de isolamento `REPEATABLE READ`, `READ COMMITTED`, `READ UNCOMMITTED` e `SERIALIZABLE` na instância primária de um cluster de banco de dados do Aurora MySQL. Esses níveis de isolamento funcionam no Aurora MySQL da mesma maneira que no RDS para MySQL.

Nível de isolamento REPEATABLE READ (leitura repetível) para instâncias leitoras

Por padrão, as instâncias de banco de dados do Aurora MySQL configuradas como réplicas somente leitura do Aurora sempre usam o nível de isolamento `REPEATABLE READ`. Essas instâncias de banco de dados ignoram quaisquer instruções `SET TRANSACTION ISOLATION LEVEL` e continuam usando o nível de isolamento `REPEATABLE READ`.

Nível de isolamento READ COMMITTED (leitura confirmada) para instâncias leitoras

Se sua aplicação incluir uma workload com alta demanda de gravação na instância primária e consultas demoradas nas réplicas do Aurora, você poderá enfrentar um atraso significativo na limpeza. Atraso na limpeza acontece quando a coleta de lixo interna é bloqueada por consultas demoradas. O sintoma visto é um valor alto para `history list length` na saída do comando `SHOW ENGINE INNODB STATUS`. Você pode monitorar esse valor usando a métrica `RollbackSegmentHistoryListLength` no CloudWatch. O atraso substancial na limpeza pode reduzir a eficácia dos índices secundários, reduzir a performance geral da consulta e gerar desperdício de espaço de armazenamento.

Se enfrentar esses problemas, você poderá definir uma configuração no nível da sessão do Aurora MySQL, `aurora_read_replica_read_committed`, para usar o nível de isolamento `READ COMMITTED` nas réplicas do Aurora. Ao aplicar essa configuração, você pode ajudar a reduzir a desaceleração e o desperdício de espaço resultantes da execução de consultas demoradas ao mesmo tempo em que as transações que modificam suas tabelas.

Recomendamos que você entenda o comportamento específico do Aurora MySQL de isolamento de `READ COMMITTED` antes de usar essa configuração. O comportamento de `READ COMMITTED` da réplica do Aurora está em conformidade com o padrão ANSI SQL. No entanto, o isolamento é menos rigoroso do que o comportamento típico de `READ COMMITTED` do MySQL com o qual você pode estar familiarizado. Portanto, os resultados da consulta com `READ COMMITTED` em uma réplica de leitura do Aurora MySQL poderão ser diferentes dos resultados da mesma consulta com `READ COMMITTED` na instância primária do Aurora MySQL ou no RDS para MySQL. Você pode usar a configuração `aurora_read_replica_read_committed` para esses casos como um relatório abrangente que verifica um banco de dados muito grande. Por outro lado, você pode evitá-la para consultas breves com pequenos conjuntos de resultados, em que a precisão e a repetibilidade são importantes.

O nível de isolamento `READ COMMITTED` não está disponível para sessões em um cluster secundário em um banco de dados global Aurora que usam o recurso de encaminhamento de gravação. Para obter informações sobre o encaminhamento de gravação, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Usar READ COMMITTED para leitores

Para usar o nível de isolamento `READ COMMITTED` para réplicas do Aurora, defina a configuração `aurora_read_replica_read_committed` como `ON`. Use essa configuração no nível da sessão

enquanto estiver conectada a uma réplica específica do Aurora. Para fazer isso, execute os comandos SQL a seguir:

```
set session aurora_read_replica_read_committed = ON;
set session transaction isolation level read committed;
```

Você pode usar essa configuração temporariamente para executar consultas únicas interativas. Você também pode executar uma aplicação de relatórios ou análise de dados que se beneficie do nível de isolamento `READ COMMITTED`, mantendo a configuração padrão inalterada para outras aplicações.

Quando a configuração `aurora_read_replica_read_committed` estiver ativada, use o comando `SET TRANSACTION ISOLATION LEVEL` para especificar o nível de isolamento das transações apropriadas.

```
set transaction isolation level read committed;
```

Diferenças no comportamento READ COMMITTED (LEITURA CONFIRMADA) das réplicas do Aurora

A configuração `aurora_read_replica_read_committed` disponibiliza o nível de isolamento `READ COMMITTED` para uma Réplica do Aurora, com comportamento de consistência otimizado para transações de longa execução. O nível de isolamento `READ COMMITTED` nas réplicas do Aurora é menos rigoroso do que nas instâncias primárias do Aurora. Por esse motivo, habilite essa configuração apenas nas Réplicas do Aurora, onde você sabe que suas consultas podem aceitar a possibilidade de certos tipos de resultados inconsistentes.

Suas consultas podem enfrentar certos tipos de anomalias de leitura quando a configuração `aurora_read_replica_read_committed` está habilitada. Dois tipos de anomalias são especialmente importantes para entender e lidar com o código da aplicação. Uma non-repeatable read (leitura não repetível) ocorre quando outra transação é confirmada enquanto sua consulta está em execução. Uma consulta demorada pode ter dados diferentes no início e no final da consulta. Uma phantom read (leitura fantasma) ocorre quando outras transações fazem com que as linhas existentes sejam reorganizadas enquanto a consulta está em execução, e uma ou mais linhas são lidas duas vezes pela consulta.

Suas consultas podem ter contagens de linhas inconsistentes como resultado de leituras fantasmas. Suas consultas também podem retornar resultados incompletos ou inconsistentes devido a leituras não repetíveis. Por exemplo, vamos supor que uma operação de junção se refira a tabelas

modificadas simultaneamente por instruções SQL como `INSERT` ou `DELETE`. Nesse caso, a consulta de junção poderá ler uma linha de uma tabela, mas não a linha correspondente de outra tabela.

O padrão SQL ANSI permite esses comportamentos para o nível de isolamento `READ COMMITTED`. No entanto, esses comportamentos são diferentes da implementação típica do MySQL no `READ COMMITTED`. Portanto, antes de habilitar a configuração `aurora_read_replica_read_committed`, verifique qualquer código SQL existente para saber se ele opera conforme o esperado no modelo de consistência mais flexível.

A contagem de linhas e outros resultados podem não ser altamente consistentes no nível de isolamento `READ COMMITTED` enquanto essa configuração estiver habilitada. Dessa forma, geralmente habilita-se a configuração apenas ao executar consultas analíticas que agregam grandes quantidades de dados e não exigem precisão absoluta. Se você não tiver esse tipo de consultas demoradas com uma workload de gravação intensa, provavelmente não precisará da configuração `aurora_read_replica_read_committed`. Sem a combinação de consultas demoradas e uma workload de gravação intensa, é improvável que você enfrente problemas com o comprimento da lista de histórico.

Example Consultas que mostram comportamento de isolamento para READ COMMITTED em réplicas do Aurora

O exemplo a seguir mostra como as consultas `READ COMMITTED` em uma Réplica do Aurora podem retornar resultados não repetíveis se as transações modificarem as tabelas associadas simultaneamente. A tabela `BIG_TABLE` contém 1 milhão de linhas antes do início de qualquer consulta. Outras instruções de linguagem de manipulação de dados (DML) adicionam, removem ou alteram linhas enquanto estão em execução.

As consultas na instância primária do Aurora no nível de isolamento `READ COMMITTED` produzem resultados previsíveis. No entanto, os custos indiretos de manter a visualização de leitura consistente durante toda a vida útil de todas as consultas demoradas podem levar a uma coleta de lixo cara posteriormente.

As consultas na Réplica do Aurora no nível de isolamento `READ COMMITTED` são otimizadas para minimizar esses custos indiretos da coleta de lixo. A desvantagem é que os resultados podem variar dependendo do fato de as consultas recuperarem linhas adicionadas, removidas ou reorganizadas por transações confirmadas enquanto a consulta está em execução. As consultas não precisam, mas têm permissão para considerar essas linhas. Para fins de demonstração, as consultas verificam apenas o número de linhas na tabela usando a função `COUNT(*)`.

Tempo	Instrução DML na instância primária do Aurora	Consulta na instância primária do Aurora com READ COMMITTED (LEITURA CONFIRMADA)	Consulta na réplica do Aurora com READ COMMITTED
T1	`INSERT INTO big_table SELECT * FROM other_table LIMIT 1000000; COMMIT;`		
T2		Q1: `SELECT COUNT(*) FROM big_table;`	Q2: `SELECT COUNT(*) FROM big_table;`
T3	`INSERT INTO big_table (c1, c2) VALUES (1, 'one more row'); COMMIT;`		
T4		Se Q1 terminar agora, o resultado será 1.000.000.	Se Q2 terminar agora, o resultado será 1.000.000 ou 1.000.001.
T5	`DELETE FROM big_table LIMIT 2; COMMIT;`		
T6		Se Q1 terminar agora, o resultado será 1.000.000.	Se Q2 terminar agora, o resultado será 1.000.000 ou 1.000.001 ou 999.999 ou 999.998.

Tempo	Instrução DML na instância primária do Aurora	Consulta na instância primária do Aurora com READ COMMITTED (LEITURA CONFIRMADA)	Consulta na réplica do Aurora com READ COMMITTED
T7	`UPDATE big_table SET c2 = CONCAT(c2 ,c2,c2); COMMIT;`		
T8		Se Q1 terminar agora, o resultado será 1.000.000.	Se Q2 terminar agora, o resultado será 1.000.000 ou 1.000.001 ou 999.999 ou, possivelmente, um número superior.
T9		Q3: `SELECT COUNT(*) FROM big_table;`	Q4: `SELECT COUNT(*) FROM big_table;`
T10		Se Q3 terminar agora, o resultado será 999.999.	Se Q4 terminar agora, o resultado será 999.999.
T11		Q5: `SELECT COUNT(*) FROM parent_table p JOIN child_tab le c ON (p.id = c.id) WHERE p.id = 1000;`	Q6: `SELECT COUNT(*) FROM parent_table p JOIN child_tab le c ON (p.id = c.id) WHERE p.id = 1000;`

Tempo	Instrução DML na instância primária do Aurora	Consulta na instância primária do Aurora com READ COMMITTED (LEITURA CONFIRMADA)	Consulta na réplica do Aurora com READ COMMITTED
T12	`INSERT INTO parent_table (id, s) VALUES (1000, 'hello'); INSERT INTO child_table (id, s) VALUES (1000, 'world'); COMMIT;`		
T13		Se Q5 terminar agora, o resultado será 0.	Se Q6 terminar agora, o resultado será 0 ou 1.

Se as consultas forem concluídas rapidamente, antes de qualquer outra transação executar instruções DML e ser confirmada, os resultados serão previsíveis e iguais entre a instância principal e a Réplica do Aurora. Vamos examinar as diferenças de comportamento em detalhes, começando pela primeira consulta.

Os resultados para Q1 são altamente previsíveis, porque `READ COMMITTED` na instância primária usa um modelo de consistência sólido semelhante ao nível de isolamento `REPEATABLE READ`.

Os resultados para Q2 podem variar dependendo de quais transações são confirmadas enquanto a consulta está em execução. Por exemplo, vamos supor que outras transações executem instruções DML e sejam confirmadas enquanto as consultas estiverem em execução. Nesse caso, a consulta na Réplica do Aurora com o nível de isolamento `READ COMMITTED` poderá ou não levar em consideração as alterações. As contagens de linhas não são previsíveis da mesma maneira que no nível de isolamento `REPEATABLE READ`. Elas também não são tão previsíveis quanto as consultas executadas no nível de isolamento `READ COMMITTED` na instância primária ou em uma instância do RDS para MySQL.

A instrução `UPDATE` em T7 não altera realmente o número de linhas na tabela. No entanto, ao alterar o comprimento de uma coluna de tamanho variável, essa instrução pode fazer com que as linhas sejam reorganizadas internamente. Uma transação `READ COMMITTED` demorada pode visualizar a versão antiga de uma linha e, posteriormente, na mesma consulta, visualizar a nova versão da mesma linha. A consulta também pode ignorar as versões antiga e nova da linha, por isso a contagem de linhas pode ser diferente da esperada.

Os resultados de Q5 e Q6 podem ser idênticos ou ligeiramente diferentes. A consulta Q6 na Réplica do Aurora em `READ COMMITTED` pode, mas não precisa, visualizar as novas linhas confirmadas enquanto a consulta estiver em execução. Ela também pode visualizar a linha de uma tabela, mas não de outra. Se a consulta de junção não encontrar uma linha correspondente nas duas tabelas, ela retornará uma contagem igual a zero. Se a consulta localizar as duas novas linhas em `PARENT_TABLE` e `CHILD_TABLE`, ela retornará uma contagem igual a um. Em uma consulta demorada, as pesquisas nas tabelas unidas podem ocorrer em momentos amplamente distintos.

> **Note**
>
> Essas diferenças de comportamento dependem do momento em que as transações são confirmadas e quando as consultas processam as linhas da tabela subjacente. Dessa forma, é mais provável que você visualize essas diferenças nas consultas de relatório que demoram minutos ou horas e são executadas em clusters do Aurora que processam transações OLTP ao mesmo tempo. Esses são os tipos de cargas de trabalho mistas que mais se beneficiam do nível de isolamento `READ COMMITTED` nas Réplicas do Aurora.

Dicas do Aurora MySQL

É possível usar dicas SQL com consultas do Aurora MySQL para ajustar a performance. Também é possível usar dicas para impedir que planos de execução para consultas importantes sejam alterados devido a condições imprevisíveis.

> **Tip**
>
> Para verificar o efeito que uma dica tem em uma consulta, examine o plano de consulta produzido pela instrução `EXPLAIN`. Compare os planos de consulta com e sem a dica.

No Aurora MySQL versão 3, é possível usar todas as dicas disponíveis no MySQL Community Edition 8.0. Para ter mais informações sobre essas dicas, consulte Dicas do Optimizer no Guia de referência do MySQL.

As dicas a seguir estão disponíveis no Aurora MySQL versão 2. Elas se aplicam a consultas que usam o recurso de junção de hash no Aurora MySQL versão 2, especialmente a consultas que usam a otimização de consultas paralelas.

PQ, NO_PQ

Especifica se deve forçar o Optimizer a usar a consulta paralela por tabela ou por consulta.

`PQ` força o Optimizer a usar a consulta paralela para tabelas especificadas ou para toda a consulta (bloco). `NO_PQ` impede que o Optimizer use a consulta paralela para tabelas especificadas ou para toda a consulta (bloco).

Esta dica está disponível no Aurora MySQL versão 2.11 e posterior. Os exemplos a seguir mostram como usar essa dica.

> **Note**
>
> A especificação de um nome de tabela força o Optimizer a aplicar a dica `PQ/NO_PQ` somente às tabelas selecionadas. Não especificar um nome de tabela força a dica `PQ/NO_PQ` em todas as tabelas afetadas pelo bloco de consulta.

```
EXPLAIN SELECT /*+ PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;

EXPLAIN SELECT /*+ NO_PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;
```

```
EXPLAIN SELECT /*+ NO_PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;
```

HASH_JOIN, NO_HASH_JOIN

Ativa ou desativa a capacidade do Optimizer de consulta paralela de escolher se deseja usar o método de otimização de junção de hash para uma consulta. `HASH_JOIN` permite que o Optimizer use junção de hash se esse mecanismo for mais eficiente. `NO_HASH_JOIN` impede que o Optimizer use junção de hash para a consulta. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior. Ela não tem efeito no Aurora MySQL versão 3.

Os exemplos a seguir mostram como usar essa dica.

```
EXPLAIN SELECT/*+ HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ NO_HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_PROBING, NO_HASH_JOIN_PROBING

Em uma consulta de junção de hash, especifica se deve ou não usar a tabela especificada para o lado de teste da junção. A consulta testa se os valores de coluna da tabela de compilação existem na tabela de teste, em vez de ler todo o conteúdo da tabela de teste. É possível usar `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` para especificar como as consultas de junção hash são processadas sem reordenar as tabelas dentro do texto da consulta. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior. Ela não tem efeito no Aurora MySQL versão 3.

Os exemplos a seguir mostram como usar essa dica. Especificar a dica `HASH_JOIN_PROBING` para a tabela `T2` tem o mesmo efeito que especificar `NO_HASH_JOIN_PROBING` para a tabela `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_PROBING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_PROBING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_BUILDING, NO_HASH_JOIN_BUILDING

Em uma consulta de junção de hash, especifica se deve ou não usar a tabela especificada para o lado de compilação da junção. A consulta processa todas as linhas desta tabela para criar a lista de valores de coluna para referência cruzada com a outra tabela. É possível usar `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` para especificar como as consultas de junção hash são processadas sem reordenar as tabelas dentro do texto da consulta. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior. Ela não tem efeito no Aurora MySQL versão 3.

Os exemplos a seguir mostram como usar essa dica. Especificar a dica `HASH_JOIN_BUILDING` para a tabela `T2` tem o mesmo efeito que especificar `NO_HASH_JOIN_BUILDING` para a tabela `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_BUILDING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_BUILDING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

JOIN_FIXED_ORDER

Especifica que as tabelas na consulta são unidas com base na ordem em que são listadas na consulta. É útil com consultas que envolvem três ou mais tabelas. Deve ser um substituto da dica `STRAIGHT_JOIN` do MySQL e é equivalente à dica JOIN_FIXED_ORDER do MySQL. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior.

Os exemplos a seguir mostram como usar essa dica.

```
EXPLAIN SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_ORDER

Especifica a ordem de junção para as tabelas na consulta. É útil com consultas que envolvem três ou mais tabelas. É equivalente à dica JOIN_ORDER do MySQL. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior.

Os exemplos a seguir mostram como usar essa dica.

```
EXPLAIN SELECT /*+ JOIN_ORDER (t4, t2, t1, t3) */ f1, f2
```

```
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_PREFIX

Especifica as tabelas a serem colocadas em primeiro lugar na ordem de junção. É útil com consultas que envolvem três ou mais tabelas. É equivalente à dica JOIN_PREFIX do MySQL. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior.

Os exemplos a seguir mostram como usar essa dica.

```
EXPLAIN SELECT /*+ JOIN_PREFIX (t4, t2) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_SUFFIX

Especifica as tabelas a serem colocadas por último na ordem de junção. É útil com consultas que envolvem três ou mais tabelas. É equivalente à dica JOIN_SUFFIX do MySQL. Esta dica está disponível no Aurora MySQL versão 2.08 e posterior.

Os exemplos a seguir mostram como usar essa dica.

```
EXPLAIN SELECT /*+ JOIN_SUFFIX (t1) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

Para obter informações sobre como usar consultas de junção de hash, consulte Otimizando grandes consultas de junção do Aurora MySQL com junções hash.

Referência de procedimentos armazenados do Aurora MySQL

Você pode gerenciar seu cluster de bancos de dados do Aurora MySQL chamando procedimentos armazenados integrados.

Tópicos

- Coletar e manter o histórico de status global
- Configurar, iniciar e interromper a replicação de logs binários (binlogs)
- Encerrar uma sessão ou consulta
- Replicar transações com GTIDs
- Alternar os logs de consulta

- Definir e mostrar a configuração de logs binários

Coletar e manter o histórico de status global

O Amazon RDS fornece um conjunto de procedimentos que gera snapshots dos valores dessas variáveis de status ao longo do tempo e as grava em uma tabela, juntamente com quaisquer alterações feitas desde o último snapshot. Essa infraestrutura é chamada de histórico de status global. Para obter mais informações, consulte Gerenciar o histórico de status global.

Os procedimentos armazenados a seguir gerenciam a forma como o histórico de status global é coletado e mantido.

Tópicos

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Gera um snapshot sob demanda para o histórico de status global.

Sintaxe

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Desativa os snapshots gerados pelo histórico de status global.

Sintaxe

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

Desativa a rotação da tabela `mysql.global_status_history`.

Sintaxe

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Ativa o histórico de status global para gerar snapshots padrão em intervalos especificados por `rds_set_gsh_collector`.

Sintaxe

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

Ativa a rotação do conteúdo da tabela `mysql.global_status_history` com o da `mysql.global_status_history_old` em intervalos especificados por `rds_set_gsh_rotation`.

Sintaxe

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Reveza o conteúdo da tabela `mysql.global_status_history` para o da `mysql.global_status_history_old` sob demanda.

Sintaxe

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Especifica o intervalo em minutos entre os snapshots gerados pelo histórico de status global.

Sintaxe

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parâmetros

intervalPeriod

O intervalo em minutos entre snapshots. O valor padrão é 5.

mysql.rds_set_gsh_rotation

Especifica o intervalo em dias entre os revezamentos da tabela `mysql.global_status_history`.

Sintaxe

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parâmetros

intervalPeriod

O intervalo em dias entre os revezamentos da tabela. O valor padrão é 7.

Configurar, iniciar e interromper a replicação de logs binários (binlogs)

É possível os seguintes procedimentos armazenados enquanto estiver conectado à instância primária em um cluster do Aurora MySQL. Esses procedimentos controlam como as transações são replicadas de um banco de dados externo para o Aurora MySQL ou do Aurora MySQL para um banco de dados externo.

Tópicos

- mysql.rds_disable_session_binlog (Aurora MySQL versão 2)
- mysql.rds_enable_session_binlog (Aurora MySQL versão 2)
- mysql.rds_import_binlog_ssl_material
- mysql.rds_next_master_log (Aurora MySQL versão 2)
- mysql.rds_next_source_log (Aurora MySQL versão 3)
- mysql.rds_remove_binlog_ssl_material
- mysql.rds_reset_external_master (Aurora MySQL versão 2)
- mysql.rds_reset_external_source (Aurora MySQL versão 3)
- mysql.rds_set_binlog_source_ssl (Aurora MySQL versão 3)
- mysql.rds_set_external_master (Aurora MySQL versão 2)
- mysql.rds_set_external_source (Aurora MySQL versão 3)
- mysql.rds_set_external_master_with_auto_position (Aurora MySQL versão 2)
- mysql.rds_set_external_source_with_auto_position (Aurora MySQL versão 3)
- mysql.rds_set_master_auto_position (Aurora MySQL versão 2)
- mysql.rds_set_read_only (Aurora MySQL versão 3)
- mysql.rds_set_session_binlog_format (Aurora MySQL versão 2)
- mysql.rds_set_source_auto_position (Aurora MySQL versão 3)
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until(Aurora MySQL versão 3)
- mysql.rds_stop_replication

mysql.rds_disable_session_binlog (Aurora MySQL versão 2)

Desativa o registro em log binário para a sessão atual definindo a variável `sql_log_bin` como `OFF`.

Sintaxe

```
CALL mysql.rds_disable_session_binlog;
```

Parâmetros

Nenhum

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

Quanto ao Aurora, esse procedimento é compatível com o Aurora MySQL versão 2.12 e em versões posteriores compatíveis com o MySQL 5.7.

> **Note**
>
> No Aurora MySQL versão 3, é possível usar o comando a seguir para desabilitar o registro em log binário para a sessão atual, se tiver o privilégio `SESSION_VARIABLES_ADMIN`:
>
> ```
> SET SESSION sql_log_bin = OFF;
> ```

mysql.rds_enable_session_binlog (Aurora MySQL versão 2)

Ativa o registro em log binário para a sessão atual definindo a variável `sql_log_bin` como `ON`.

Sintaxe

```
CALL mysql.rds_enable_session_binlog;
```

Parâmetros

Nenhum

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

Quanto ao Aurora, esse procedimento é compatível com o Aurora MySQL versão 2.12 e em versões posteriores compatíveis com o MySQL 5.7.

> **Note**
>
> No Aurora MySQL versão 3, é possível usar o comando a seguir para habilitar o registro em log binário para a sessão atual, se tiver o privilégio `SESSION_VARIABLES_ADMIN`:
>
> ```
> SET SESSION sql_log_bin = ON;
> ```

mysql.rds_import_binlog_ssl_material

Importa o certificado da autoridade de certificação, o certificado de cliente e a chave de cliente em um cluster de bancos de dados do Aurora MySQL. As informações são necessárias para a comunicação de SSL e a replicação criptografada.

> **Note**
>
> Atualmente, esse procedimento é compatível com o Aurora MySQL versão 2: 2.09.2, 2.10.0, 2.10.1, e 2.11.0; e versão 3: 3.01.1 e posteriores.

Sintaxe

```
CALL mysql.rds_import_binlog_ssl_material (
  ssl_material
);
```

Parâmetros

ssl_material

A carga de JSON que contém o conteúdo dos arquivos de formato .pem a seguir de um cliente MySQL:

- "ssl_ca":"*Certificado de autoridade de certificação*"
- "ssl_cert":"*Certificado de cliente*"
- "ssl_key":"*Chave de cliente*"

Observações de uso

Prepare para a replicação criptografada antes de executar este procedimento:

- Se você não tem o SSL habilitado na instância de banco de dados de origem externa do MySQL e não tem uma chave de cliente e um certificado de cliente preparados, habilite o SSL no servidor de banco de dados MySQL e gere a chave de cliente e o certificado de cliente necessários.
- Se o SSL estiver habilitado na instância de banco de dados de origem externa, forneça uma chave e um certificado de cliente para o cluster de bancos de dados Aurora MySQL. Se você não os tiver, gere uma nova chave e certificado para o cluster de bancos de dados Aurora MySQL. Para assinar o certificado de cliente, é necessário ter a chave da autoridade de certificação usada para configurar o SSL na instância de banco de dados de origem externa do MySQL.

Para obter mais informações, consulte Creating SSL certificates and keys using openssl na documentação do MySQL.

> **Important**
>
> Após preparar para a replicação criptografada, use uma conexão SSL para executar este procedimento. A chave de cliente não deve ser transferida por meio de uma conexão incerta.

Este procedimento importa informações SSL de um banco de dados MySQL externo em um cluster de banco de dados Aurora MySQL. As informações de SSL estão em arquivos de formato .pem que contêm as informações SSL do cluster de banco de dados Aurora MySQL. Durante a replicação criptografada, o cluster do banco de dados de Aurora MySQL age como um cliente para o servidor de banco de dados do MySQL. Os certificados e chaves do cliente Aurora MySQL estão nos arquivos em formato .pem.

Você pode copiar informações desses arquivos no parâmetro `ssl_material` na carga de JSON correta. Para suportar a replicação criptografada, importe essas informações SSL no cluster do banco de dados Aurora MySQL.

A carga de JSON deve estar no seguinte formato.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
ssl_ca_pem_body_code
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
```

```
ssl_cert_pem_body_code
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
ssl_key_pem_body_code
-----END RSA PRIVATE KEY-----\n"}'
```

Exemplos

O exemplo a seguir importa informações de SSL no Aurora MySQL. Em arquivos de formato .pem, o código do corpo geralmente é maior que o código de corpo exibido no exemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

mysql.rds_next_master_log (Aurora MySQL versão 2)

Altera a posição do log da instância de banco de dados de origem para o início do próximo log binário na instância de banco de dados de origem. Use este procedimento somente se estiver recebendo o erro 1236 de E/S de replicação em uma réplica de leitura.

Sintaxe

```
CALL mysql.rds_next_master_log(
```

```
curr_master_log
);
```

Parâmetros

curr_master_log

O índice do arquivo de log mestre atual. Por exemplo, se o arquivo atual dor denominado `mysql-bin-changelog.012345`, o índice será 12345. Para determinar o nome do arquivo de log do mestre, execute o comando `SHOW REPLICA STATUS` e veja o campo `Master_Log_File`.

Observações de uso

O usuário mestre deve executar o procedimento `mysql.rds_next_master_log`.

> **Warning**
>
> Chame `mysql.rds_next_master_log` somente se a replicação falhar após um failover de uma instância de banco de dados multi-AZ que for a origem da replicação, e o campo `Last_IO_Errno` do `SHOW REPLICA STATUS` reportar o erro 1236 de E/S. Chamar `mysql.rds_next_master_log` pode resultar em perda de dados na réplica de leitura caso as transações na instância de origem não tenham sido gravadas no log binário no disco antes do evento de failover.

Exemplos

Suponha que a replicação falhe em uma réplica de leitura do Aurora MySQL. A execução de `SHOW REPLICA STATUS\G` na réplica de leitura retorna o seguinte resultado:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
```

```
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

O campo `Last_IO_Errno` mostra que a instância está recebendo o erro 1236 de E/S. O campo `Master_Log_File` mostra que o nome do arquivo é `mysql-bin-changelog.012345`, o que significa que o índice de arquivos de log é `12345`. Para resolver o erro, chame `mysql.rds_next_master_log` com o seguinte parâmetro:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log (Aurora MySQL versão 3)

Altera a posição do log da instância de banco de dados de origem para o início do próximo log binário na instância de banco de dados de origem. Use este procedimento somente se estiver recebendo o erro 1236 de E/S de replicação em uma réplica de leitura.

Sintaxe

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

Parâmetros

curr_source_log

O índice do arquivo de log de origem atual. Por exemplo, se o arquivo atual dor denominado `mysql-bin-changelog.012345`, o índice será 12345. Para determinar o nome do arquivo de log de origem atual, execute o comando `SHOW REPLICA STATUS` e veja o campo `Source_Log_File`.

Observações de uso

O usuário administrativo precisa executar o procedimento `mysql.rds_next_source_log`.

Warning

Chame `mysql.rds_next_source_log` somente se a replicação falhar após um failover de uma instância de banco de dados multi-AZ que for a origem da replicação, e o campo `Last_IO_Errno` do `SHOW REPLICA STATUS` reportar o erro 1236 de E/S. Chamar `mysql.rds_next_source_log` pode resultar em perda de dados na réplica de leitura caso as transações na instância de origem não tenham sido gravadas no log binário no disco antes do evento de failover. Você pode reduzir a chance de que isso aconteça configurando os parâmetros `sync_binlog` e `innodb_support_xa` da instância de origem como 1, embora isso possa reduzir o desempenho.

Exemplos

Suponha que a replicação falhe em uma réplica de leitura do Aurora MySQL. A execução de `SHOW REPLICA STATUS\G` na réplica de leitura retorna o seguinte resultado:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
```

```
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

O campo `Last_IO_Errno` mostra que a instância está recebendo o erro 1236 de E/S. O campo `Source_Log_File` mostra que o nome do arquivo é `mysql-bin-changelog.012345`, o que significa que o índice de arquivos de log é `12345`. Para resolver o erro, chame `mysql.rds_next_source_log` com o seguinte parâmetro:

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_remove_binlog_ssl_material

Remove o certificado de autoridade de certificado, o certificado de cliente e a chave de cliente para comunicação SSL e replicação criptografada. Essas informações são importada usando mysql.rds_import_binlog_ssl_material.

Sintaxe

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_reset_external_master (Aurora MySQL versão 2)

Reconfigura uma instância de banco de dados do Aurora MySQL para que não seja mais uma réplica de leitura de uma instância do MySQL executada fora do Amazon RDS.

> **Important**
>
> Para executar esse procedimento, `autocommit` deve estar habilitado. Para habilitá-lo, defina o parâmetro `autocommit` como 1. Para obter informações sobre como modificar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Sintaxe

```
CALL mysql.rds_reset_external_master;
```

Observações de uso

O usuário mestre deve executar o procedimento `mysql.rds_reset_external_master`. Esse procedimento deve ser executado na instância de banco de dados MySQL a ser removida como a réplica de leitura de uma instância do MySQL sendo executada externamente ao Amazon RDS.

> **Note**
>
> Oferecemos esses procedimentos armazenados principalmente para permitir a replicação com instâncias do MySQL externas ao Amazon RDS. Recomendamos que você use réplicas do Aurora para gerenciar a replicação em um cluster de banco de dados do Aurora MySQL sempre que possível. Para obter informações sobre como gerenciar a replicação em clusters de banco de dados do Aurora MySQL, consulte Usar réplicas do Aurora.

Para obter mais informações sobre como usar a replicação para importar dados de uma instância do MySQL executada fora do Aurora MySQL, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

mysql.rds_reset_external_source (Aurora MySQL versão 3)

Reconfigura uma instância de banco de dados do Aurora MySQL para que não seja mais uma réplica de leitura de uma instância do MySQL executada fora do Amazon RDS.

> **Important**
>
> Para executar esse procedimento, `autocommit` deve estar habilitado. Para habilitá-lo, defina o parâmetro `autocommit` como 1. Para obter informações sobre como modificar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Sintaxe

```
CALL mysql.rds_reset_external_source;
```

Observações de uso

O usuário administrativo precisa executar o procedimento `mysql.rds_reset_external_source`. Esse procedimento deve ser executado na instância de banco de dados MySQL a ser removida como a réplica de leitura de uma instância do MySQL sendo executada externamente ao Amazon RDS.

> **Note**
>
> Oferecemos esses procedimentos armazenados principalmente para permitir a replicação com instâncias do MySQL externas ao Amazon RDS. Recomendamos que você use réplicas do Aurora para gerenciar a replicação em um cluster de banco de dados do Aurora MySQL sempre que possível. Para obter informações sobre como gerenciar a replicação em clusters de banco de dados do Aurora MySQL, consulte Usar réplicas do Aurora.

mysql.rds_set_binlog_source_ssl (Aurora MySQL versão 3)

Habilita a criptografia de `SOURCE_SSL` para replicação de logs binários. Consulte mais informações em CHANGE REPLICATION SOURCE TO statement na documentação do MySQL.

Sintaxe

```
CALL mysql.rds_set_binlog_source_ssl(mode);
```

Parâmetros

modo

Um valor que indica se a criptografia de `SOURCE_SSL` está habilitada:

- `0`: a criptografia `SOURCE_SSL` está desabilitada. O padrão é `0`.
- `1`: a criptografia `SOURCE_SSL` está habilitada. É possível configurar a criptografia usando SSL ou TLS.

Observações de uso

Esse procedimento é compatível com o Aurora MySQL versão 3.06 e posterior.

mysql.rds_set_external_master (Aurora MySQL versão 2)

Configura uma instância de banco de dados do Aurora MySQL para que seja uma réplica de leitura de uma instância do MySQL executada fora do Amazon RDS.

O procedimento `mysql.rds_set_external_master` está obsoleto e será removido em uma versão futura. Use `mysql.rds_set_external_source` em vez disso.

> **Important**
>
> Para executar esse procedimento, `autocommit` deve estar habilitado. Para habilitá-lo, defina o parâmetro `autocommit` como 1. Para obter informações sobre como modificar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Sintaxe

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parâmetros

host_name

O nome de host ou o endereço IP da instância do MySQL executada externamente ao Amazon RDS que se tornará a instância de banco de dados de origem.

host_port

A porta usada para executar a instância do MySQL executada externamente ao Amazon RDS a ser configurada como a instância de banco de dados de origem. Se sua configuração de rede inclui replicação de porta Secure Shell (SSH) que converte o número da porta, especifique o número da porta exposto pelo SSH.

replication_user_name

O ID de um usuário com as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância do MySQL executada externamente ao Amazon RDS. Recomendamos que você forneça uma conta que seja usada unicamente para a replicação com a instância externa.

replication_user_password

A senha do ID de usuário especificada em `replication_user_name`.

mysql_binary_log_file_name

O nome do log binário na instância de banco de dados de origem que contém as informações de replicação.

mysql_binary_log_file_location

O local no log binário `mysql_binary_log_file_name` no qual a replicação começa a ler as informações de replicação.

É possível determinar o nome e a localização do arquivo de binlog executando `SHOW MASTER STATUS` na instância do banco de dados de origem.

ssl_encryption

Um valor que especifica se a criptografia do Secure Sockets Layer (SSL) será usada na conexão de replicação. 1 especifica para usar criptografia de SSL, 0 especifica para não usar criptografia. O padrão é 0.

> **Note**
>
> A opção `MASTER_SSL_VERIFY_SERVER_CERT` não tem suporte. Essa opção é definida como 0, o que significa que a conexão é criptografada, mas os certificados não são verificados.

Observações de uso

O usuário mestre deve executar o procedimento `mysql.rds_set_external_master`. Esse procedimento deve ser executado na instância de banco de dados MySQL que será configurada como a réplica de leitura de uma instância do MySQL externa ao Amazon RDS.

Antes de executar `mysql.rds_set_external_master`, é necessário configurar a instância do MySQL executada externamente ao Amazon RDS para ser uma instância de banco de dados de

origem. Para conectar-se à instância do MySQL sendo executada externamente ao Amazon RDS, você deve especificar valores de `replication_user_name` e `replication_user_password` que indicam um usuário de replicação com permissões de `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância externa do MySQL.

Como configurar uma instância externa do MySQL como uma instância de banco de dados de origem

1. Usando o cliente do MySQL de sua escolha, conecte-se à instância externa do MySQL e crie uma conta de usuário a ser usada para a replicação. Veja um exemplo a seguir.

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

   > **Note**
   >
   > Especifique uma senha diferente do prompt mostrado aqui como prática recomendada de segurança.

2. Na instância externa do MySQL, conceda privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` para seu usuário de replicação. O exemplo a seguir concede privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` ao usuário 'repl_user' em todos os bancos de dados de seu domínio.

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Para usar a replicação criptografada, configure a instância de banco de dados de origem para usar conexões SSL. Além disso, importe o certificado da autoridade de certificação, o certificado e a chave de clientes na instância de banco de dados ou no cluster do banco de dados usando o procedimento mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Oferecemos esses procedimentos armazenados principalmente para permitir a replicação com instâncias do MySQL externas ao Amazon RDS. Recomendamos que você use réplicas do Aurora para gerenciar a replicação em um cluster de banco de dados do Aurora MySQL sempre que possível. Para obter informações sobre como gerenciar a replicação em clusters de banco de dados do Aurora MySQL, consulte Usar réplicas do Aurora.

Depois de chamar `mysql.rds_set_external_master` para configurar uma instância de banco de dados do Amazon RDS, é possível chamar mysql.rds_start_replication na réplica de leitura para iniciar o processo de replicação. É possível chamar mysql.rds_reset_external_master (Aurora MySQL versão 2) para remover a configuração da réplica de leitura.

Quando `mysql.rds_set_external_master` é chamado, o Amazon RDS grava a hora, o usuário e uma ação do `set master` nas tabelas `mysql.rds_history` e `mysql.rds_replication_status`.

Exemplos

Ao executar em uma instância de banco de dados MySQL, o exemplo a seguir configura a instância do banco de dados para ser uma réplica de leitura de uma instância do MySQL que é executada externamente ao Amazon RDS.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source (Aurora MySQL versão 3)

Configura uma instância de banco de dados do Aurora MySQL para que seja uma réplica de leitura de uma instância do MySQL executada fora do Amazon RDS.

> **Important**
>
> Para executar esse procedimento, `autocommit` deve estar habilitado. Para habilitá-lo, defina o parâmetro `autocommit` como 1. Para obter informações sobre como modificar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Sintaxe

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parâmetros

host_name

O nome de host ou o endereço IP da instância do MySQL executada externamente ao Amazon RDS que se tornará a instância de banco de dados de origem.

host_port

A porta usada para executar a instância do MySQL executada externamente ao Amazon RDS a ser configurada como a instância de banco de dados de origem. Se sua configuração de rede inclui replicação de porta Secure Shell (SSH) que converte o número da porta, especifique o número da porta exposto pelo SSH.

replication_user_name

O ID de um usuário com as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância do MySQL executada externamente ao Amazon RDS. Recomendamos que você forneça uma conta que seja usada unicamente para a replicação com a instância externa.

replication_user_password

A senha do ID de usuário especificada em `replication_user_name`.

mysql_binary_log_file_name

O nome do log binário na instância de banco de dados de origem que contém as informações de replicação.

mysql_binary_log_file_location

O local no log binário `mysql_binary_log_file_name` no qual a replicação começa a ler as informações de replicação.

É possível determinar o nome e a localização do arquivo de binlog executando `SHOW MASTER STATUS` na instância do banco de dados de origem.

ssl_encryption

Um valor que especifica se a criptografia do Secure Sockets Layer (SSL) será usada na conexão de replicação. 1 especifica para usar criptografia de SSL, 0 especifica para não usar criptografia. O padrão é 0.

> **Note**
>
> Você deve ter importado um certificado SSL personalizado usando mysql.rds_import_binlog_ssl_material para habilitar essa opção. Se você não importou um certificado SSL personalizado, defina esse parâmetro como 0 e use mysql.rds_set_binlog_source_ssl (Aurora MySQL versão 3) para habilitar o SSL para replicação de log binário.
> A opção `SOURCE_SSL_VERIFY_SERVER_CERT` não tem suporte. Essa opção é definida como 0, o que significa que a conexão é criptografada, mas os certificados não são verificados.

Observações de uso

O usuário administrativo precisa executar o procedimento `mysql.rds_set_external_source`. Esse procedimento deve ser executado na instância de banco de dados do Aurora MySQL que será configurada como a réplica de leitura de uma instância do MySQL externa ao Amazon RDS.

Antes de executar `mysql.rds_set_external_source`, é necessário configurar a instância do MySQL executada externamente ao Amazon RDS para ser uma instância de banco de dados de origem. Para conectar-se à instância do MySQL sendo executada externamente ao Amazon RDS, você deve especificar valores de `replication_user_name` e `replication_user_password` que indicam um usuário de replicação com permissões de `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância externa do MySQL.

Como configurar uma instância externa do MySQL como uma instância de banco de dados de origem

1. Usando o cliente do MySQL de sua escolha, conecte-se à instância externa do MySQL e crie uma conta de usuário a ser usada para a replicação. Veja um exemplo a seguir.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > Especifique uma senha diferente do prompt mostrado aqui como prática recomendada de segurança.

2. Na instância externa do MySQL, conceda privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` para seu usuário de replicação. O exemplo a seguir concede privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` ao usuário 'repl_user' em todos os bancos de dados de seu domínio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Para usar a replicação criptografada, configure a instância de banco de dados de origem para usar conexões SSL. Além disso, importe o certificado de autoridade de certificação, o certificado e a chave de clientes na instância de banco de dados ou no cluster de banco de dados usando o procedimento mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Oferecemos esses procedimentos armazenados principalmente para permitir a replicação com instâncias do MySQL externas ao Amazon RDS. Recomendamos que você use réplicas do Aurora para gerenciar a replicação em um cluster de banco de dados do Aurora MySQL sempre que possível. Para obter informações sobre como gerenciar a replicação em clusters de banco de dados do Aurora MySQL, consulte Usar réplicas do Aurora.

Depois de chamar `mysql.rds_set_external_source` para configurar uma instância de banco de dados do Aurora MySQL, é possível chamar mysql.rds_start_replication na réplica de leitura para iniciar o processo de replicação. É possível chamar mysql.rds_reset_external_source (Aurora MySQL versão 3) para remover a configuração da réplica de leitura.

Quando `mysql.rds_set_external_source` é chamado, o Amazon RDS grava a hora, o usuário e uma ação do `set master` nas tabelas `mysql.rds_history` e `mysql.rds_replication_status`.

Exemplos

Quando executado em uma instância de banco de dados do Aurora MySQL, o exemplo a seguir configura a instância de banco de dados para ser uma réplica de leitura de uma instância do MySQL que é executada externamente ao Amazon RDS.

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position (Aurora MySQL versão 2)

Configura uma instância primária do Aurora MySQL para aceitar a replicação de entrada de uma instância externa do MySQL. Esse procedimento também configura a replicação com base em identificadores de transação global (GTIDs).

Esse procedimento não configura a replicação atrasada, pois o Aurora MySQL não oferece suporte à replicação atrasada.

Sintaxe

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parâmetros

host_name

O nome de host ou o endereço IP da instância do MySQL sendo executada externamente ao Aurora que se torna a origem de replicação.

host_port

A porta utilizada para executar a instância do MySQL sendo executada externamente ao Aurora a ser configurada como a origem de replicação. Se sua configuração de rede inclui replicação de porta Secure Shell (SSH) que converte o número da porta, especifique o número da porta exposto pelo SSH.

replication_user_name

O ID de um usuário com as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância do MySQL executada externamente ao Aurora. Recomendamos que você forneça uma conta que seja usada unicamente para a replicação com a instância externa.

replication_user_password

A senha do ID de usuário especificada em `replication_user_name`.

ssl_encryption

Essa opção não está implementada atualmente. O padrão é 0.

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

O usuário mestre deve executar o procedimento `mysql.rds_set_external_master_with_auto_position`. O usuário principal executa esse procedimento na instância primária de um cluster de bancos de dados Aurora MySQL que atua como um destino de replicação. Este pode ser o destino de replicação de uma instância de banco de dados externa do MySQL ou um cluster de bancos de dados Aurora MySQL.

Esse procedimento é compatível com o Aurora MySQL versão 2. Para o Aurora MySQL versão 3, prefira usar o procedimento mysql.rds_set_external_source_with_auto_position (Aurora MySQL versão 3).

Antes de executar o `mysql.rds_set_external_master_with_auto_position`, configure a instância de banco de dados externa do MySQL para ser uma origem de replicação. Para conectar-se à instância externa do MySQL, especifique valores para `replication_user_name` e `replication_user_password`. Esses valores devem indicar um usuário de replicação que tenha as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância externa do MySQL.

Para configurar uma instância externa do MySQL como uma origem de replicação

1. Usando o cliente do MySQL de sua escolha, conecte-se à instância externa do MySQL e crie uma conta de usuário a ser usada para replicação. Veja um exemplo a seguir.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Para a instância externa do MySQL, conceda os privilégios `REPLICATION CLIENT` e `REPLICATION SLAVE` ao seu usuário de replicação. O exemplo a seguir concede privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` ao usuário `'repl_user'` em todos os bancos de dados de seu domínio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Quando você chama `mysql.rds_set_external_master_with_auto_position`, o Amazon RDS registra algumas informações. Essas informações incluem o horário, o usuário e uma ação de `"set master"` nas tabelas `mysql.rds_history` e `mysql.rds_replication_status`.

Para ignorar uma transação específica baseada em GTID que seja conhecida por causar problemas, use o procedimento armazenado mysql.rds_skip_transaction_with_gtid(Aurora MySQL versões 2 e 3). Para obter mais informações sobre como trabalhar com a replicação baseada em GTID, consulte Usar a replicação baseada em GTID.

Exemplos

Quando executado em uma instância primária do Aurora, o exemplo a seguir configura o cluster do Aurora para atuar como uma réplica de leitura de uma instância do MySQL executada fora do Aurora.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_external_source_with_auto_position (Aurora MySQL versão 3)

Configura uma instância primária do Aurora MySQL para aceitar a replicação de entrada de uma instância externa do MySQL. Esse procedimento também configura a replicação com base em identificadores de transação global (GTIDs).

Sintaxe

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parâmetros

host_name

O nome de host ou o endereço IP da instância do MySQL sendo executada externamente ao Aurora que se torna a origem de replicação.

host_port

A porta utilizada para executar a instância do MySQL sendo executada externamente ao Aurora a ser configurada como a origem de replicação. Se sua configuração de rede inclui replicação de porta Secure Shell (SSH) que converte o número da porta, especifique o número da porta exposto pelo SSH.

replication_user_name

O ID de um usuário com as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância do MySQL executada externamente ao Aurora. Recomendamos que você forneça uma conta que seja usada unicamente para a replicação com a instância externa.

replication_user_password

A senha do ID de usuário especificada em `replication_user_name`.

ssl_encryption

Essa opção não está implementada atualmente. O padrão é 0.

> **Note**
>
> Use mysql.rds_set_binlog_source_ssl (Aurora MySQL versão 3) para habilitar o SSL para replicação de logs binários.

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

O usuário administrativo precisa executar o procedimento `mysql.rds_set_external_source_with_auto_position`. O usuário administrativo executa esse procedimento na instância primária de um cluster de bancos de dados Aurora MySQL que atua como um destino de replicação. Este pode ser o destino de replicação de uma instância de banco de dados externa do MySQL ou um cluster de bancos de dados Aurora MySQL.

Esse procedimento é compatível com o Aurora MySQL versão 3. Esse procedimento não configura a replicação atrasada, pois o Aurora MySQL não oferece suporte à replicação atrasada.

Antes de executar o `mysql.rds_set_external_source_with_auto_position`, configure a instância de banco de dados externa do MySQL para ser uma origem de replicação. Para

conectar-se à instância externa do MySQL, especifique valores para `replication_user_name` e `replication_user_password`. Esses valores devem indicar um usuário de replicação que tenha as permissões `REPLICATION CLIENT` e `REPLICATION SLAVE` na instância externa do MySQL.

Para configurar uma instância externa do MySQL como uma origem de replicação

1. Usando o cliente do MySQL de sua escolha, conecte-se à instância externa do MySQL e crie uma conta de usuário a ser usada para replicação. Veja um exemplo a seguir.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Para a instância externa do MySQL, conceda os privilégios `REPLICATION CLIENT` e `REPLICATION SLAVE` ao seu usuário de replicação. O exemplo a seguir concede privilégios de `REPLICATION CLIENT` e `REPLICATION SLAVE` ao usuário `'repl_user'` em todos os bancos de dados de seu domínio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Quando você chama `mysql.rds_set_external_source_with_auto_position`, o Amazon RDS registra algumas informações. Essas informações incluem o horário, o usuário e uma ação de `"set master"` nas tabelas `mysql.rds_history` e `mysql.rds_replication_status`.

Para ignorar uma transação específica baseada em GTID que seja conhecida por causar problemas, use o procedimento armazenado mysql.rds_skip_transaction_with_gtid(Aurora MySQL versões 2 e 3). Para obter mais informações sobre como trabalhar com a replicação baseada em GTID, consulte Usar a replicação baseada em GTID.

Exemplos

Quando executado em uma instância primária do Aurora, o exemplo a seguir configura o cluster do Aurora para atuar como uma réplica de leitura de uma instância do MySQL executada fora do Aurora.

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_master_auto_position (Aurora MySQL versão 2)

Define o modo de replicação de base nas posições do arquivo de log binário ou nos identificadores de transações globais (GTIDs).

Sintaxe

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

Parâmetros

auto_position_mode

Um valor que indica se será usada a replicação de posição do arquivo de log ou a replicação com base no GTID:

- `0`: usar o método de replicação com base na posição do arquivo de log binário. O padrão é `0`.
- `1`: usar o método de replicação com base no GTID.

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_set_master_auto_position`.

Esse procedimento é compatível com o Aurora MySQL versão 2.

mysql.rds_set_read_only (Aurora MySQL versão 3)

Ativa ou desativa globalmente o modo `read_only` da instância de banco de dados.

Sintaxe

```
CALL mysql.rds_set_read_only(mode);
```

Parâmetros

modo

Um valor que indica se o modo `read_only` está ativado ou desativado globalmente para a instância de banco de dados:

- `0`: OFF. O padrão é `0`.

- 1 – ON

Observações de uso

O procedimento `mysql.rds_set_read_only` armazenado modifica somente o parâmetro `read_only`. O parâmetro `innodb_read_only` não pode ser alterado nas instâncias de banco de dados do leitor.

A alteração do parâmetro `read_only` não persiste na reinicialização. Para fazer alterações permanentes em `read_only`, é necessário usar o parâmetro `read_only` do cluster de banco de dados.

Esse procedimento é compatível com o Aurora MySQL versão 3.06 e posterior.

mysql.rds_set_session_binlog_format (Aurora MySQL versão 2)

Define o formato de log binário para a sessão atual.

Sintaxe

```
CALL mysql.rds_set_session_binlog_format(format);
```

Parâmetros

formato

Um valor que indica o formato do log binário da sessão atual:

- `STATEMENT`: a origem de replicação grava eventos no log binário com base em declarações SQL.
- `ROW`: a origem de replicação grava eventos no log binário que indicam alterações em linhas individuais da tabela.
- `MIXED`: o registro em log geralmente é baseado em declarações SQL, mas muda para linhas em determinadas condições. Para receber mais informações, consulte Mixed Binary Logging Format na documentação do MySQL.

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

Para usar esse procedimento armazenado, você deve ter o registro em log binário configurado para a sessão atual.

Quanto ao Aurora, esse procedimento é compatível com o Aurora MySQL versão 2.12 e em versões posteriores compatíveis com o MySQL 5.7.

mysql.rds_set_source_auto_position (Aurora MySQL versão 3)

Define o modo de replicação de base nas posições do arquivo de log binário ou nos identificadores de transações globais (GTIDs).

Sintaxe

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

Parâmetros

auto_position_mode

Um valor que indica se será usada a replicação de posição do arquivo de log ou a replicação com base no GTID:

- `0`: usar o método de replicação com base na posição do arquivo de log binário. O padrão é `0`.
- 1: usar o método de replicação com base no GTID.

Observações de uso

Para um cluster de bancos de dados Aurora MySQL, você chama esse procedimento armazenado enquanto está conectado à instância primária.

O usuário administrativo precisa executar o procedimento `mysql.rds_set_source_auto_position`.

mysql.rds_skip_repl_error

Ignora e exclui um erro de replicação em uma réplica de leitura de banco de dados MySQL.

Sintaxe

```
CALL mysql.rds_skip_repl_error;
```

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_skip_repl_error` em uma réplica de leitura. Para obter mais informações sobre esse procedimento, consulte Ignorar o erro de replicação atual.

Para determinar se há erros, execute o comando `SHOW REPLICA STATUS\G` do MySQL. Se um erro de replicação não for crítico, execute `mysql.rds_skip_repl_error` para ignorá-lo. Se houver vários, `mysql.rds_skip_repl_error` exclui o primeiro erro, depois avisa que há outros ainda. Assim, você pode usar `SHOW REPLICA STATUS\G` para determinar o plano de ação correto para o próximo erro. Para obter informações sobre os valores retornados, consulte Instrução SHOW REPLICA STATUS na documentação do MySQL.

Para obter mais informações sobre como lidar com erros de replicação no Aurora MySQL, consulte Diagnosticar e resolver uma falha de replicação de leitura do MySQL .

Erro de replicação interrompida

Ao chamar o procedimento `mysql.rds_skip_repl_error`, você pode receber uma mensagem de erro informando que a réplica está inativa ou desativada.

Essa mensagem de erro aparece quando você executa o procedimento na instância primária em vez da réplica de leitura. Você deve executar esse procedimento na réplica de leitura para que o procedimento funcione.

Essa mensagem de erro também poderá aparecer se você executar o procedimento na réplica de leitura, mas a replicação não poderá ser reiniciada com êxito.

Se você precisar ignorar um grande número de erros, o atraso de replicação poderá aumentar além do período de retenção padrão para arquivos de log binário (binlog). Nesse caso, você poderá encontrar um erro fatal, com os arquivos binlog sendo limpos antes de sua reprodução na réplica de leitura. Essa remoção faz com que a replicação pare, e você não consegue chamar o comando `mysql.rds_skip_repl_error` para ignorar erros de replicação.

É possível mitigar esse problema aumentando o número de horas em que os arquivos binlog são retidos na instância de banco de dados de origem. Após aumentar o período de retenção de log binário, você pode reiniciar a replicação e chamar o comando `mysql.rds_skip_repl_error` conforme necessário.

Para definir o período de retenção do binlog, use o procedimento mysql.rds_set_configuration e especifique um parâmetro de configuração `'binlog retention hours'`, juntamente com o

número de horas de retenção dos arquivos binlog no cluster do banco de dados. O exemplo a seguir define o período de retenção para arquivos de log binário em 48 horas.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

Inicia a replicação de um cluster de banco de dados do Aurora MySQL.

> **Note**
>
> Você pode usar o procedimento armazenado mysql.rds_start_replication_until(Aurora MySQL versão 3) ou mysql.rds_start_replication_until_gtid(Aurora MySQL versão 3) para iniciar a replicação de uma instância de banco de dados do Aurora MySQL e interromper a replicação no local do arquivo de log binário especificado.

Sintaxe

```
CALL mysql.rds_start_replication;
```

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_start_replication`.

Para importar dados de uma instância do MySQL fora do Amazon RDS, chame `mysql.rds_start_replication` na réplica de leitura para iniciar o processo de replicação depois de ter chamado mysql.rds_set_external_master (Aurora MySQL versão 2) ou mysql.rds_set_external_source (Aurora MySQL versão 3) para criar a configuração de replicação. Para obter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Para exportar dados para uma instância do MySQL fora do Amazon RDS, chame `mysql.rds_start_replication` e `mysql.rds_stop_replication` na réplica de leitura para controlar algumas ações de replicação, como a remoção de logs binários. Para obter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Também é possível chamar `mysql.rds_start_replication` na réplica de leitura para reiniciar qualquer processo de replicação que tenha sido interrompido anteriormente chamando `mysql.rds_stop_replication`. Para obter mais informações, consulte Erro de replicação interrompida.

mysql.rds_start_replication_until(Aurora MySQL versão 3)

Inicia a replicação de um cluster de banco de dados do Aurora MySQL e interrompe a replicação no local do arquivo de log binário especificado.

Sintaxe

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

Parâmetros

replication_log_file

O nome do log binário na instância de banco de dados de origem que contém as informações de replicação.

replication_stop_point

O local no log binário `replication_log_file` no qual a replicação será interrompida.

Observações de uso

O usuário mestre deve executar o procedimento `mysql.rds_start_replication_until`.

Esse procedimento é compatível com o Aurora MySQL versão 3.04 e posterior.

O procedimento armazenado `mysql.rds_start_replication_until` não é compatível com a replicação gerenciada, o que inclui o seguinte:

- Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS
- Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora

O nome do arquivo especificado para o parâmetro `replication_log_file` deve corresponder ao nome do arquivo do log binário da instância de banco de dados de origem.

Quando o parâmetro `replication_stop_point` especifica um local de parada no passado, a replicação é interrompida imediatamente.

Exemplos

O exemplo a seguir inicia a replicação e replica as alterações até que ela atinja o local `120` no arquivo de log binário `mysql-bin-changelog.000777`.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

Interrompe a replicação de uma instância de banco de dados MySQL.

Sintaxe

```
CALL mysql.rds_stop_replication;
```

Observações de uso

O usuário mestre deve executar o procedimento `mysql.rds_stop_replication`.

Se você estiver configurando a replicação para importar dados de uma instância do MySQL em execução externamente ao Amazon RDS, chame `mysql.rds_stop_replication` na réplica de leitura para encerrar o processo de replicação após a importação ter sido concluída. Para obter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

Se estiver configurando a replicação para exportar dados para uma instância do MySQL externa ao Amazon RDS, chame `mysql.rds_start_replication` e `mysql.rds_stop_replication` na réplica de leitura para controlar algumas ações de replicação, como a remoção de logs binários. Para obter mais informações, consulte Replicação entre Aurora e o MySQL ou entre Aurora e outro cluster de banco de dados do Aurora (replicação de log binário).

O procedimento armazenado `mysql.rds_stop_replication` não é compatível com a replicação gerenciada, o que inclui o seguinte:

- Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS
- Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora

Encerrar uma sessão ou consulta

Os procedimentos armazenados a seguir encerram uma sessão ou consulta.

Tópicos

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

Encerra uma conexão ao servidor MySQL.

Sintaxe

```
CALL mysql.rds_kill(processID);
```

Parâmetros

processID

A identidade do thread de conexão a ser encerrada.

Observações de uso

Cada conexão ao servidor do MySQL é executada em um thread separado. Para encerrar uma conexão, use o procedimento `mysql.rds_kill` e passe o ID de thread dessa conexão. Para obter o ID de thread, use o comando SHOW PROCESSLIST do MySQL.

Exemplos

O exemplo a seguir encerra uma conexão com um ID de thread de 4243:

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

Encerra uma consulta em execução no servidor MySQL.

Sintaxe

```
CALL mysql.rds_kill_query(processID);
```

Parâmetros

processID

A identidade do processo ou thread que está executando a consulta a ser encerrada.

Observações de uso

Para encerrar uma consulta em execução no servidor MySQL, use o procedimento `mysql_rds_kill_query` e passe o ID do thread que está executando a consulta. O procedimento então encerra a conexão.

Para obter o ID, consulte a tabela INFORMATION_SCHEMA PROCESSLIST do MySQL ou use o comando SHOW PROCESSLIST do MySQL. O valor na coluna ID de `SHOW PROCESSLIST` ou `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` é *processID*.

Exemplos

O seguinte exemplo encerra uma consulta com um ID de thread de consulta 230040:

```
CALL mysql.rds_kill_query(230040);
```

Replicar transações com GTIDs

Os procedimentos armazenados a seguir controlam como as transações são replicadas usando identificadores de transações globais (GTIDs) com o Aurora MySQL. Para saber como usar a replicação com base em GTIDs com o Aurora MySQL, consulte Usar a replicação baseada em GTID.

Tópicos

- mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versão 3)
- mysql.rds_gtid_purged (Aurora MySQL versão 3)
- mysql.rds_skip_transaction_with_gtid(Aurora MySQL versões 2 e 3)
- mysql.rds_start_replication_until_gtid(Aurora MySQL versão 3)

mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versão 3)

Configura a opção `ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS` da instrução `CHANGE REPLICATION SOURCE TO`. Faz com que o canal de replicação atribua um GTID a transações replicadas que não têm um. Assim, é possível realizar a replicação de logs binários de uma origem que não utiliza replicação baseada em GTID para uma réplica que a utiliza. Para obter mais informações, consulte a Instrução CHANGE REPLICATION SOURCE TO e o tópico sobre Replicação de uma origem sem GTIDs para uma réplica com GTIDs, no Guia de referência do MySQL.

Sintaxe

```
CALL mysql.rds_assign_gtids_to_anonymous_transactions(gtid_option);
```

Parâmetros

`gtid_option`

Valor da string. Os valores permitidos são `OFF`, `LOCAL` ou um UUID especificado.

Observações de uso

Esse procedimento tem o mesmo efeito que a emissão da instrução `CHANGE REPLICATION SOURCE TO ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS = gtid_option` na comunidade do MySQL.

O GTID deve se transformar `ON` para que *`gtid_option`* seja definido como `LOCAL` ou um UUID específico.

O padrão é `OFF`, o que significa que o recurso não é utilizado.

`LOCAL` atribui um GTID que inclui o próprio UUID da réplica (a configuração `server_uuid`).

Transmitir um parâmetro que é um UUID atribui um GTID que inclui o UUID especificado, como a configuração `server_uuid` do servidor de origem de replicação.

Exemplos

Para desabilitar esse recurso:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('OFF');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: OFF |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Para utilizar o próprio UUID da réplica:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('LOCAL');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: LOCAL |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Para utilizar um UUID especificado:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('317a4760-
f3dd-3b74-8e45-0615ed29de0e');
+-------------------------------------------------------------------------------------------------
+
| Message |
+-------------------------------------------------------------------------------------------------
+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: 317a4760-
f3dd-3b74-8e45-0615ed29de0e |
```

```
+-----------------------------------------------------------------------------------------------
+
1 row in set (0.07 sec)
```

mysql.rds_gtid_purged (Aurora MySQL versão 3)

Define o valor global da variável de sistema `gtid_purged` como determinado conjunto de identificadores de transação global (GTID). A variável de sistema `gtid_purged` é um conjunto de GTID que consiste nos GTIDs de todas as transações que foram confirmadas no servidor mas não existem em nenhum arquivo de log binário no servidor.

Para permitir a compatibilidade com o MySQL 8.0, há duas maneiras de definir o valor de `gtid_purged`:

- Substituir o valor de `gtid_purged` por seu conjunto de GTIDs especificado.
- Anexar seu conjunto de GTIDs especificado ao conjunto de GTIDs que `gtid_purged` já contém.

Sintaxe

Como substituir o valor de `gtid_purged` por seu conjunto de GTIDs especificado:

```
CALL mysql.rds_gtid_purged (gtid_set);
```

Como anexar o valor de `gtid_purged` ao seu conjunto de GTIDs especificado:

```
CALL mysql.rds_gtid_purged (+gtid_set);
```

Parâmetros

`gtid_set`

O valor de *`gtid_set`* deve ser um superconjunto do valor atual de `gtid_purged` e não pode fazer uma intersecção com `gtid_subtract(gtid_executed,gtid_purged)`. Ou seja, o novo conjunto de GTIDs deve incluir todos os GTIDs que já estavam em `gtid_purged` e não pode incluir nenhum GTID em `gtid_executed` que ainda não tenha sido eliminado. O parâmetro *`gtid_set`* também não pode incluir nenhum GTID que esteja no conjunto `gtid_owned` global, os GTIDs para transações que estão sendo processadas no momento no servidor.

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_gtid_purged`.

Esse procedimento é compatível com o Aurora MySQL versão 3.04 e posterior.

Exemplos

O exemplo a seguir atribui o GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` à variável `gtid_purged` global.

```
CALL mysql.rds_gtid_purged('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_skip_transaction_with_gtid(Aurora MySQL versões 2 e 3)

Ignora a replicação de uma transação com o identificador de transação global (GTID) especificado em uma instância primária do Aurora.

Você pode usar esse procedimento para a recuperação de desastres, quando uma transação baseada em GTID específica for conhecida por causar desastres. Use esse procedimento armazenado para ignorar a transação problemática. Exemplos de transações problemáticas incluem transações que desabilitam a replicação, excluem dados importantes ou fazem com que a instância de banco de dados se torne indisponível.

Sintaxe

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

Parâmetros

gtid_to_skip

O GTID da transação de replicação a ser ignorada.

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_skip_transaction_with_gtid`.

Esse procedimento é compatível com o Aurora MySQL versões 2 e 3.

Exemplos

O exemplo a seguir ignora a replicação da transação com o GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid(Aurora MySQL versão 3)

Inicia a replicação de um cluster de banco de dados do Aurora MySQL e interrompe a replicação logo depois do identificador de transação global (GTID) especificado.

Sintaxe

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

Parâmetros

gtid

O GTID após o qual a replicação será interrompida.

Observações de uso

O usuário principal deve executar o procedimento `mysql.rds_start_replication_until_gtid`.

Esse procedimento é compatível com o Aurora MySQL versão 3.04 e posterior.

O procedimento armazenado `mysql.rds_start_replication_until_gtid` não é compatível com a replicação gerenciada, o que inclui o seguinte:

- Replicar clusters de banco de dados do Amazon Aurora MySQL entre Regiões da AWS
- Migrar de uma instância de banco de dados do RDS para MySQL para um cluster de banco de dados do Amazon Aurora MySQL usando uma réplica de leitura do Aurora

Quando o parâmetro `gtid` especifica uma transação que já tenha sido executada pela réplica, a replicação é interrompida imediatamente.

Exemplos

O exemplo a seguir inicia a replicação e replica as alterações até atingir o GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

Alternar os logs de consulta

Os procedimentos armazenados a seguir fazem a rotação dos logs do MySQL para tabelas de backup. Para ter mais informações, consulte Arquivos de log do banco de dados AuroraMySQL.

Tópicos

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

Reveza a tabela `mysql.general_log` com uma tabela de backup.

Sintaxe

```
CALL mysql.rds_rotate_general_log;
```

Observações de uso

Você pode revezar a tabela `mysql.general_log` com uma tabela de backup, chamando o procedimento `mysql.rds_rotate_general_log`. Quando as tabelas de log são revezadas, a tabela de log atual é copiada para uma tabela de log de backup e as entradas na tabela de log atual são removidas. Se uma tabela de log de backup já existir, então ela será excluída antes que a tabela de log atual seja copiada ao backup. Você pode consultar a tabela de log de backup, se necessário. A tabela de log de backup para a tabela `mysql.general_log` é denominada `mysql.general_log_backup`.

É possível executar esse procedimento somente quando o parâmetro `log_output` está definido como `TABLE`.

mysql.rds_rotate_slow_log

Reveza a tabela `mysql.slow_log` com uma tabela de backup.

Sintaxe

```
CALL mysql.rds_rotate_slow_log;
```

Observações de uso

Você pode revezar a tabela `mysql.slow_log` com uma tabela de backup, chamando o procedimento `mysql.rds_rotate_slow_log`. Quando as tabelas de log são revezadas, a tabela de log atual é copiada para uma tabela de log de backup e as entradas na tabela de log atual são removidas. Se uma tabela de log de backup já existir, então ela será excluída antes que a tabela de log atual seja copiada ao backup.

Você pode consultar a tabela de log de backup, se necessário. A tabela de log de backup para a tabela `mysql.slow_log` é denominada `mysql.slow_log_backup`.

Definir e mostrar a configuração de logs binários

Os procedimentos armazenados a seguir definem e mostram parâmetros de configuração, como para retenção de arquivos de log binários.

Tópicos

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

Especifica o número de horas para retenção de logs binários ou o número de segundos para atrasar a replicação.

Sintaxe

```
CALL mysql.rds_set_configuration(name,value);
```

Parâmetros

name

(Opcional) O nome do parâmetro de configuração a definir.

value

O valor do parâmetro de configuração.

Observações de uso

O procedimento `mysql.rds_set_configuration` oferece suporte aos seguintes parâmetros de configuração:

- horas de retenção do log binário

Os parâmetros de configuração são armazenados permanentemente e sobrevivem a qualquer reinicialização ou failover da instância de banco de dados.

horas de retenção do log binário

O parâmetro `binlog retention hours` é usado para especificar o número de horas para reter arquivos de log binários. O Amazon Aurora normalmente elimina um log binário o mais rápido possível, mas o log binário ainda pode ser necessário para a replicação com um banco de dados MySQL externo ao Aurora.

O valor padrão de `binlog retention hours` é `NULL`. No Aurora MySQL, `NULL` significa que os logs binários são limpos lentamente. Os registros binários do Aurora MySQL podem permanecer no sistema por determinado período, que geralmente não passa de um dia.

Para especificar o número de horas para reter os logs binários em um cluster de banco de dados, use o procedimento armazenado `mysql.rds_set_configuration` e especifique um período com tempo suficiente para que a replicação ocorra, conforme exibido no exemplo a seguir.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> Não é possível usar o valor `0` para `binlog retention hours`.

Para clusters de banco de dados do Aurora MySQL versão 2.11.0 e posterior e clusters de banco de dados versão 3, o valor máximo de `binlog retention hours` é 2.160 (90 dias).

Após configurar o período de retenção, monitore o uso de armazenamento da instância de banco de dados para garantir que os logs binários retidos não consumam muito armazenamento.

mysql.rds_show_configuration

A quantidade de horas em que os logs binários são mantidos.

Sintaxe

```
CALL mysql.rds_show_configuration;
```

Observações de uso

Para verificar o número de horas durante as quais o Amazon RDS vai reter os logs binários, use o procedimento armazenado `mysql.rds_show_configuration`.

Exemplos

O exemplo a seguir mostra exibe o período de retenção:

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

Tabelas information_schema específicas do Aurora MySQL

O Aurora MySQL tem determinadas tabelas `information_schema` que são específicas do Aurora.

information_schema.aurora_global_db_instance_status

A tabela `information_schema.aurora_global_db_instance_status` contém informações sobre o status de todas as instâncias de banco de dados nos clusters de banco de dados primários e secundários de um banco de dados global. A tabela a seguir mostra as colunas que você pode usar. As colunas restantes são somente para uso interno do Aurora.

> **Note**
>
> As tabelas de esquema de informações só está disponível com bancos de dados globais do Aurora MySQL versão 3.04.0 e posterior.

Coluna	Tipo de dados	Descrição
SERVER_ID	varchar(100)	O identificador da instância de banco de dados.
SESSION_ID	varchar(100)	Um identificador exclusivo da sessão atual. O valor de `MASTER_SESSION_ID` identifica a instância de banco de dados do leitor (primário).
AWS_REGION	varchar(100)	A Região da AWS onde essa instância de banco de dados

Coluna	Tipo de dados	Descrição
		global é executada. Para obter uma lista de regiões, consulte Disponibilidade de regiões.
DURABLE_LSN	bigint não assinado	O número de sequência de logs (LSN) que se tornou durável no armazenamento. Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.
HIGHEST_LSN_RCVD	bigint não assinado	O LSN mais alto recebido pela instância de banco de dados da instância de banco de dados do gravador.
OLDEST_READ_VIEW_TRX_ID	bigint não assinado	O ID da transação mais antiga que a instância de banco de dados do gravador pode limpar.
OLDEST_READ_VIEW_LSN	bigint não assinado	O LSN mais antigo usado pela instância de banco de dados para fazer a leitura no armazenamento.

Coluna	Tipo de dados	Descrição
VISIBILITY_LAG_IN_MSEC	float(10,0) não assinado	Para leitores no cluster de banco de dados primário, até que ponto essa instância de banco de dados está atrasada em relação à instância de banco de dados de gravador em milissegundos. Para leitores em um cluster de banco de dados secundário, até que ponto essa instância de banco de dados está atrasada em relação ao volume secundário em milissegundos.

information_schema.aurora_global_db_status

A tabela `information_schema.aurora_global_db_status` contém informações sobre diversos aspectos do atraso do banco de dados global do Aurora, especificamente o atraso do armazenamento do Aurora subjacente (o chamado atraso de durabilidade) e o atraso entre o objetivo de ponto de recuperação (RPO). A tabela a seguir mostra as colunas que você pode usar. As colunas restantes são somente para uso interno do Aurora.

> **Note**
>
> As tabelas de esquema de informações só está disponível com bancos de dados globais do Aurora MySQL versão 3.04.0 e posterior.

Coluna	Tipo de dados	Descrição
AWS_REGION	varchar(100)	A Região da AWS onde essa instância de banco de dados global é executada. Para obter

Coluna	Tipo de dados	Descrição
		uma lista de regiões, consulte Disponibilidade de regiões.
HIGHEST_LSN_WRITTEN	bigint não assinado	O número de sequência de logs (LSN) mais alto nesse cluster de banco de dados no momento. Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.
DURABILITY_LAG_IN_MILLISECONDS	float(10,0) não assinado	A diferença nos valores de carimbo de data/hora entre o `HIGHEST_LSN_WRITTEN` em um cluster de banco de dados secundário e o `HIGHEST_LSN_WRITTEN` no cluster de banco de dados primário. Esse valor é sempre 0 no cluster de banco de dados primário do banco de dados global do Aurora.

Coluna	Tipo de dados	Descrição
RPO_LAG_IN_MILLISECONDS	float(10,0) não assinado	O atraso do objetivo de ponto de recuperação (RPO). O atraso do RPO é o tempo necessário para que a transação COMMIT mais recente do usuário seja armazenada em um cluster de banco de dados secundário após ser armazenada no cluster de banco de dados primário de um banco de dados Aurora global. Esse valor é sempre 0 no cluster de banco de dados primário do banco de dados global do Aurora. Em termos simples, essa métrica calcula o objetivo do ponto de recuperação de cada cluster de banco de dados do Aurora MySQL no banco de dados global do Aurora, ou seja, quantos dados poderão ser perdidos se houver uma interrupção. Tal como o atraso, o RPO é medido em tempo.

Coluna	Tipo de dados	Descrição
LAST_LAG_CALCULATION_TIMESTAMP	datetime	O carimbo de data/hora que especifica quando os valores foram calculados pela última vez para `DURABILITY_LAG_IN_MILLISECONDS` e `RPO_LAG_IN_MILLISECONDS`. Um valor de tempo como `1970-01-01 00:00:00+00` indica que este é o cluster de banco de dados primário.
OLDEST_READ_VIEW_TRX_ID	bigint não assinado	O ID da transação mais antiga que a instância de banco de dados do gravador pode limpar.

information_schema.replica_host_status

A tabela `information_schema.replica_host_status` contém informações de replicação. As colunas que você pode usar são mostradas na tabela a seguir. As colunas restantes são somente para uso interno do Aurora.

Coluna	Tipo de dados	Descrição
CPU	double	A porcentagem de uso da CPU do host de réplica.
IS_CURRENT	tinyint	Se a réplica é atual.
LAST_UPDATE_TIMESTAMP	datetime(6)	Hora em que a última atualização ocorreu. Utilizado para determinar se um registro está obsoleto.

Coluna	Tipo de dados	Descrição
REPLICA_LAG_IN_MILLISECONDS	double	O atraso da réplica em milissegundos.
SERVER_ID	varchar(100)	O ID do servidor de banco de dados.
SESSION_ID	varchar(100)	O ID da sessão do banco de dados. Utilizado para determinar se uma instância de banco de dados é uma instância do gravador ou do leitor.

> **Note**
>
> Quando uma instância de réplica fica para trás, as informações consultadas em sua tabela `information_schema.replica_host_status` podem estar desatualizadas. Nessa situação, recomendamos que você consulte a instância do gravador em vez disso. Embora a tabela `mysql.ro_replica_status` tenha informações semelhantes, não recomendamos que você a use.

information_schema.aurora_forwarding_processlist

A tabela `information_schema.aurora_forwarding_processlist` contém informações sobre os processos envolvidos no encaminhamento de gravação.

O conteúdo dessa tabela é visível somente na instância de banco de dados do gravador de um cluster de banco de dados com o encaminhamento de gravação global ou no cluster ativado. Um conjunto de resultados vazio é retornado nas instâncias de banco de dados do leitor.

Campo	Tipo de dados	Descrição
ID	bigint	O identificador da conexão na instância de banco de dados do gravador. Esse identificador é o mesmo valor exibido na coluna `Id` da declaração `SHOW PROCESSLIST` e devolvida pela função `CONNECTION_ID()` dentro do segmento.
USER	varchar(32)	O usuário do MySQL que emitiu a declaração.
HOST	varchar(255)	O cliente do MySQL que emitiu a declaração. Para declarações encaminhadas, esse campo mostra o endereço do host do cliente da aplicação que estabeleceu a conexão na instância de banco de dados do leitor de encaminhamento.
Banco de dados	varchar(64)	O banco de dados padrão para o segmento.
COMMAND	varchar(16)	O tipo de comando que o segmento está executando em nome do cliente ou `Sleep` se a sessão estiver inativa. Para conhecer as descrições dos comandos de segmento, consulte Thread Command Values na documentação do MySQL.
TIME	int	O tempo em segundos que o segmento esteve no estado atual.
STATE	varchar(64)	Uma ação, um evento ou um estado que indica o que o segmento está fazendo. Para conhecer as descrições dos valores de estado, consulte General Thread States na documentação do MySQL.
INFO	longtext	A declaração que o segmento está executando ou `NULL` se não estiver executando uma declaração. A declaração pode ser a enviada ao servidor ou uma declaração mais interna se a declaração executar outras declarações.

Campo	Tipo de dados	Descrição
IS_FORWARDED	bigint	Indica se o segmento foi encaminhado de uma instância de banco de dados do leitor.
REPLICA_SESSION_ID	bigint	O identificador de conexão na réplica do Aurora. Esse identificador é o mesmo valor exibido na coluna `Id` da declaração `SHOW PROCESSLIST` na instância de banco de dados do leitor de encaminhamento do Aurora.
REPLICA_INSTANCE_IDENTIFIER	varchar(64)	O identificador da instância de banco de dados do segmento de encaminhamento.
REPLICA_CLUSTER_NAME	varchar(64)	O identificador do cluster de banco de dados do segmento de encaminhamento. Para encaminhamento de gravação no cluster, esse identificador é o mesmo cluster de banco de dados da instância de banco de dados do gravador.
REPLICA_REGION	varchar(64)	A Região da AWS da qual o segmento de encaminhamento se origina. Para encaminhamento de gravação no cluster, essa região é a mesma Região da AWS da instância de banco de dados do gravador.

Atualizações do mecanismo de banco de dados Amazon Aurora MySQL

O Amazon Aurora lança atualizações regularmente. Essas atualizações são aplicadas a clusters de bancos de dados Aurora durante janelas de manutenção do sistema. O cronograma de aplicação das atualizações depende da configuração da região e da janela de manutenção para o cluster de banco de dados, bem como do tipo de atualização.

As versões do Amazon Aurora são disponibilizadas para todas as Regiões da AWS ao longo de vários dias. Algumas regiões podem mostrar temporariamente uma versão do mecanismo que ainda não está disponível em uma região diferente.

As atualizações são aplicadas a todas as instâncias em um cluster de banco de dados simultaneamente. Como uma atualização exige a reinicialização de um banco de dados em todas as instâncias de um cluster de banco de dados, você enfrenta de 20 a 30 segundos de inatividade. Depois disso, você poderá retomar usando o cluster ou os clusters de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console de gerenciamento da AWS.

Para obter detalhes sobre as versões do Aurora MySQL compatíveis com o Amazon Aurora, consulte Notas de lançamento do Aurora MySQL.

Depois disso, é possível aprender como escolher a versão correta do Aurora MySQL para o cluster, como especificar a versão ao criar ou atualizar um cluster e os procedimentos para atualizar um cluster de uma versão para outra com interrupção mínima.

Tópicos

- Conferir os números de versão do Aurora MySQL
- Suporte de longo prazo (LTS) e versões beta do Amazon Aurora MySQL
- Preparação para o fim do suporte padrão do Amazon Aurora edição compatível com o MySQL versão 2
- Preparar para o fim da vida útil do Amazon Aurora, edição compatível com MySQL versão 1
- Como atualizar os clusters de banco de dados de Amazon Aurora MySQL
- Atualizações e correções feitas no mecanismo de banco de dados do Amazon Aurora MySQL

Conferir os números de versão do Aurora MySQL

Embora o Aurora Edição compatível com MySQL seja compatível com os mecanismos de banco de dados MySQL, o Aurora MySQL inclui recursos e correções de bugs que são específicos para versões específicas do Aurora MySQL. Os desenvolvedores de aplicações podem verificar a versão do Aurora MySQL em suas aplicações usando SQL. Os administradores de banco de dados podem verificar e especificar versões do Aurora MySQL ao criar ou atualizar instâncias de banco de dados e clusters de bancos de dados Aurora MySQL

Tópicos

- Verificação ou especificação de versões de mecanismo do Aurora MySQL por meio da AWS
- Verificar versões do Aurora MySQL usando SQL

Verificação ou especificação de versões de mecanismo do Aurora MySQL por meio da AWS

Ao executar tarefas administrativas usando o Console de gerenciamento da AWS, a AWS CLI, ou a API RDS, especifique a versão do Aurora MySQL em um formato alfanumérico descritivo.

Desde o Aurora MySQL versão 2, as versões do mecanismo Aurora têm a seguinte sintaxe.

```
mysql-major-version.mysql_aurora.aurora-mysql-version
```

A porção *mysql-major-version-* é `5.7` ou `8.0`. Esse valor representa a versão do protocolo cliente e nível geral de suporte a recursos MySQL para a versão correspondente do Aurora MySQL.

A *aurora-mysql-version* é um valor pontilhado com três partes: a versão principal do Aurora MySQL, a versão secundária do Aurora MySQL e o nível de patch. A versão principal é `2` ou `3`. Esses valores representam o Aurora MySQL compatível com MySQL 5.7 ou 8.0, respectivamente. A versão secundária representa o lançamento de recursos na série 2.x ou 3.x. O nível de patch começa em `0` para cada versão secundária e representa o conjunto de correções de bugs subsequentes que se aplicam à versão secundária. Ocasionalmente, um novo recurso é incorporado em uma versão secundária, mas não fica visível imediatamente. Nesses casos, o recurso passa por ajustes e é publicado em um nível de patch posterior.

Todas as versões 2.x do mecanismo do Aurora MySQL são compatíveis com o Community MySQL 5.7.12 ou posterior. Todas as versões 3.x do mecanismo Aurora MySQL são compatíveis com o

MySQL 8.0.23 ou posterior. Consulte as notas da versão 3.x específica para encontrar a versão correspondente compatível com o MySQL.

Por exemplo, as versões de mecanismo do Aurora MySQL 3.04.0 e 2.11.2 são as seguintes.

```
8.0.mysql_aurora.3.04.0
5.7.mysql_aurora.2.11.2
```

> **Note**
>
> Não existe uma correspondência individual entre as versões do MySQL da comunidade e as versões 2.x do Aurora MySQL. Para o Aurora MySQL versão 3, o mapeamento é mais direto. Para verificar quais correções de erro e novos recursos estão em uma versão específica do Aurora MySQL, consulte Atualizações no mecanismo de banco de dados do Amazon Aurora MySQL versão 3 e Atualizações no mecanismo de banco de dados do Amazon Aurora MySQL versão 2 nas Notas de lançamento do Aurora MySQL. Para obter uma lista cronológica de novos recursos e lançamentos, consulte Histórico do documento. Para verificar a versão mínima necessária para uma correção relacionada à segurança, consulte Vulnerabilidades de segurança corrigidas no Aurora MySQL em Notas de lançamento do Aurora MySQL.

Especifique a versão do mecanismo Aurora MySQL em alguns comandos da AWS CLI e operações da API do RDS. Por exemplo, especifique a opção `--engine-version` ao executar os comandos da AWS CLI create-db-cluster e modify-db-cluster. Especifique o parâmetro `EngineVersion` ao executar as operações da API do RDS CreateDBCluster e ModifyDBCluster.

No Aurora MySQL versão 2 e posteriores, a versão do mecanismo no Console de gerenciamento da AWS também inclui a versão do Aurora. Atualizar o cluster altera o valor exibido. Essa alteração ajuda a especificar e verificar as versões exatas do Aurora MySQL, sem a necessidade de se conectar ao cluster ou executar os comandos SQL.

> **Tip**
>
> Para clusters do Aurora gerenciados pelo CloudFormation, essa alteração na configuração `EngineVersion` pode acionar ações por meio do CloudFormation. Para obter informações sobre como o CloudFormation trata alterações feitas na configuração `EngineVersion`, consulte a documentação do CloudFormation.

Verificar versões do Aurora MySQL usando SQL

Os números de versão Aurora que você pode recuperar em sua aplicação usando consultas SQL usam o formato *<major version>*.*<minor version>*.*<patch version>*. É possível obter esse número de versão para qualquer instância de banco de dados em seu cluster do Aurora MySQL consultando a variável do sistema `AURORA_VERSION`. Para obter esse número de versão, use uma das seguintes consultas.

```
select aurora_version();
select @@aurora_version;
```

Essas consultas produzem uma saída semelhante à seguinte.

```
mysql> select aurora_version(), @@aurora_version;
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

Os números de versão que o console, CLI e API do RDS retorna usando as técnicas descritas em Verificação ou especificação de versões de mecanismo do Aurora MySQL por meio da AWS geralmente são mais descritivos.

Suporte de longo prazo (LTS) e versões beta do Amazon Aurora MySQL

O Aurora MySQL oferece suporte de longo prazo (LTS) e versões beta para algumas versões do mecanismo do Aurora MySQL.

Tópicos

- Versões de suporte de longo prazo (LTS) do Aurora MySQL
- Versões beta do Aurora MySQL

Versões de suporte de longo prazo (LTS) do Aurora MySQL

Cada nova versão do Aurora MySQL permanece disponível por um determinado período para que você use ao criar ou atualizar um cluster de banco de dados. Após esse período, é necessário atualizar os clusters que usam essa versão. Você pode atualizar manualmente o cluster antes do

término do período de suporte ou o Aurora pode atualizá-lo automaticamente quando a versão do Aurora MySQL não for mais compatível.

O Aurora designa determinadas versões do Aurora MySQL como versões de “suporte de longo prazo” (LTS). Os clusters de banco de dados que usam versões LTS podem permanecer na mesma versão por mais tempo e passar por menos ciclos de atualização que os clusters que usam versões não LTS. Os clusters de banco de dados que usam versões do LTS podem permanecer na mesma versão secundária por pelo menos três anos ou até o final do suporte padrão para a versão principal, o que ocorrer primeiro. Quando é necessário atualizar um cluster de banco de dados que está em uma versão LTS, o Aurora o atualiza para a próxima versão LTS. Dessa forma, o cluster não precisa ser atualizado novamente por um longo tempo.

Durante a vida útil de uma versão LTS do Aurora MySQL, novos níveis de patch apresentam correções em questões importantes. Os níveis de patch não incluem novos recursos. Você pode optar por aplicar ou não esses patches aos clusters de banco de dados que executam a versão LTS. Recomendamos que os clientes que executam versões LTS atualizem para a versão de patch mais recente da versão secundária LTS pelo menos uma vez por ano para aproveitar as correções operacionais e de segurança de alta gravidade. Para determinadas correções críticas, a Amazon pode executar uma atualização gerenciada para um nível de patch na mesma versão LTS. Essas atualizações gerenciadas são realizadas automaticamente na janela de manutenção do cluster. Todas as versões do Aurora MySQL (versões LTS e não LTS) passam por extensos testes operacionais e de estabilidade. Algumas versões secundárias são designadas como versões LTS para permitir que os clientes permaneçam nelas por mais tempo sem atualizar para uma versão secundária mais recente.

Para a maioria dos clusters do Aurora MySQL, recomendamos que você atualize para a versão mais recente, em vez de usar a versão LTS. Isso aproveita o Aurora como um serviço gerenciado e fornece acesso aos recursos e correções de erros mais recentes. As versões LTS destinam-se a clusters com as seguintes características:

- Não é possível permitir tempo de inatividade na aplicação Aurora MySQL para atualizações, exceto para ocorrências raras de patches importantes.
- O ciclo de testes para o cluster e as aplicações associadas exige muito tempo para cada atualização no mecanismo de banco de dados Aurora MySQL.
- A versão do banco de dados para o cluster do Aurora MySQL possui todos os recursos do mecanismo de banco de dados e correções de erros que sua aplicação precisa.

As versões atuais do LTS para Aurora MySQL são as seguintes:

- Aurora MySQL versão 3.10.*.
- Aurora MySQL versão 3.04.*.

Para obter detalhes sobre a versão do LTS, consulte Atualizações no mecanismo de banco de dados do Amazon Aurora MySQL versão 3 nas Notas de lançamento do Aurora MySQL.

> **Note**
>
> Recomendamos que você desative as atualizações automáticas de versões secundárias para versões LTS. Defina o parâmetro `AutoMinorVersionUpgrade` como `false` ou desmarque a caixa de seleção Habilitar o upgrade automático da versão secundária no Console de gerenciamento da AWS.
> Se você não o desabilitar, o cluster de banco de dados poderá ser atualizado para uma versão não LTS.

Versões beta do Aurora MySQL

Uma versão beta do Aurora MySQL é somente uma correção de segurança antecipada em um número limitado de Regiões da AWS. Essas correções serão implantadas posteriormente de forma mais ampla em todas as regiões com a próxima versão do patch.

A numeração de uma versão beta é semelhante à de uma versão secundária do Aurora MySQL, mas com um quarto dígito extra, por exemplo, 2.12.0.1 ou 3.05.0.1.

Para ter mais informações, consulte Database engine updates for Amazon Aurora MySQL version 2 e Database engine updates for Amazon Aurora MySQL version 3 nas Release Notes for Aurora MySQL

Preparação para o fim do suporte padrão do Amazon Aurora edição compatível com o MySQL versão 2

O Amazon Aurora edição compatível com MySQL versão 2 (compatível com MySQL 5.7) está programado para atingir o fim do suporte padrão em 31 de outubro de 2024. Recomendamos que você faça upgrade de todos os clusters que executam o Aurora MySQL versão 2 para o Aurora

MySQL versão 3 padrão (compatível com o MySQL 8.0) ou posterior antes que o Aurora MySQL versão 2 chegue ao fim do período de suporte padrão. Em 31 de outubro de 2024, o Amazon RDS inscreverá automaticamente os bancos de dados no Suporte estendido do Amazon RDS. Se estiver executando o Amazon Aurora MySQL versão 2 (com compatibilidade com o MySQL 5.7) em um cluster do Aurora Serverless versão 1, isso não se aplica a você. Se você quiser fazer upgrade dos clusters da versão 1 do Aurora Serverless para o Aurora MySQL versão 3, consulte Caminho de atualização para clusters de banco de dados do Aurora Serverless v1.

É possível encontrar as próximas datas de fim de suporte das principais versões do Aurora MySQL em Release calendar for Aurora MySQL major versions.

Se tiver clusters executando o Aurora MySQL versão 2, você receberá avisos periódicos com as informações mais recentes sobre como fazer upgrade assim que o suporte padrão chegar próximo ao fim. Atualizaremos esta página periodicamente com as informações mais recentes.

Fim do cronograma de suporte padrão

1. A partir de agora até 31 de outubro de 2024: você pode atualizar clusters do Aurora MySQL versão 2 (com compatibilidade com MySQL 5.7) para o Aurora MySQL versão 3 (com compatibilidade com o MySQL 8.0).
2. 31 de outubro de 2024: nessa data, o suporte padrão da versão 2 do Aurora MySQL chegará ao fim chegará e o Amazon RDS inscreverá automaticamente seus clusters no Suporte estendido do Amazon RDS.

Nós inscreveremos você automaticamente no Suporte estendido do RDS. Para obter mais informações, consulte Suporte estendido do Amazon RDS com Amazon Aurora.

Encontrar clusters afetados por esse processo de fim de vida

Para encontrar clusters afetados por esse processo de fim de vida, use os procedimentos a seguir.

> **Important**
>
> Execute essas instruções em cada Região da AWS e para cada Conta da AWS em que seus recursos estão localizados.

Console

Para encontrar um cluster do Aurora MySQL versão 2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Na caixa Filtrar por bancos de dados, insira 5.7.
4. Verifique o Aurora MySQL na coluna do mecanismo.

AWS CLI

Para encontrar clusters afetados por esse processo de fim de vida usando a AWS CLI, chame o comando describe-db-clusters. Você pode usar seguinte script de exemplo:

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?(Engine==`aurora-
mysql` && contains(EngineVersion,`5.7.mysql_aurora`))].{EngineVersion:EngineVersion,
 DBClusterIdentifier:DBClusterIdentifier, EngineMode:EngineMode}' --output table
 --region us-east-1

        +-----------------------------------------------------------------+
        |                       DescribeDBClusters                        |
        +---------------------+---------------+-------------------------+
        |         DBCI        |      EM       |          EV             |
        +---------------------+---------------+-------------------------+
        |    aurora-mysql2    |  provisioned  | 5.7.mysql_aurora.2.11.3 |
        | aurora-serverlessv1 |   serverless  | 5.7.mysql_aurora.2.11.3 |
        +---------------------+---------------+-------------------------+
```

API do RDS

Para encontrar clusters de banco de dados Aurora MySQL que executam o Aurora MySQL versão 2, use a operação de API DescribeDBClusters do RDS com os seguintes parâmetros obrigatórios.

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nome

    - engine
  - Values.Value.N
    - ['aurora']

Suporte estendido do Amazon RDS

Você pode usar o Suporte estendido do Amazon RDS sobre o MySQL 5.7 da comunidade sem nenhum custo até a data de fim do suporte, 31 de outubro de 2024. Em 31 de outubro de 2024, o Amazon RDS inscreve automaticamente os bancos de dados no Suporte estendido do RDS para o Aurora MySQL versão 2. O Suporte estendido do RDS para Aurora é um serviço pago que fornece até 28 meses adicionais de suporte para o Aurora MySQL versão 2 até o final do Suporte estendido do RDS em fevereiro de 2027. O RDS Extended Support só será oferecido para as versões secundárias 2.11 e 2.12 do Aurora MySQL. Para usar o Amazon Aurora MySQL versão 2 após o final do suporte padrão, planeje executar os bancos de dados em uma dessas versões secundárias antes de 31 de outubro de 2024.

Para ter mais informações sobre o Suporte estendido do RDS, como cobranças e outras considerações, consulte Suporte estendido do Amazon RDS com Amazon Aurora.

Fazer um upgrade

A atualização entre versões principais requer planejamento e teste mais extensos do que para uma versão secundária. O processo pode levar um tempo relevante. Queremos ver a atualização como um processo de três etapas, com atividades antes da atualização, para a atualização e depois da atualização.

Antes da atualização:

Recomendamos conferir a compatibilidade da aplicação, procedimentos de manutenção e considerações semelhantes para o cluster atualizado, ou seja, confirmar se as aplicações funcionarão conforme o esperado após a atualização. Aqui estão cinco recomendações que ajudarão você a ter uma melhor experiência de atualização.

- Primeiro, é fundamental entender Como funciona a atualização da versão principal no local Aurora MySQL.
- Depois, explore as técnicas de upgrade disponíveis quando Atualizar o Aurora MySQL versão 2 para a versão 3.

- Para ajudar você a decidir o momento certo e a abordagem apropriada para upgrade, conheça as diferenças entre o Aurora MySQL versão 3 e seu ambiente atual com Comparar o Aurora MySQL versão 2 e o Aurora MySQL versão 3.
- Depois de escolher a opção mais conveniente e que funciona melhor, experimente uma atualização simulada no local em um cluster clonado, usando Planejando uma atualização de versão principal para um cluster de Aurora MySQL.
- Revise as Verificações prévias de atualização da versão principal do Aurora MySQL. O pré-verificador de atualização pode ser executado e determinar se o banco de dados pode ser atualizado com êxito e se haverá problemas de incompatibilidade da aplicação após a atualização, bem como de desempenho, procedimentos de manutenção e considerações semelhantes.
- Nem todos os tipos ou versões de clusters do Aurora MySQL podem usar o mecanismo de atualização local. Para obter mais informações, consulte Processo de atualização da versão principal de Aurora MySQL.

Em caso de dúvidas ou preocupações, a equipe de suporte da AWS está disponível por meio dos fóruns da comunidade e do Premium Support.

Fazer a atualização:

É possível usar uma das técnicas de atualização a seguir. A quantidade de tempo de inatividade que o sistema enfrentará depende da técnica de atualização escolhida.

- Implantações azul/verde: para situações em que a principal prioridade é reduzir o tempo de inatividade da aplicação, você também pode usar implantações azul/verde do Amazon RDS para fazer upgrade da versão principal em clusters de banco de dados provisionados do Amazon Aurora. Uma implantação azul/verde cria um ambiente de teste que copia o ambiente de produção. Você pode fazer determinadas alterações no cluster de banco de dados do Aurora no ambiente verde (preparação) sem afetar as workloads de produção. A transição normalmente leva menos de um minuto, sem perda de dados. Para obter mais informações, consulte Visão geral das implantações azul/verde do Amazon Aurora. Isso minimiza o tempo de inatividade, mas exige que você execute recursos adicionais durante a atualização.
- Upgrades no local: é possível realizar um upgrade no local no qual o Aurora executa automaticamente um processo de pré-verificação para você, coloca o cluster off-line, faz backup do cluster, faz upgrade e coloca o cluster novamente on-line. Um upgrade no local da versão principal pode ser feito com alguns cliques e não requer outra coordenação nem failovers com outros clusters, mas envolve tempo de inatividade. Para obter mais informações, consulte . Como realizar uma atualização local

- Restauração de snapshot: você pode fazer upgrade do cluster do Aurora MySQL versão 2 com a restauração de um snapshot do Aurora MySQL versão 2 em um cluster do Aurora MySQL versão 3. O processo consiste em tirar um snapshot e depois restaurar. Esse processo envolve a interrupção do banco de dados, pois você está fazendo a restauração com base em um snapshot.

Após a atualização:

Após a atualização, você precisa monitorar atentamente o sistema (aplicação e banco de dados) e, se necessário, fazer ajustes. Seguir de perto as etapas de pré-atualização minimizará as alterações necessárias. Para obter mais informações, consulte Solucionar problemas de desempenho do banco de dados do Amazon Aurora MySQL.

Para saber mais sobre os métodos, o planejamento, o teste e a solução de problemas dos upgrades da versão principal do Aurora MySQL, leia Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL atentamente, incluindo Solução de problemas para atualização no local de Aurora MySQL. Observe também que alguns tipos de instâncias não são compatíveis com o Aurora MySQL versão 3. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Caminho de atualização para clusters de banco de dados do Aurora Serverless v1

A atualização entre versões principais requer planejamento e teste mais extensos do que para uma versão secundária. O processo pode levar um tempo relevante. Queremos ver a atualização como um processo de três etapas, com atividades antes da atualização, para a atualização e depois da atualização.

O Aurora MySQL versão 2 (compatível com o MySQL 5.7) continuará recebendo o suporte padrão para clusters do Aurora Serverless v1.

Se você quiser fazer a atualização do Amazon Aurora MySQL 3 (compatível com o MySQL 8) e continuar executando o Aurora Serverless, você pode usar o Amazon Aurora Serverless v2. Para entender as diferenças entre Aurora Serverless v1 e Aurora Serverless v2, consulte Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1.

Upgrade para Aurora Serverless v2: é possível atualizar um cluster do Aurora Serverless v1 para o Aurora Serverless v2. Para obter mais informações, consulte Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.

Preparar para o fim da vida útil do Amazon Aurora, edição compatível com MySQL versão 1

A edição compatível com o Amazon Aurora MySQL versão 1 (com compatibilidade com MySQL 5.6) está planejada para atingir o fim da vida útil em 28 de fevereiro de 2023. A Amazon aconselha que você atualize todos os clusters (provisionados e do Aurora Serverless) que executam o Aurora MySQL versão 1 para o Aurora MySQL versão 2 (compatível com o MySQL 5.7) ou o Aurora MySQL versão 3 (compatível com o MySQL 8.0). Faça isso antes que o Aurora MySQL versão 1 atinja o final do período de suporte.

Para clusters de banco de dados provisionados do Aurora, você pode concluir atualizações do Aurora MySQL versão 1 para o Aurora MySQL versão 2 por vários métodos. Instruções para o mecanismo de upgrade no local podem ser encontradas noComo realizar uma atualização local. Outra maneira de concluir o upgrade é criar um snapshot de um cluster do Aurora MySQL versão 1 e restaurá-lo para um cluster do Aurora MySQL versão 2. Ou você pode seguir um processo de várias etapas que executa clusters antigos e novos lado a lado. Para obter mais detalhes sobre cada método, consulte Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL.

Para clusters de banco de dados do Aurora Serverless v1, você pode realizar uma atualização no local do Aurora MySQL versão 1 para o Aurora MySQL versão 2. Para obter mais detalhes sobre esse método, consulte Modificar um cluster de banco de dados do Aurora Serverless v1

Para clusters de banco de dados provisionados do Aurora, você pode concluir atualizações do Aurora MySQL versão 1 para o Aurora MySQL versão 3 usando um processo de atualização de dois estágios:

1. Atualize o Aurora MySQL versão 1 para o Aurora MySQL versão 2 com os métodos descritos anteriormente.
2. Atualize o Aurora MySQL versão 2 para o Aurora MySQL versão 3 com os mesmos métodos utilizados para atualizar a versão 1 para a versão 2. Para obter mais detalhes, consulte Atualizar o Aurora MySQL versão 2 para a versão 3. Anote o Diferenças de recursos entre as versões 2 e 3 do Aurora MySQL.

Você pode encontrar as próximas datas de fim de vida útil para as principais versões do Aurora em Versões do Amazon Aurora. A Amazon atualiza automaticamente todos os clusters que você não atualiza antes da data de fim da vida útil. Após a data de fim da vida útil, essas atualizações

automáticas para a versão principal subsequente ocorrerão durante uma janela de manutenção agendada para clusters.

A seguir são apresentados marcos adicionais para atualizar clusters do Aurora MySQL versão 1 (provisionados e do Aurora Serverless) que estão chegando ao fim da vida útil. Para cada uma, a hora de início é 00:00, Horário Universal Coordenado (UTC).

1. Agora até 28 de fevereiro de 2023 : você pode, a qualquer momento, iniciar atualizações dos clusters do Aurora MySQL versão 1 (com compatibilidade com MySQL 5.6) para o Aurora MySQL versão 2 (com compatibilidade com o MySQL 5.7). A partir do Aurora MySQL versão 2, você pode atualizar para o Aurora MySQL versão 3 (com compatibilidade com o MySQL 8.0) para clusters de banco de dados provisionados do Aurora.
2. 16 de janeiro de 2023: após esse período, você não pode criar clusters nem instâncias do Aurora MySQL versão 1 a partir do Console de gerenciamento da AWS ou da AWS Command Line Interface (AWS CLI). Também não é possível adicionar várias regiões secundárias a um Aurora Global Database. Isso pode afetar sua capacidade de se recuperar de uma paralisação não planejada, conforme descrito em Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada, porque você não pode concluir as etapas 5 e 6 após esse período. Você também não poderá criar uma réplica de leitura entre regiões executando o Aurora MySQL versão 1. Você ainda pode fazer o seguinte para clusters existentes do Aurora MySQL versão 1 até 28 de fevereiro de 2023:
   - Crie um snapshot de um cluster do Aurora MySQL versão 1 para a mesma versão do cluster do snapshot de um cluster original.
   - Adicionar réplicas de leitura (não aplicável para clusters de banco de dados do Aurora Serverless).
   - Altere a configuração da instância.
   - Realize restauração em um ponto anterior no tempo.
   - Crie clones de clusters existentes da versão 1.
   - Crie uma réplica de leitura entre regiões executando o Aurora MySQL versão 2 ou superior.
3. 28 de fevereiro de 2023: após esse período, planejamos atualizar automaticamente os clusters da versão 1 do Aurora MySQL para a versão padrão do Aurora MySQL versão 2 dentro de uma janela de manutenção agendada a seguir. A restauração de snapshots de banco de dados Aurora MySQL versão 1 resulta em uma atualização automática do cluster restaurado para a versão padrão do Aurora MySQL versão 2 naquele momento.

A atualização entre versões principais requer planejamento e teste mais extensos do que para uma versão secundária. O processo pode levar um tempo relevante.

Para situações em que a principal prioridade é reduzir o tempo de inatividade, você também pode usar implantações azul/verde para realizar a atualização da versão principal em clusters de banco de dados do Amazon Aurora provisionados. Uma implantação azul/verde cria um ambiente de teste que copia o ambiente de produção. Você pode fazer alterações no cluster de banco de dados do Aurora no ambiente verde (preparação) sem afetar as workloads de produção. A transição normalmente leva menos de um minuto, sem perda de dados e sem necessidade de alterações na aplicação. Para ter mais informações, consulte Visão geral das implantações azul/verde do Amazon Aurora.

Após a conclusão da atualização, também pode ser necessário fazer um trabalho de acompanhamento. Por exemplo, pode ser necessário um acompanhamento devido a diferenças na compatibilidade SQL, da forma que certos recursos relacionados ao MySQL funcionem ou das configurações de parâmetros entre as versões antiga e a nova.

Para saber mais sobre os métodos, o planejamento, o teste e a solução de problemas das atualizações da versão principal do Aurora MySQL, leia Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL atentamente.

Encontrar clusters afetados por esse processo de fim de vida

Para encontrar clusters afetados por esse processo de fim de vida, use os procedimentos a seguir.

> **Important**
>
> Execute essas instruções em cada Região da AWS e para cada Conta da AWS em que seus recursos estão localizados.

Console

Para encontrar um cluster do Aurora MySQL versão 1

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Na caixa Filter by databases (Filtrar por bancos de dados), insira 5.6.
4. Verifique o Aurora MySQL na coluna do mecanismo.

AWS CLI

Para encontrar clusters afetados por esse processo de fim de vida usando a AWS CLI, chame o comando describe-db-clusters. Você pode usar seguinte script de exemplo:

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?Engine==`aurora`].
{EV:EngineVersion, DBCI:DBClusterIdentifier, EM:EngineMode}' --output table --region
 us-east-1

        +-----------------------------------------+
        |            DescribeDBClusters           |
        +---------------+--------------+----------+
        |     DBCI      |     EM       |    EV    |
        +---------------+--------------+----------+
        |  my-database-1|  serverless  |  5.6.10a |
        +---------------+--------------+----------+
```

API do RDS

Para encontrar clusters de banco de dados Aurora MySQL que executam o Aurora MySQL versão 1, use a operação de API DescribeDBClusters do RDS com os seguintes parâmetros obrigatórios.

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nome
      - engine
    - Values.Value.N
      - ['aurora']

Como atualizar os clusters de banco de dados de Amazon Aurora MySQL

Você pode atualizar um cluster de bancos de dados Aurora MySQL para obter correções de erros, novos recursos de Aurora MySQL ou alterar para uma versão totalmente nova do mecanismo de banco de dados subjacente. As seções a seguir mostram como:

> **Note**
>
> O tipo de atualização que você faz depende de quanto tempo de inatividade paga para o cluster, quanto testes de verificação planeja fazer, a importância das correções de erros específicas ou novos recursos para o seu caso de uso e se planeja fazer atualizações pequenas frequentes ou atualizações ocasionais que ignora, várias versões intermediárias. Para cada atualização, você pode alterar a versão principal, a versão secundária e o nível de patch para o cluster. Se você não estiver familiarizado com a distinção entre versões principais, versões secundárias e níveis de patch de Aurora MySQL, você pode ler as informações anteriores em Conferir os números de versão do Aurora MySQL.

> **Tip**
>
> Você pode minimizar o tempo de inatividade necessário para a atualização do cluster de banco de dados usando uma implantação azul/verde. Para ter mais informações, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Tópicos

- Atualizando a versão secundária ou o nível de patch de um cluster de banco de dados de Aurora MySQL
- Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL

Atualizando a versão secundária ou o nível de patch de um cluster de banco de dados de Aurora MySQL

É possível usar os seguintes métodos para atualizar a versão secundária de um cluster de banco de dados ou para aplicar um patch em um cluster de banco de dados:

- Atualizar o Aurora MySQL modificando a versão do mecanismo (para Aurora MySQL versões 2 e 3)
- Habilitar atualizações automáticas entre versões secundárias do Aurora MySQL

Para obter informações sobre como a aplicação de patches sem tempo de inatividade pode reduzir interrupções durante o processo de atualização, consulte Como usar os patches com tempo de inatividade zero.

Para ter informações sobre como realizar a atualização da versão secundária do cluster de banco de dados do Aurora MySQL, consulte os tópicos a seguir.

Tópicos

- Antes de realizar um upgrade da versão secundária
- Pré-verificações de atualização de versão secundária do Aurora MySQL
- Atualizar o Aurora MySQL modificando a versão do mecanismo
- Habilitar atualizações automáticas entre versões secundárias do Aurora MySQL
- Como usar os patches com tempo de inatividade zero
- Técnica alternativa de atualização azul-verde

Antes de realizar um upgrade da versão secundária

Recomendamos que você execute as seguintes ações para reduzir o tempo de inatividade durante um upgrade de versão secundária:

- A manutenção do cluster de banco de dados do Aurora deve ser realizada durante um período de baixo tráfego. Use o Insights de Performance para identificar esses períodos a fim de configurar as janelas de manutenção corretamente. Consulte mais informações sobre o Insights de Performance em Monitoring DB load with Performance Insights on Amazon RDS. Consulte mais informações sobre a janela de manutenção do cluster de banco de dados em Ajustar a janela de manutenção do cluster de banco de dados preferencial.
- Use SDKs da AWS compatíveis com oscilações e recuos exponenciais como uma prática recomendada. Consulte mais informações em Exponential Backoff And Jitter.

Pré-verificações de atualização de versão secundária do Aurora MySQL

Quando você inicia uma atualização de versão secundária, o Amazon Aurora executa pré-verificações automaticamente.

Essas pré-verificações são obrigatórias. Você não pode optar por ignorá-las. As pré-verificações fornecem os seguintes benefícios:

- Elas permitem evitar o tempo de inatividade não planejado durante a atualização.
- Se houver incompatibilidades, o Amazon Aurora impedirá a atualização e fornecerá um log para que você saiba sobre elas. Dessa forma, você poderá usar o log para preparar o banco de dados para o upgrade e reduzir essas incompatibilidades. Consulte informações detalhadas sobre como remover incompatibilidades em Preparing your installation for upgrade na documentação do MySQL.

As pré-verificações são executadas antes que a instância de banco de dados seja interrompida para a atualização, o que significa que elas não causam nenhum tempo de inatividade quando são executadas. Se as verificações prévias encontrarem uma incompatibilidade, o Aurora cancelará automaticamente a atualização antes que a instância de banco de dados seja interrompida. O Aurora também gera um evento para a incompatibilidade. Para ter mais informações sobre eventos do Amazon Aurora, consulte Trabalhar com a notificação de eventos do Amazon RDS.

O Aurora registra informações detalhadas sobre cada incompatibilidade no arquivo de log `PrePatchCompatibility.log`. Na maioria dos casos, a entrada de log inclui um link para a documentação do MySQL para corrigir a incompatibilidade. Para obter mais informações sobre como exibir arquivos de log, consulte Como visualizar e listar arquivos de log do banco de dados.

Devido à natureza das pré-verificações, eles analisam os objetos do seu banco de dados. Essa análise resulta no consumo do recurso e aumenta o tempo para que a atualização seja concluída.

Atualizar o Aurora MySQL modificando a versão do mecanismo

O upgrade da versão secundária de um cluster do Aurora MySQL aplica correções adicionais e novos recursos a um cluster existente.

Esse tipo de upgrade se aplica a clusters do Aurora MySQL em que tanto a versão original como a versão atualizada têm a mesma versão principal do Aurora MySQL, a 2 ou a 3. O processo é rápido e direto porque não envolve uma conversão para os metadados do Aurora MySQL ou a reorganização de seus dados de tabela.

Execute esse tipo de atualização modificando a versão do mecanismo do cluster de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI, ou a API do RDS. Por exemplo, se o cluster estiver executando o Aurora MySQL 3.x, escolha uma versão 3.x posterior.

Se você estiver executando uma atualização secundária em um Aurora Global Database, atualize todos os clusters secundários antes de atualizar o cluster primário.

> **Note**
>
> Para atualizar a versão secundária para o Aurora MySQL versão 3.04.* ou posterior ou a versão 2.12.*, use o seguinte processo:
>
> 1. Remova todas as regiões secundárias do cluster global. Siga as etapas em Remover um cluster de um banco de dados global do Amazon Aurora.
> 2. Atualize a versão do mecanismo da região primária para a versão 3.04.* ou posterior, ou versão 2.12.*, conforme aplicável. Siga as etapas em To modify the engine version of a DB cluster.
> 3. Adicione regiões secundárias ao cluster global. Siga as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Como modificar a versão do mecanismo de um cluster de banco de dados

- Usando o console: modifique as propriedades de seu cluster. Na janela Modify DB cluster (Modificar o cluster de banco de dados), altere a versão do mecanismo do Aurora MySQL na caixa DB engine version (Versão do mecanismo de banco de dados). Se você não estiver familiarizado com o procedimento geral para modificar um cluster, siga as instruções em Modificar o cluster de banco de dados usando o console, a CLI e a API.
- Usando a AWS CLI: chame o comando modify-db-cluster da AWS CLI e especifique o nome do cluster de banco de dados para a opção `--db-cluster-identifier` e a versão do mecanismo para a opção `--engine-version`.

  Por exemplo, para fazer upgrade para o Aurora MySQL versão 3.04.1, defina a opção `--engine-version` como `8.0.mysql_aurora.3.04.1`. Especifique a opção `--apply-immediately` para atualizar imediatamente a versão de banco de dados para o cluster de banco de dados.
- Usando a API do RDS – Call the ModifyDBCluster API operation (Chame a operação de API ModifyDBCluster) e especifique o nome do cluster de banco de dados do parâmetro `DBClusterIdentifier` e a versão do mecanismo do parâmetro `EngineVersion`. Defina o parâmetro `ApplyImmediately` como `true` para atualizar imediatamente a versão do mecanismo do cluster de banco de dados.

Habilitar atualizações automáticas entre versões secundárias do Aurora MySQL

Para um cluster de bancos de dados Amazon Aurora MySQL, você pode especificar que o Aurora atualize o cluster de banco de dados automaticamente para novas versões secundárias. Isso é feito definindo-se a propriedade `AutoMinorVersionUpgrade` (Upgrade automático de versões secundárias no Console de gerenciamento da AWS) do cluster de banco de dados.

Os upgrades automáticos ocorrem durante as janelas de manutenção. Se as instâncias de banco de dados individuais no cluster de banco de dados tiverem janelas de manutenção diferentes da janela de manutenção do cluster, a janela de manutenção do cluster terá precedência.

A atualização automática de versão secundária não se aplica aos seguintes tipos de clusters do Aurora MySQL:

- Clusters que fazem parte de um banco de dados global Aurora
- Clusters que têm réplicas entre regiões

A duração da interrupção varia de acordo com a workload, o tamanho do cluster, a quantidade de dados de log binário e se o Aurora pode usar o recurso de aplicar patches de tempo de inatividade zero (ZDP). O Aurora reinicia o cluster de banco de dados, então você poderá enfrentar um curto período de indisponibilidade antes de retomar o uso do cluster. Em específico, a quantidade de dados do log binário afeta o tempo de recuperação. A instância de banco de dados processa os dados de log binário durante a recuperação. Assim, um alto volume de dados de log binário aumenta o tempo de recuperação.

> **Note**
>
> O Aurora só executará atualizações automáticas se todas as instâncias de banco de dados no cluster de banco de dados tiverem a configuração `AutoMinorVersionUpgrade` habilitada. Para ter informações sobre como defini-la e como ela funciona quando aplicada em níveis de cluster e de instância, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.
> Portanto, se existir um caminho de atualização para as instâncias do cluster de banco de dados para uma versão secundária do mecanismo de banco de dados com o recurso `AutoUpgrade` definido como verdadeiro, a atualização ocorrerá. A configuração de `AutoUpgrade` é dinâmica e definida pelo RDS.
> As atualizações automáticas de versões secundárias são realizadas para a versão secundária padrão.

É possível usar um comando de CLI como o seguinte para conferir o status da configuração `AutoMinorVersionUpgrade` para todas as instâncias de banco de dados em seus clusters do Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Esse comando gerará uma saída semelhante à seguinte:

```
[
  {
      "DBInstanceIdentifier": "db-t2-medium-instance",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-t2-small-original-size",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "instance-2020-05-01-2332",
      "DBClusterIdentifier": "cluster-57-2020-05-01-4615",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

Neste exemplo, a opção Habilitar atualização automática da versão secundária está desativada para o cluster de banco de dados `cluster-57-2020-06-03-6411` porque está desativada para uma das instâncias de banco de dados no cluster.

Como usar os patches com tempo de inatividade zero

A execução de atualizações para clusters de bancos de dados Aurora MySQL envolve a possibilidade de uma interrupção quando o banco de dados é desligado e enquanto ele está sendo atualizado. Por padrão, se você iniciar a atualização enquanto o banco de dados estiver ocupado, perderá todas as conexões e transações que o cluster de banco de dados está processando. Se você esperar até que o banco de dados fique ocioso para realizar a atualização, talvez seja necessário esperar muito tempo.

O recurso de aplicação de patches com tempo de inatividade zero (ZDP) tenta, com o melhor esforço, preservar as conexões do cliente por meio de um upgrade do Aurora MySQL. Se a ZDP for concluída com êxito, as sessões da aplicação serão preservadas, e o mecanismo de banco de dados será reiniciado enquanto o upgrade estiver em andamento. A reinicialização do mecanismo de banco de dados pode causar uma queda na taxa de transferência com duração de alguns segundos a aproximadamente um minuto.

O ZDP não se aplica ao seguinte:

- Patches e upgrades do sistema operacional (SO)
- Atualizações da versão principal

O ZDP está disponível para todas as versões do Aurora MySQL e as classes de instâncias de banco de dados.

O ZDP não é compatível com o Aurora Serverless v1 nem com bancos de dados globais do Aurora.

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Uso de classes de instância T para desenvolvimento e testes.

Você pode ver métricas de atributos importantes durante o ZDP no log de erros do MySQL. Você também pode ver informações sobre quando o Aurora MySQL usa o ZDP ou escolhe não usar o ZDP na página Events (Eventos) no Console de gerenciamento da AWS.

No Aurora MySQL, o Aurora pode executar um patch de tempo de inatividade zero estando ou não habilitada a replicação de logs binários. Se a replicação de logs binários estiver habilitada, o Aurora MySQL descartará automaticamente a conexão com o destino do log binário durante uma operação de ZDP. O Aurora MySQL se reconecta automaticamente ao destino do log binário e retoma a replicação quando a reinicialização é concluída.

O ZDP também funciona em combinação com os aprimoramentos de reinicialização no Aurora MySQL. A aplicação de patches à instância de banco de dados do gravador corrige automaticamente os leitores ao mesmo tempo. Depois de executar o patch, o Aurora restaura as conexões nas instâncias de banco de dados do gravador e do leitor.

A ZDP pode não ser concluída com êxito nas seguintes condições:

- Consultas ou transações de longa execução estão em andamento. Se o Aurora puder realizar a ZDP nesse caso, todas as transações em aberto serão canceladas, mas suas conexões serão retidas.
- Tabelas temporárias, bloqueios de usuário ou de tabela estão em uso; por exemplo, enquanto as declarações da linguagem de definição de dados (DDL) são executadas. O Aurora descarta essas conexões.
- Existem alterações de parâmetros pendentes.

Se nenhuma janela de tempo apropriada para executar a ZDP estiver disponível devido a uma ou mais dessas condições, a aplicação de patch será revertida para o comportamento padrão.

Embora as conexões permaneçam intactas após uma operação ZDP bem-sucedida, algumas variáveis e recursos são reinicializados. Os seguintes tipos de informações não são preservados por meio de uma reinicialização causada pela aplicação de patches com tempo de inatividade zero:

- Variáveis globais. O Aurora restaura variáveis de sessão, mas não restaura variáveis globais após a reinicialização.
- Variáveis de status. Especificamente, o valor do tempo de atividade informado pelo status do mecanismo é redefinido após uma reinicialização que usa os mecanismos ZDR ou ZDP.
- `LAST_INSERT_ID`.
- Estado de `auto_increment` na memória para tabelas. O estado de incremento automático na memória é reinicializado. Para ter mais informações sobre valores de incremento automático, consulte o Guia de referência do MySQL.
- Informações diagnósticas das tabelas `INFORMATION_SCHEMA` e `PERFORMANCE_SCHEMA`. Essas informações de diagnóstico também aparecem na saída de comandos como `SHOW PROFILE` e `SHOW PROFILES`.

As seguintes atividades relacionadas à reinicialização do tempo de inatividade zero são relatadas na página Events (Eventos):

- Tentar atualizar o banco de dados com tempo de inatividade zero.
- Tentar atualizar o banco de dados com tempo de inatividade zero. O evento relata quanto tempo o processo demorou. O evento também informa quantas conexões foram preservadas durante

a reinicialização e quantas conexões foram descartadas. Você pode consultar o log de erros do banco de dados para ver mais detalhes sobre o que aconteceu durante a reinicialização.

Técnica alternativa de atualização azul-verde

Em algumas situações, a prioridade é executar um switchover imediato do cluster antigo para um atualizado. Nessas situações, você pode seguir um processo de várias etapas que executa clusters antigos e novos lado a lado. Nesse caso, você replica dados do cluster antigo para o novo até que esteja pronto para que o novo cluster assuma o controle. Para obter mais detalhes, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Realizar a atualização da versão principal de um cluster de bancos de dados do Amazon Aurora MySQL

Em um número de versão do Aurora MySQL, como 3.04.1, o 3 representa a versão principal. O Aurora MySQL versão 2 é compatível com o MySQL 5.7. O Aurora MySQL versão 3 é compatível com o MySQL 8.0.

A atualização entre versões principais requer planejamento e teste mais extensos do que para uma versão secundária. O processo pode levar um tempo relevante. Após a conclusão da atualização, também pode ser necessário fazer um trabalho de acompanhamento. Por exemplo, isso pode ocorrer devido a diferenças na compatibilidade de SQL ou da forma que determinados recursos relacionados ao MySQL funcionam. Ou isso pode ocorrer devido às diferentes configurações de parâmetros entre as versões antiga e nova.

Sumário

- Atualizar o Aurora MySQL versão 2 para a versão 3
- Processo de atualização da versão principal de Aurora MySQL
- Como funciona a atualização da versão principal no local Aurora MySQL
- Planejando uma atualização de versão principal para um cluster de Aurora MySQL
  - Simular a atualização clonando o cluster de banco de dados
  - Implantações azul/verde
- Verificações prévias de atualização da versão principal do Aurora MySQL
  - Processo de pré-verificação do Aurora MySQL
  - Formato de log da pré-verificação do Aurora MySQL

- Exemplos de saída do log da pré-verificação do Aurora MySQL
- Desempenho da pré-verificação do Aurora MySQL
- Resumo das pré-verificações de atualização do MySQL Community
- Resumo das pré-verificações de atualização do Aurora MySQL
- Referência de descrições da pré-verificação do Aurora MySQL
  - Erros
    - Pré-verificações do MySQL que relatam erros
    - Pré-verificações do Aurora MySQL que relatam erros
  - Avisos
    - Pré-verificações do MySQL que exibem avisos
    - Pré-verificações do Aurora MySQL que exibem avisos
  - Avisos
  - Erros, avisos ou alertas
- Como realizar uma atualização local
  - Como as atualizações locais afetam os grupos de parâmetros de um cluster
  - Alterações nas propriedades do cluster entre as versões do Aurora MySQL
  - Principais atualizações no local para bancos de dados globais
  - Considerações sobre retrocesso
- Tutorial de atualização no local de Aurora MySQL
- Descobrir os motivos das falhas de atualização da versão principal do Aurora MySQL
- Solução de problemas para atualização no local de Aurora MySQL
- Limpeza pós-upgrade do Aurora MySQL versão 3
  - Índices espaciais

Atualizar o Aurora MySQL versão 2 para a versão 3

Se você tiver um cluster compatível com o MySQL 5.7 e quiser atualizá-lo para um cluster compatível com o MySQL 8.0, você poderá realizar um processo de atualização no próprio cluster. Esse tipo de atualização é uma atualização in-loco, em comparação com as atualizações que você faz criando um novo cluster. Esta técnica mantém o mesmo endpoint e outras características do cluster. A atualização é relativamente rápida porque não requer a cópia de todos os seus dados para um novo volume de cluster. Essa estabilidade ajuda a minimizar quaisquer alterações de configuração em

seus aplicativos. Ele também ajuda a reduzir a quantidade de testes para o cluster atualizado. Isso ocorre porque o número de instâncias de banco de dados e suas classes de instância permanecem iguais.

O mecanismo de atualização no local envolve encerrar o cluster de banco de dados enquanto a operação é realizada. O Aurora executa um desligamento limpo e conclui as operações pendentes, como reverter transação e desfazer limpeza. Para ter mais informações, consulte Como funciona a atualização da versão principal no local Aurora MySQL.

O método de atualização no local é conveniente, porque é simples de executar e minimiza as alterações de configuração para aplicações associadas. Por exemplo, uma atualização no local preserva os endpoints e o conjunto de instâncias de banco de dados para o cluster. No entanto, o tempo necessário para uma atualização local pode variar dependendo das propriedades do esquema e a ocupação do cluster. Portanto, dependendo das necessidades do cluster, é possível escolher entre as técnicas de atualização:

- Atualização no local
- Implantação azul/verde
- Restauração por snapshot

> **Note**
>
> Se você usar a AWS CLI ou a API do RDS para o método de atualização de restauração de snapshot, deverá executar uma operação subsequente para criar uma instância de banco de dados do gravador no cluster de banco de dados restaurado.

Para obter informações gerais sobre o Aurora MySQL versão 3 e os novos recursos, consulte Aurora MySQL versão 3 compatível com o MySQL 8.0.

Para obter detalhes sobre o planejamento de uma atualização, consulte Planejando uma atualização de versão principal para um cluster de Aurora MySQL e Como realizar uma atualização local.

Processo de atualização da versão principal de Aurora MySQL

Nem todos os tipos ou versões de clusters do Aurora MySQL podem usar o mecanismo de atualização local. Você pode aprender o caminho de atualização apropriado para cada cluster de Aurora MySQL com base na tabela a seguir.

Tipo de cluster de banco de dados de Aurora MySQL	Pode usar a atualização no local?	Ação
Cluster provisionado do Aurora MySQL, versão 2	Sim	A atualização no local é compatível com clusters do Aurora MySQL compatíveis com o MySQL 5.7. Para obter informações sobre o upgrade para o Aurora MySQL versão 3, consulte Planejando uma atualização de versão principal para um cluster de Aurora MySQL e Como realizar uma atualização local.
Cluster provisionado do Aurora MySQL, versão 3	Não aplicável	Use um procedimento de atualização de versão secundária para atualizar entre as versões 3 do Aurora MySQL.
Aurora Serverless v1Cluster do	Não aplicável	O Aurora Serverless v1 só é compatível com o Aurora MySQL na versão 2.
Aurora Serverless v2Cluster do	Não aplicável	O Aurora Serverless v2 só é compatível com o Aurora MySQL na versão 3.
Cluster em um banco de dados Aurora global	Sim	Para fazer upgrade do Aurora MySQL versão 2 para a versão 3, siga o procedimento para fazer um upgrade no local para clusters em um banco de dados Aurora global. Execute o upgrade no cluster global. O Aurora atualiza automaticamente todos os clusters secundários no banco de dados global ao mesmo tempo. Se você usar AWS CLI ou a API do RDS, chame o comando `modify-global-cluster` ou a operação `ModifyGlobalCluster` em vez de `modify-db-cluster` ou `ModifyDBCluster` . Será possível realizar um upgrade no local do Aurora MySQL versão 2 para a versão 3 somente se o

Tipo de cluster de banco de dados de Aurora MySQL	Pode usar a atualização no local?	Ação
		parâmetro `lower_case_table_names` estiver definido como padrão e você reinicializar o banco de dados global. Para obter mais informações, consulte Atualizações de versão principal.
Cluster de consulta paralela	Sim	Você pode executar um upgrade no local.
Cluster que é o destino da replicação de log binário	Talvez	Se a replicação de log binário for de um cluster do Aurora MySQL, você poderá executar um upgrade no local. Você não poderá executar o upgrade se a replicação de log binário for de uma instância de banco de dados do RDS para MySQL ou de uma instância de banco de dados MySQL on-premises. Nesse caso, você pode atualizar usando o mecanismo de recuperação de snapshot.
Cluster com zero instâncias de banco de dados	Não	Com a AWS CLI ou a API do RDS, você pode criar um cluster de Aurora MySQL sem nenhuma instância de banco de dados anexa. Da mesma forma, você também pode remover todas as instâncias de banco de dados de um cluster do Aurora MySQL, deixando os dados no volume do cluster intactos. Embora um cluster tenha zero instâncias de banco de dados, você não pode executar uma atualização no local. O mecanismo de atualização requer uma instância de gravador no cluster para realizar conversões nas tabelas do sistema, arquivos de dados e assim por diante. Nesse caso, use a AWS CLI ou a API do RDS para criar uma instância de gravador para o cluster. Em seguida, você pode executar uma atualização no local.

Tipo de cluster de banco de dados de Aurora MySQL	Pode usar a atualização no local?	Ação
Cluster com retrocesso habilitado	Sim	É possível executar uma atualização no local para um cluster do Aurora MySQL que usa o recurso de retrocesso. No entanto, após a atualização, você não pode retroceder o cluster para um momento anterior à atualização.

Como funciona a atualização da versão principal no local Aurora MySQL

Aurora MySQL executa uma atualização de versão principal como um processo de vários estágios. Você pode verificar o status atual de uma atualização. Algumas das etapas de atualização também fornecem informações sobre o progresso. Conforme cada etapa começa, Aurora MySQL registra um evento. Você pode analisar eventos conforme ocorrem na página Events (Eventos) no console do RDS. Para ter mais informações sobre como trabalhar com eventos, consulte Trabalhar com a notificação de eventos do Amazon RDS.

> **Important**
>
> Uma vez que o processo começa, é executado até que a atualização seja bem-sucedida ou ocorra uma falha. Você não pode cancelar a atualização enquanto está em andamento. Se houver falha na atualização, Aurora retorna todas as alterações e seu cluster tem a mesma versão do mecanismo, metadados e outros como anteriormente.

O processo de atualização consiste em três etapas:

1. O Aurora realiza uma série de verificações prévias antes de iniciar o processo de atualização. Seu cluster continua em execução enquanto Aurora faz essas verificações. Por exemplo, o cluster não pode ter nenhuma transação XA no estado preparado ou estar processando qualquer instrução DDL (Data Definition Language, linguagem de definição de dados). Por exemplo, você pode precisar desligar aplicativos que estão enviando alguns tipos de instruções SQL. Ou você pode simplesmente esperar até que algumas instruções de longa duração sejam concluídas.

Em seguida, tente a atualização novamente. Algumas verificações testam condições que não impedem a atualização, mas podem fazer a atualização demorar muito tempo.

Se Aurora detectar que quaisquer condições necessárias não foram atendidas, altere as condições identificadas nos detalhes do evento. Siga as orientações em Solução de problemas para atualização no local de Aurora MySQL. Se Aurora detectar condições que podem causar uma atualização lenta, planeje monitorar a atualização durante um período prolongado.

2. Aurora torna o cluster offline. Em seguida, Aurora executa um conjunto semelhante de testes como no estágio anterior, para confirmar que não surgiram novos problemas durante o processo de desligamento. Se nesse momento o Aurora detectar condições que impediriam o upgrade, o Aurora cancelará o upgrade e colocará o cluster de volta online. Neste caso, confirme quando as condições já não se aplicam e comece a atualização novamente.
3. Aurora cria um snapshot do volume do cluster. Suponha que você descubra compatibilidade ou outros tipos de problemas após a conclusão da atualização. Ou suponha que você deseja realizar testes usando os clusters original e atualizado. Nesses casos, você pode restaurar a partir deste snapshot para criar um novo cluster com a versão original do mecanismo e os dados originais.

   > **Tip**
   >
   > Este snapshot é um snapshot manual. No entanto, Aurora pode criar e continuar com o processo de atualização, mesmo que você tenha atingido sua cota para snapshots manuais. Este snapshot permanece estável (se necessário) até que você o exclua. Depois de concluir todos os testes após a atualização, você pode excluir esse snapshot para minimizar as cobranças de armazenamento.

4. Aurora clona o volume do cluster. A clonagem é uma operação rápida que não envolve copiar os dados reais da tabela. Se Aurora encontrar um problema durante a atualização, retornará para os dados originais do volume de cluster clonado e coloca o cluster novamente online. O volume clonado temporário durante a atualização não está sujeito ao limite usual do número de clones para um único volume de cluster.
5. Aurora executa um desligamento limpo para a instância de banco de dados do gravador. Durante o desligamento limpo, os eventos de progresso são registrados a cada 15 minutos para as seguintes operações. Você pode analisar eventos conforme ocorrem na página Events (Eventos) no console do RDS.
   - Aurora limpa os registros de desfazer para versões antigas de linhas.
   - Aurora reverte quaisquer transações não confirmadas.

6. Aurora atualiza a versão do mecanismo na instância de banco de dados do gravador:
   - Aurora instala o binário para a nova versão do mecanismo na instância de banco de dados do gravador.
   - Aurora usa a instância de banco de dados do gravador para atualizar seus dados para o formato compatível com o MySQL 5.7. Durante esse estágio, Aurora modifica as tabelas do sistema e executa outras conversões que afetam os dados no volume do cluster. Em particular, Aurora atualiza os metadados de partição nas tabelas do sistema para serem compatíveis com o formato de partição MySQL 5.7. Esse estágio pode levar muito tempo se as tabelas em seu cluster tiverem muitas partições.

     Se algum erro ocorrer durante este estágio, você pode encontrar os detalhes nos logs de erros do MySQL. Depois de iniciar esse estágio, se o processo de atualização falhar por qualquer motivo, Aurora restaura os dados originais do volume de cluster clonado.
7. Aurora atualiza a versão do mecanismo nas instâncias de banco de dados do leitor.
8. O processo de atualização está concluído. O Aurora registra um evento final para indicar que o processo de atualização foi concluído corretamente. Agora seu cluster de banco de dados está executando a nova versão principal.

Planejando uma atualização de versão principal para um cluster de Aurora MySQL

Para ajudar você a decidir o momento certo e a abordagem apropriada para atualização, conheça as diferenças entre o Aurora MySQL versão 3 e o ambiente atual:

- Se estiver convertendo do RDS para o MySQL 8.0 ou do MySQL 8.0 Community Edition, consulte Comparar o Aurora MySQL versão 3 e o MySQL 8.0 Community Edition.
- Se estiver realizando a atualização do Aurora MySQL versão 2, do RDS para MySQL 5.7 ou do MySQL Community Edition 5.7, consulte Comparar o Aurora MySQL versão 2 e o Aurora MySQL versão 3.
- Crie novas versões compatíveis com o MySQL 8.0 de qualquer grupo de parâmetros personalizado. Aplique todos os valores de parâmetros personalizados necessários aos novos grupos de parâmetros. Consulte Alterações de parâmetros do Aurora MySQL versão 3 para saber mais sobre as alterações nos parâmetros.
- Analise o esquema de banco de dados e as definições de objetos do Aurora MySQL versão 2 para saber sobre o uso das novas palavras-chave reservadas introduzidas no MySQL 8.0 Community Edition. Faça isso antes de fazer a atualização. Para obter mais informações, consulte Novas palavras-chave e palavras reservadas no MySQL 8.0 na documentação do MySQL.

Você também pode encontrar mais dicas e considerações de upgrade específicas do MySQL em Alterações no MySQL 8.0, no Guia de referência do MySQL. Por exemplo, você pode utilizar o comando `mysqlcheck --check-upgrade` para analisar seus bancos de dados existentes do Aurora MySQL e identificar possíveis problemas de upgrade.

> **Note**
>
> Recomendamos o uso de classes maiores de instâncias de banco de dados ao atualizar para o Aurora MySQL versão 3 utilizando a atualização no local ou a técnica de restauração de snapshot. Os exemplos são db.r5.24xlarge e db.r6g.16xlarge. Isso ajuda o processo de atualização a ser concluído mais rapidamente usando a maior parte da capacidade de CPU disponível na instância de banco de dados. Você pode mudar para a classe da instância de banco de dados que deseja após a conclusão da atualização da versão principal.

Depois de concluir a atualização em si, você pode seguir os procedimentos pós-upgrade no Limpeza pós-upgrade do Aurora MySQL versão 3. Por fim, teste a funcionalidade e a performance da aplicação.

Se você estiver convertendo do RDS do MySQL ou MySQL Community Edition, siga o procedimento de migração explicado em Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL. Em alguns casos, é possível utilizar a replicação de logs binários para sincronizar seus dados com um cluster do Aurora MySQL versão 3 como parte da migração. Nesse caso, o sistema de origem deve executar uma versão compatível com o cluster de banco de dados de destino.

Para garantir que suas aplicações e procedimentos administrativos funcionem sem problemas após a atualização de um cluster entre versões principais, faça um planejamento e preparação antecipados. Para visualizar os tipos de código de gerenciamento para atualizar seus scripts da AWS CLI ou aplicações com base na API do RDS, consulte Como as atualizações locais afetam os grupos de parâmetros de um cluster. Consulte também Alterações nas propriedades do cluster entre as versões do Aurora MySQL.

Para conhecer os problemas que você pode encontrar durante a atualização, consulte Solução de problemas para atualização no local de Aurora MySQL. Para problemas que levem mais tempo para atualização, você pode testar essas condições com antecedência e corrigi-las.

> **Note**
>
> A atualização no local requer que o cluster de banco de dados fique desligado enquanto a operação é realizada. O Aurora MySQL executa um encerramento limpo e conclui as operações pendentes, como uma limpeza do tipo desfazer. Uma atualização poderá levar muito tempo se houver necessidade de limpar muitos registros de desfazer. Recomendamos realizar a atualização somente após a redução do tamanho da lista de histórico (HLL). Um valor geralmente aceitável para o HLL é 100 mil ou menos. Veja este post de blog para obter mais informações.

Simular a atualização clonando o cluster de banco de dados

Você pode conferir a compatibilidade da aplicação, a performance, procedimentos de manutenção e considerações semelhantes para o cluster atualizado. Para fazer isso, você pode realizar uma simulação da atualização antes do procedimento real. Esta técnica pode ser especialmente útil para clusters de produção. Aqui, é importante minimizar o tempo de inatividade e ter o cluster atualizado pronto para funcionar assim que a atualização for concluída.

Use as seguintes etapas:

1. Crie um clone do cluster original. Siga o procedimento em Clonar um volume para um cluster de banco de dados do Amazon Aurora.
2. Configure um conjunto semelhante de instâncias de banco de dados de gravador e leitor como no cluster original.
3. Execute uma atualização no local do cluster clonado. Siga o procedimento em Como realizar uma atualização local.

   Inicie a atualização imediatamente após criar o clone. Dessa forma, o volume do cluster ainda é idêntico ao estado do cluster original. Se o clone ficar ocioso antes de fazer a atualização, Aurora executa processos de limpeza do banco de dados em segundo plano. Nesse caso, a atualização do clone não é uma simulação precisa de atualizar o cluster original.
4. Teste a compatibilidade de aplicações, performance, procedimentos de administração e assim por diante usando o cluster clonado.
5. Se você encontrar algum problema, ajuste seus planos de atualização para contabilizá-los. Por exemplo, adapte qualquer código de aplicação para ser compatível com o conjunto de recursos da versão posterior. Faça uma estimativa da atualização que provavelmente demorará com base

na quantidade de dados no cluster. Você também pode optar por agendar a atualização para um momento em que o cluster não está ocupado.

6. Quando suas aplicações e workloads funcionarem corretamente com o cluster de teste, você poderá realizar a atualização no local para o cluster de produção.
7. Trabalhe para minimizar o tempo de inatividade total do cluster durante uma atualização de versão principal. Para fazer isso, a workload no cluster deve ser baixa ou zero no momento da atualização. Em particular, certifique-se de que não há transações de longa duração em andamento quando iniciar a atualização.

Implantações azul/verde

Em algumas situações, a prioridade é executar um switchover imediato do cluster antigo para um atualizado. Nessas situações, você pode seguir um processo de várias etapas que executa clusters antigos e novos lado a lado. Nesse caso, você replica dados do cluster antigo para o novo até que esteja pronto para que o novo cluster assuma o controle. Para obter mais detalhes, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Verificações prévias de atualização da versão principal do Aurora MySQL

A atualização do MySQL de uma versão principal para outra, como passar do MySQL 5.7 para o MySQL 8.0, envolve algumas mudanças de arquitetura significativas que exigem planejamento e preparação cuidadosos. Ao contrário de atualizações de versões secundárias, nos quais o foco está principalmente na atualização do software do mecanismo de banco de dados e, em alguns casos, das tabelas do sistema, as atualizações do MySQL principal geralmente introduzem mudanças fundamentais na forma como o banco de dados armazena e gerencia os metadados.

Para ajudar você a identificar essas incompatibilidades, ao atualizar o Aurora MySQL versão 2 para a versão 3, o Aurora executa automaticamente verificações de compatibilidade de atualização (pré-verificações) para examinar objetos no cluster de banco de dados e identificar incompatibilidades conhecidas que podem impedir que a atualização continue. Consulte detalhes sobre as pré-verificações do Aurora MySQL em Referência de descrições da pré-verificação do Aurora MySQL. As pré-verificações do Aurora são executadas complementarmente àquelas executadas pelo utilitário de verificação de atualização do MySQL Community.

Essas pré-verificações são obrigatórias. Você não pode optar por ignorá-las. As pré-verificações oferecem os seguintes benefícios:

- Elas podem reduzir a possibilidade de falhas de atualização capazes de prolongar o tempo de inatividade.
- Se houver incompatibilidades, o Amazon Aurora impedirá que a atualização continue e fornecerá um log para que você saiba mais sobre elas. Você pode então resolver essas incompatibilidades usando o log para preparar o banco de dados para a atualização para a versão 3. Consulte informações detalhadas sobre como resolver incompatibilidades em Preparing Your Installation for Upgrade na documentação do MySQL e em Upgrading to MySQL 8.0? Here is what you need to know... no blog do MySQL Server.

  Para ter informações sobre como atualizar para o MySQL 8.0, consulte Upgrading MySQL na documentação do MySQL.

As pré-verificações são executadas antes que o cluster de banco de dados fique off-line para a atualização da versão principal. Se as verificações prévias encontrarem uma incompatibilidade, o Aurora cancelará automaticamente a atualização antes que a instância de banco de dados seja interrompida. O Aurora também gera um evento para a incompatibilidade. Para ter mais informações sobre eventos do Amazon Aurora, consulte Trabalhar com a notificação de eventos do Amazon RDS.

Após a conclusão das pré-verificações, o Aurora registra informações detalhadas sobre cada incompatibilidade no arquivo `upgrade-prechecks.log`. Na maioria dos casos, a entrada de log inclui um link para a documentação do MySQL para corrigir a incompatibilidade. Para obter mais informações sobre como exibir arquivos de log, consulte Como visualizar e listar arquivos de log do banco de dados.

> **Note**
>
> Devido à natureza das pré-verificações, eles analisam os objetos do seu banco de dados. Essa análise resulta no consumo do recurso e aumenta o tempo para que a atualização seja concluída. Consulte mais informações sobre as considerações de desempenho da pré-verificação em Processo de pré-verificação do Aurora MySQL.

Sumário

- Processo de pré-verificação do Aurora MySQL
- Formato de log da pré-verificação do Aurora MySQL
- Exemplos de saída do log da pré-verificação do Aurora MySQL
- Desempenho da pré-verificação do Aurora MySQL
- Resumo das pré-verificações de atualização do MySQL Community
- Resumo das pré-verificações de atualização do Aurora MySQL
- Referência de descrições da pré-verificação do Aurora MySQL
  - Erros
    - Pré-verificações do MySQL que relatam erros
    - Pré-verificações do Aurora MySQL que relatam erros
  - Avisos
    - Pré-verificações do MySQL que exibem avisos
    - Pré-verificações do Aurora MySQL que exibem avisos
  - Avisos
  - Erros, avisos ou alertas

Processo de pré-verificação do Aurora MySQL

Conforme descrito anteriormente, o processo de atualização do Aurora MySQL exige a execução de verificações de compatibilidade (pré-verificações) no banco de dados para que a atualização da versão principal possa continuar.

Para atualizações no local, as pré-verificações são executadas na instância de banco de dados do gravador enquanto ela está on-line. Se a pré-verificação for bem-sucedida, a atualização prosseguirá. Se forem encontrados erros, eles serão registrados no arquivo `upgrade-prechecks.log` e a atualização será cancelada. Antes de tentar fazer a atualização novamente, resolva os erros exibidos no arquivo `upgrade-prechecks.log`.

Quanto a atualizações de restauração de snapshots, a pré-verificação é executada durante o processo de restauração. Se ela for bem-sucedida, o banco de dados será atualizado para a nova versão do Aurora MySQL. Se forem encontrados erros, eles serão registrados no arquivo `upgrade-prechecks.log` e a atualização será cancelada. Antes de tentar fazer a atualização novamente, resolva os erros exibidos no arquivo `upgrade-prechecks.log`.

Para obter mais informações, consulte Descobrir os motivos das falhas de atualização da versão principal do Aurora MySQL e Referência de descrições da pré-verificação do Aurora MySQL.

Para monitorar o status da pré-verificação, é possível visualizar os eventos a seguir no cluster de banco de dados.

Status da pré-verificação	Mensagem do evento	Ação
Started	Preparação da atualização em andamento: iniciando pré-verif icações de atualização on-line.	Nenhum
Failed	O cluster de banco de dados está em um estado que não permite atualização: as pré-verificações de atualizaç ão falharam. Consulte mais detalhes no arquivo upgrade-p rechecks.log. Consulte mais informações sobre a solução de problemas	Verifique `upgrade-p rechecks.log` para ver se há erros. Corrija os erros. Tente fazer a atualização novamente.

Status da pré-verificação	Mensagem do evento	Ação
	da causa da falha na atualizaç ão em https://docs.aws.amazon.co m/AmazonRDS/latest/ AuroraUserGuide/Aurora MySQL.Upgrading.Tr oubleshooting.html	
Bem-sucedida	Preparação da atualização em andamento: pré-verif icações de atualização on-line concluídas.	A pré-verificação foi bem-sucedida sem exibir nenhum erro. Examine os avisos e alertas em `upgrade-prechecks. log` .

Consulte mais informações sobre a visualização de eventos em Visualizar eventos do Amazon RDS.

Formato de log da pré-verificação do Aurora MySQL

Depois que as verificações de compatibilidade da atualização (pré-verificações) forem concluídas, você poderá examinar o arquivo `upgrade-prechecks.log`. O arquivo de log contém os resultados, os objetos afetados e as informações de correção de cada pré-verificação.

Os erros impedem a atualização. Você deve resolvê-los antes de tentar fazer a atualização novamente.

Os avisos e alertas são menos importantes, mas ainda assim recomendamos que você os examine cuidadosamente para garantir que não haja problemas de compatibilidade com a workload da aplicação. Resolva todos os problemas identificados o quanto antes.

O arquivo de log tem o seguinte formato:

- `targetVersion`: a versão compatível com MySQL da atualização do Aurora MySQL.
- `auroraServerVersion`: a versão do Aurora MySQL na qual a pré-verificação foi executada.

- `auroraTargetVersion`: a versão do Aurora MySQL para a qual você está fazendo a atualização.
- `checksPerformed`: contém a lista de pré-verificações realizadas.
- `id`: o nome da pré-verificação que está sendo executada.
- `title`: uma descrição da pré-verificação que está sendo executada.
- `status`: isso não indica se a pré-verificação foi bem-sucedida ou falhou, mas mostra o status da consulta de pré-verificação:
  - `OK`: a consulta de pré-verificação foi executada e concluída com êxito.
  - `ERROR`: a consulta de pré-verificação falhou ao ser executada. Isso pode ocorrer devido a problemas como restrições de recursos, reinicializações inesperadas da instância ou interrupção da consulta de pré-verificação de compatibilidade.

    Consulte mais informações neste exemplo.
- `description`: uma descrição geral da incompatibilidade e informações de como corrigir o problema.
- `documentationLink`: quando aplicável, um link para a documentação relevante do Aurora MySQL ou do MySQL é indicado aqui. Para obter mais informações, consulte Referência de descrições da pré-verificação do Aurora MySQL.
- `detectedProblems`: se a pré-verificação exibir um erro, aviso ou alerta, isso mostrará os detalhes da incompatibilidade e os objetos incompatíveis, quando aplicável:
  - `level`: o nível da incompatibilidade detectado pela pré-verificação. Os níveis válidos são os seguintes:
    - `Error`: a atualização não poderá continuar enquanto você não resolver a incompatibilidade.
    - `Warning`: a atualização pode continuar, mas um objeto, uma sintaxe ou uma configuração obsoletos foram detectados. Examine os avisos cuidadosamente e resolva-os o quanto antes para evitar problemas em versões futuras.
    - `Notice`: a atualização pode continuar, mas um objeto, uma sintaxe ou uma configuração obsoletos foram detectados. Examine os alertas cuidadosamente e resolva-os o quanto antes para evitar problemas em versões futuras.
  - `dbObject`: o nome do objeto do banco de dados no qual a incompatibilidade foi detectada.
  - `description`: uma descrição detalhada da incompatibilidade e informações de como corrigir o problema.
- `errorCount`: o número de erros de incompatibilidade detectados. Eles impedem a atualização.

- `warningCount`: o número de avisos de incompatibilidade detectados. Eles não impedem a atualização, mas resolva-os o quanto antes para evitar problemas em versões futuras.
- `noticeCount`: o número de alertas de incompatibilidade detectados. Eles não impedem a atualização, mas resolva-os o quanto antes para evitar problemas em versões futuras.
- `Summary`: um resumo da contagem de erros, avisos e alertas de compatibilidade da pré-verificação.

Exemplos de saída do log da pré-verificação do Aurora MySQL

Os exemplos a seguir mostram a saída do log de pré-verificação que provavelmente você verá. Consulte detalhes sobre as pré-verificações que são executadas em Referência de descrições da pré-verificação do Aurora MySQL.

Status da pré-verificação OK, sem incompatibilidades detectadas

A consulta de pré-verificação foi concluída com êxito. Nenhuma incompatibilidade foi detectada.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinytext",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny text columns",
  "status": "OK",
  "detectedProblems": []
},
```

Status da pré-verificação OK, sem erros detectados

A consulta de pré-verificação foi concluída com êxito. Um erro foi detectado.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexes",
  "status": "OK",
  "description": "Consider dropping the prefix indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test25.sbtest1",
        "description": "Table `test25`.`sbtest1` has an index `idx_t1` on geometry
 column/s. Mysql 8.0 does not support this type of index on a geometry column
```

```
 https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run 'DROP
 INDEX `idx_t1` ON `test25`.`sbtest1`;"
      },
 }
```

Status da pré-verificação OK, aviso detectado

Podem ser exibidos avisos quando uma pré-verificação é bem-sucedida ou malsucedida.

Aqui, a consulta de pré-verificação foi concluída com êxito. Dois avisos foram detectados.

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
    ]
}
```

Status da pré-verificação ERROR, sem incompatibilidades informadas

A consulta de pré-verificação falhou e exibiu um erro; portanto, não foi possível verificar as incompatibilidades.

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "ERROR",
  "description": "Can't connect to MySQL server on 'localhost:3306' (111) at
 13/08/2024 12:22:20 UTC. This failure can occur due to low memory available on the
 instance for executing upgrade prechecks. Please check 'FreeableMemory' Cloudwatch
 metric to verify the available memory on the instance while executing prechecks. If
 instance ran out of memory, we recommend to retry the upgrade on a higher instance
 class."
}
```

Essa falha pode ocorrer devido à reinicialização inesperada da instância ou à interrupção de uma consulta de pré-verificação de compatibilidade no banco de dados durante a execução. Por exemplo, em classes menores de instâncias de banco de dados, isso pode ocorrer quando a memória disponível na instância está baixa.

É possível usar a métrica `FreeableMemory` do Amazon CloudWatch para verificar a memória disponível na instância ao executar pré-verificações. Se a instância ficar sem memória, recomendamos tentar fazer a atualização novamente em uma classe maior de instância de banco de dados. Em alguns casos, você pode usar uma implantação azul/verde. Isso permite que as pré-verificações e atualizações sejam executadas no cluster de banco de dados “verde”, independentemente da workload de produção, que também consome recursos do sistema.

Para obter mais informações, consulte Solução de problemas de uso de memória para bancos de dados Aurora MySQL.

Resumo da pré-verificação, um erro e três avisos detectados

As pré-verificações de compatibilidade também contêm informações sobre as versões de origem e destino do Aurora MySQL e um resumo das contagens de erros, avisos e alertas no final da saída da pré-verificação.

Por exemplo, o resultado a seguir mostra que foi feita uma tentativa de atualização do Aurora MySQL 2.11.6 para o Aurora MySQL 3.07.1. A atualização exibiu um erro, três avisos e nenhum alerta. Como as atualizações não podem continuar quando é exibido um erro, você deve resolver o problema de compatibilidade routineSyntaxCheck e tentar fazer a atualização novamente.

```
{
  "serverAddress": "/tmp%2Fmysql.sock",
  "serverVersion": "5.7.12 - MySQL Community Server (GPL)",
```

```
 "targetVersion": "8.0.36",
 "auroraServerVersion": "2.11.6",
 "auroraTargetVersion": "3.07.1",
 "outfilePath": "/rdsdbdata/tmp/PreChecker.log",
 "checksPerformed": [{
     ... output for each individual precheck ...
     .
     .
     {
       "id": "oldTemporalCheck",
       "title": "Usage of old temporal type",
       "status": "OK",
         "detectedProblems": []
     },
     {
       "id": "routinesSyntaxCheck",
       "title": "MySQL 8.0 syntax check for routine-like objects",
       "status": "OK",
       "description": "The following objects did not pass a syntax check with
 the latest MySQL 8.0 grammar. A common reason is that they reference names that
 conflict with new reserved keywords. You must update these routine definitions and
 `quote` any such references before upgrading.",
       "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
       "detectedProblems": [{
           "level": "Error",
           "dbObject": "test.select_res_word",
           "description": "at line 2,18: unexpected token 'except'"
       }]
     },
     .
     .
     .
     {
       "id": "zeroDatesCheck",
       "title": "Zero Date, Datetime, and Timestamp values",
       "status": "OK",
       "description": "Warning: By default zero date/datetime/timestamp values
 are no longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are
 included in SQL_MODE by default. These modes should be used with strict mode as
 they will be merged with strict mode in a future release. If you do not include
 these modes in your SQL_MODE setting, you are able to insert date/datetime/
timestamp values that contain zeros. It is strongly advised to replace zero values
 with valid ones, as they may not work correctly in the future.",
       "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
```

```
        "detectedProblems": [{
            "level": "Warning",
            "dbObject": "global.sql_mode",
            "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
            },
            {
            "level": "Warning",
            "dbObject": "session.sql_mode",
            "description": " of 8 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
            }
          ]
      },
      .
      .
      .
  }],
  "errorCount": 1,
  "warningCount": 3,
  "noticeCount": 0,
  "Summary": "1 errors were found. Please correct these issues before upgrading to
 avoid compatibility issues."
}
```

Desempenho da pré-verificação do Aurora MySQL

As pré-verificações de compatibilidade são executadas antes que a instância de banco de dados fique off-line para a atualização. Portanto, em circunstâncias normais, elas não causam tempo de inatividade na instância de banco de dados durante a execução. No entanto, elas podem afetar a workload da aplicação em execução na instância de banco de dados do gravador. As pré-verificações acessam o dicionário de dados por meio de tabelas information_schema, que podem ser lentas se houver muitos objetos de banco de dados. Considere os seguintes fatores:

- A duração da pré-verificação varia de acordo com o número de objetos do banco de dados, como tabelas, colunas, rotinas e restrições. Clusters de banco de dados com um grande número de objetos podem levar mais tempo para serem executados.

  Por exemplo, removedFunctionsCheck pode levar mais tempo e usar mais recursos com base no número de objetos armazenados.

- Para atualizações no local, o uso de uma classe de instância de banco de dados maior (por exemplo, db.r5.24xlarge ou db.r6g.16xlarge) pode ajudar a concluir a atualização mais rapidamente usando mais CPU. Você pode reduzir o tamanho após a atualização.
- As consultas em `information_schema` em vários bancos de dados podem ser lentas, principalmente quando há muitos objetos e em instâncias de banco de dados menores. Nesses casos, considere usar clonagem, restauração de snapshots ou uma implantação azul/verde para atualizações.
- O uso de recursos (CPU, memória) da pré-verificação pode aumentar com mais objetos, resultando em tempos de execução mais longos em instâncias de banco de dados menores. Nesses casos, considere testar o uso de clonagem, restauração de snapshots ou uma implantação azul/verde para atualizações.

  Se as pré-verificações falharem devido à falta de recursos, você poderá detectar isso no log de pré-verificação usando a saída de status:

```
"status": "ERROR",
```

Para obter mais informações, consulte Como funciona a atualização da versão principal no local Aurora MySQL e Planejando uma atualização de versão principal para um cluster de Aurora MySQL.

Resumo das pré-verificações de atualização do MySQL Community

Veja a seguir uma lista geral de incompatibilidades entre o MySQL 5.7 e 8.0:

- O cluster de banco de dados compatível com o MySQL 5.7 não deve usar recursos incompatíveis com o MySQL 8.0.

  Para obter mais informações, consulte Recursos removidos no MySQL 8.0 na documentação do MySQL.
- Não deve haver violações de palavra-chave ou palavra reservada. Algumas palavras-chave podem ser reservadas no MySQL 8.0 que não eram reservadas anteriormente.

  Para obter mais informações, consulte Palavras-chave e palavras reservadas na documentação do MySQL.
- Para suporte melhorado do Unicode, considere a conversão de objetos que usam o conjunto de caracteres `utf8mb3` para que usem o conjunto de caracteres `utf8mb4`. O conjunto de caracteres `utf8mb3` está obsoleto. Além disso, considere o uso de `utf8mb4` para referências de conjuntos

de caracteres em vez de `utf8`, pois, no momento, `utf8` é um alias para o conjunto de caracteres `utf8mb3`.

Para obter mais informações, consulte The utf8mb3 character set (3-byte UTF-8 unicode encoding) na documentação do MySQL.

- Não deve haver tabelas do InnoDB com um formato de linha não padrão.
- Não deve haver atributos de tipo de tamanho `ZEROFILL` ou `display`.
- Não deve haver tabela particionada que usa um mecanismo de armazenamento que não tem suporte para particionamento nativo,
- Não deve haver tabelas no banco de dados do sistema `mysql` do MySQL 5.7 com o mesmo nome de uma tabela usada pelo dicionário de dados do MySQL 8.0.
- Não deve haver tabelas que usam funções ou tipos de dados obsoletos.
- Não deve haver nomes de restrição de chave externa maiores que 64 caracteres.
- Não deve haver modos obsoletos do SQL definidos na configuração de variável do sistema `sql_mode`.
- Não deve haver tabelas nem procedimentos armazenados com elementos de coluna `ENUM` ou `SET` individuais que excedam 255 caracteres.
- Não deve haver partições de tabela que residam em espaços de tabela compartilhados do InnoDB.
- Não deve haver referências circulares nos caminhos do arquivo de dados do espaço de tabela.
- Não deve haver consultas e definições de programa armazenadas que usem qualificadores `ASC` ou `DESC` para cláusulas `GROUP BY`.
- Não deve haver variáveis do sistema removidas, e as variáveis do sistema devem usar os novos valores padrão para o MySQL 8.0.
- Não deve haver valores zero (`0`) de data, data e hora ou carimbo de data e hora.
- Não deve haver inconsistências em esquemas resultantes da remoção ou da corrupção de arquivos.
- Não deve haver nomes de tabela que contenham a string de caracteres `FTS`.
- Não deve haver tabelas do InnoDB que pertençam a um mecanismo diferente.
- Não deve haver nomes de tabelas ou esquemas inválidos para o MySQL 5.7.

Consulte detalhes sobre as pré-verificações que são executadas em Referência de descrições da pré-verificação do Aurora MySQL.

Para ter informações sobre como atualizar para o MySQL 8.0, consulte Upgrading MySQL na documentação do MySQL. Consulte uma descrição geral das alterações feitas no MySQL 8.0 em What Is New in MySQL 8.0 na documentação do MySQL.

Resumo das pré-verificações de atualização do Aurora MySQL

O Aurora MySQL tem requisitos específicos quanto à atualização da versão 2 para a versão 3, incluindo o seguinte:

- Não deve haver nenhuma sintaxe SQL obsoleta, como `SQL_CACHE`, `SQL_NO_CACHE` e `QUERY_CACHE`, em visualizações, rotinas, gatilhos e eventos.
- Não deve haver colunas `FTS_DOC_ID` presentes em nenhuma tabela sem o índice `FTS`.
- Não deve haver nenhuma incompatibilidade de definição de coluna entre o dicionário de dados do InnoDB e a definição real da tabela.
- Todos os nomes de bancos de dados e tabelas devem estar em minúsculas quando o parâmetro `lower_case_table_names` é definido como 1.
- Eventos e gatilhos não devem ter um definidor vazio ou ausente ou um contexto de criação inválido.
- Todos os nomes de gatilhos em um banco de dados devem ser exclusivos.
- A recuperação de DDL e o DDL rápido não são compatíveis com o Aurora MySQL versão 3. Não deve haver artefatos nos bancos de dados relacionados a esses recursos.
- Tabelas com o formato de linha `REDUNDANT` ou `COMPACT` não podem ter índices maiores que 767 bytes.
- O tamanho do prefixo dos índices definidos nas colunas de texto `tiny` não pode exceder 255 bytes. Com o conjunto de caracteres `utf8mb4`, isso limita o tamanho do prefixo aceito a 63 caracteres.

  Uma extensão maior de prefixo era permitida no MySQL 5.7 usando o parâmetro `innodb_large_prefix`. Esse parâmetro está obsoleto no MySQL 8.0.
- Não deve haver nenhuma inconsistência de metadados do InnoDB na tabela `mysql.host`.
- Não deve haver incompatibilidade do tipo de dados da coluna nas tabelas do sistema.
- Não deve haver transações XA no estado `prepared`.
- Os nomes das colunas nas visualizações não podem exceder 64 caracteres.
- Caracteres especiais em procedimentos armazenados não podem ser inconsistentes.
- As tabelas não podem ter inconsistência no caminho do arquivo de dados.

Consulte detalhes sobre as pré-verificações que são executadas em Referência de descrições da pré-verificação do Aurora MySQL.

Referência de descrições da pré-verificação do Aurora MySQL

As pré-verificações de atualização do Aurora MySQL são descritas detalhadamente aqui.

Sumário

- Erros
  - Pré-verificações do MySQL que relatam erros
  - Pré-verificações do Aurora MySQL que relatam erros
- Avisos
  - Pré-verificações do MySQL que exibem avisos
  - Pré-verificações do Aurora MySQL que exibem avisos
- Avisos
- Erros, avisos ou alertas

Erros

As pré-verificações a seguir geram erros quando a pré-verificação falha e impede que a atualização continue.

Tópicos

- Pré-verificações do MySQL que relatam erros
- Pré-verificações do Aurora MySQL que relatam erros

Pré-verificações do MySQL que relatam erros

As seguintes pré-verificações são do MySQL Community:

- checkTableMysqlSchema
- circularDirectoryCheck
- columnsWhichCannotHaveDefaultsCheck
- depreciatedSyntaxCheck
- engineMixupCheck

- enumSetElementLengthCheck
- foreignKeyLengthCheck
- getDuplicateTriggers
- getEventsWithNullDefiner
- getMismatchedMetadata
- getTriggersWithNullDefiner
- getValueOfVariablelower_case_table_names
- groupByAscSyntaxCheck
- mysqlEmptyDotTableSyntaxCheck
- mysqlIndexTooLargeCheck
- mysqlInvalid57NamesCheck
- mysqlOrphanedRoutinesCheck
- mysqlSchemaCheck
- nonNativePartitioningCheck
- oldTemporalCheck
- partitionedTablesInSharedTablespaceCheck
- removedFunctionsCheck
- routineSyntaxCheck
- schemaInconsistencyCheck

checkTableMysqlSchema

Nível de pré-verificação: erro

Problemas relatados pelo comando **`check table x for upgrade`** para o esquema **`mysql`**

Antes de iniciar a atualização para o Aurora MySQL versão 3, `check table for upgrade` é executado em cada tabela no esquema `mysql` na instância de banco de dados. O comando `check table for upgrade` examina as tabelas em busca de possíveis problemas que possam surgir durante uma atualização para uma versão mais recente do MySQL. Executar esse comando antes de tentar fazer a atualização pode ajudar a identificar e resolver quaisquer incompatibilidades com antecedência, facilitando o respectivo processo.

Esse comando executa várias verificações em cada tabela, como as seguintes:

- Verificar se os metadados e a estrutura da tabela são compatíveis com a versão de destino do MySQL
- Verificar se há algum recurso obsoleto ou removido usado pela tabela
- Garantir que a atualização da tabela seja feito adequadamente sem perda de dados

Consulte mais informações em CHECK TABLE Statement na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "checkTableMysqlSchema",
  "title": "Issues reported by 'check table x for upgrade' command for mysql
 schema.",
  "status": "OK",
  "detectedProblems": []
}
```

A saída dessa pré-verificação depende do erro encontrado e de quando ele foi encontrado, pois `check table for upgrade` executa várias verificações.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

circularDirectoryCheck

Nível de pré-verificação: erro

Referências circulares a diretórios nos caminhos de arquivo de dados do espaço de tabela

Desde o MySQL 8.0.17, a cláusula `CREATE TABLESPACE ... ADD DATAFILE` não permite mais referências circulares a diretórios. Para evitar problemas de atualização, remova todas as referências circulares a diretórios dos caminhos de arquivo de dados do espaço de tabela antes de fazer a atualização para o Aurora MySQL versão 3.

Exemplos de resultado:

```
{
```

```
  "id": "circularDirectory",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "description": "Error: Following tablespaces contain circular directory references
 (e.g. the reference '/../') in data file paths which as of MySQL 8.0.17 are not
 permitted by the CREATE TABLESPACE ... ADD DATAFILE clause. An exception to the
 restriction exists on Linux, where a circular directory reference is permitted if
 the preceding directory is a symbolic link. To avoid upgrade issues, remove any
 circular directory references from tablespace data file paths before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-innodb-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "ts2",
        "description": "circular reference in datafile path: '/home/ec2-user/dbdata/
mysql_5_7_44/../ts2.ibd'",
        "dbObjectType": "Tablespace"
      }
  ]
}
```

Se você receber esse erro, recrie suas tabelas usando um espaço de tabela de arquivo por tabela. Use caminhos de arquivo padrão para todos os espaços e definições de tabela.

> **Note**
>
> O Aurora MySQL não é compatível com espaços de tabela gerais ou comandos `CREATE TABLESPACE`.
>
> Antes de recriar espaços de tabela, consulte Online DDL Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

Após a recriação, a pré-verificação é concluída com êxito, permitindo que a atualização prossiga.

```
{
  "id": "circularDirectoryCheck",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "detectedProblems": []
```

```
},
```

columnsWhichCannotHaveDefaultsCheck

Nível de pré-verificação: erro

Colunas que não podem ter valores padrão

Nas versões anteriores ao MySQL 8.0.13, as colunas `BLOB`, `TEXT`, `GEOMETRY` e `JSON` não podem ter valores padrão. Remova todas as cláusulas padrão dessas colunas antes de fazer a atualização para o Aurora MySQL versão 3. Consulte mais informações sobre alterações no tratamento padrão desses tipos de dados em Data Type Default Values na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "description": "Error: The following columns are defined as either BLOB, TEXT,
 GEOMETRY or JSON and have a default value set. These data types cannot have default
 values in MySQL versions prior to 8.0.13, while starting with 8.0.13, the default
 value must be specified as an expression. In order to fix this issue, please use
 the ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT statement.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/data-type-
defaults.html#data-type-defaults-explicit",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test_blob_default.geo_col",
        "description": "geometry"
      }
  ]
},
```

A pré-verificação retorna um erro porque a coluna `geo_col` na tabela `test.test_blob_default` está usando um tipo de dados `BLOB`, `TEXT`, `GEOMETRY` ou `JSON` com um valor padrão especificado.

Observando a definição da tabela, podemos ver que a coluna `geo_col` é definida como `geo_col geometry NOT NULL default ''`.

```
mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Remova essa cláusula padrão para permitir que a pré-verificação seja bem-sucedida.

> **Note**
>
> Antes de executar instruções `ALTER TABLE` ou recriar espaços de tabela, consulte Online DDL Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

```
mysql> ALTER TABLE test_blob_default modify COLUMN geo_col geometry NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

A pré-verificação é concluída com êxito e você pode tentar fazer a atualização novamente.

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "detectedProblems": []
},
```

depreciatedSyntaxCheck

Nível de pré-verificação: erro

Uso de palavras-chave obsoletas na definição

O MySQL 8.0 removeu o cache de consultas. Por isso, algumas sintaxes de SQL específicas de cache de consultas foram removidas. Se algum dos objetos do banco de dados contiver as palavras-chave `QUERY CACHE`, `SQL_CACHE` ou `SQL_NO_CACHE`, um erro de pré-verificação será exibido. Para resolver esse problema, recrie esses objetos removendo as palavras-chave mencionadas.

Exemplos de resultado:

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "description": "Error: The following DB objects contain keywords like 'QUERY
 CACHE', 'SQL_CACHE', 'SQL_NO_CACHE' which are not supported in major version 8.0.
 It is recommended to drop these DB objects or rebuild without any of the above
 keywords before upgrade.",
  "detectedProblems": [
      {
"level": "Error",
"dbObject": "test.no_query_cache_check",
"description": "PROCEDURE uses depreciated words in definition"
      }
  ]
}
```

A pré-verificação informa que o procedimento armazenado `test.no_query_cache_check` está usando uma das palavras-chave removidas. Observando a definição do procedimento, podemos ver que ele usa `SQL_NO_CACHE`.

```
mysql> show create procedure test.no_query_cache_check\G
*************************** 1. row ***************************
           Procedure: no_query_cache_check
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `no_query_cache_check`()
BEGIN
    SELECT SQL_NO_CACHE k from sbtest1 where id > 10 and id < 20 group by k asc;
END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
```

```
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Remova a palavra-chave.

```
mysql> drop procedure test.no_query_cache_check;
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter //

mysql> CREATE DEFINER=`reinvent`@`%` PROCEDURE `no_query_cache_check`() BEGIN
 SELECT k from sbtest1 where id > 10 and id < 20 group by k asc; END//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Após a remoção da palavra-chave, a pré-verificação é concluída com êxito.

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "detectedProblems": []
}
```

engineMixupCheck

Nível de pré-verificação: erro

Tabela reconhecida pelo InnoDB que pertence a um mecanismo diferente

De modo semelhante a schemaInconsistencyCheck, essa pré-verificação verifica se os metadados da tabela no MySQL são consistentes antes de prosseguir com a atualização.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

Exemplos de resultado:

```
{
```

```
  "id": "engineMixupCheck",
  "title": "Tables recognized by InnoDB that belong to a different engine",
  "status": "OK",
  "description": "Error: Following tables are recognized by InnoDB engine while the
 SQL layer believes they belong to a different engine. Such situation may happen
 when one removes InnoDB table files manually from the disk and creates e.g. a
 MyISAM table with the same name.\n\nA possible way to solve this situation is to
 e.g. in case of MyISAM table:\n\n1. Rename the MyISAM table to a temporary name
 (RENAME TABLE).\n2. Create some dummy InnoDB table (its definition does not need to
 match), then copy (copy, not move) and rename the dummy .frm and .ibd files to the
 orphan name using OS file commands.\n3. The orphan table can be then dropped (DROP
 TABLE), as well as the dummy table.\n4. Finally the MyISAM table can be renamed
 back to its original name.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.general_log_backup",
        "description": "recognized by the InnoDB engine but belongs to CSV"
      }
  ]
}
```

enumSetElementLengthCheck

Nível de pré-verificação: erro

Definições de coluna **`ENUM`** e **`SET`** contendo elementos com tamanho superior a 255 caracteres

Tabelas e procedimentos armazenados não devem ter elementos de coluna `ENUM` ou `SET` que excedam 255 caracteres ou 1.020 bytes. Antes do MySQL 8.0, o tamanho máximo combinado era de 64 K, mas o 8.0 limita elementos individuais a 255 caracteres ou 1.020 bytes (com suporte a multibyte). Se houver uma falha na pré-verificação referente a `enumSetElementLengthCheck`, modifique os elementos que excedam esses novos limites antes de tentar fazer a atualização novamente.

Exemplos de resultado:

```
{
  "id": "enumSetElementLengthCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
```

```
  "description": "Error: The following columns are defined as either ENUM or SET and
 contain at least one element longer that 255 characters. They need to be altered so
 that all elements fit into the 255 characters limit.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/string-type-
overview.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.large_set.s",
        "description": "SET contains element longer than 255 characters"
      }
  ]
},
```

A pré-verificação relata um erro porque a coluna `s` na tabela `test.large_set` contém um elemento SET com mais de 255 caracteres.

Após a redução do tamanho de SET dessa coluna, a pré-verificação é concluída com êxito, permitindo que a atualização prossiga.

```
{
  "id": "enumSetElementLenghtCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "detectedProblems": []
},
```

foreignKeyLengthCheck

Nível de pré-verificação: erro

Nomes de restrição de chave externa com mais de 64 caracteres

No MySQL, o tamanho dos identificadores é limitado a 64 caracteres, conforme descrito na documentação do MySQL. Tendo em vista os problemas identificados, em que as extensões das chaves externas podiam se igualar ou exceder esse valor, causando falhas de atualização, essa pré-verificação foi implementada. Se você encontrar erros com essa pré-verificação, altere ou renomeie a restrição para que ela tenha menos de 64 caracteres, antes de tentar fazer a atualização novamente.

Exemplos de resultado:

```
{
  "id": "foreignKeyLength",
  "title": "Foreign key constraint names longer than 64 characters",
  "status": "OK",
  "detectedProblems": []
}
```

getDuplicateTriggers

Nível de pré-verificação: erro

Todos os nomes de gatilho em um banco de dados devem ser exclusivos.

Devido a mudanças na implementação do dicionário de dados, o MySQL 8.0 não permite gatilhos com distinção entre letras maiúsculas e minúsculas em um banco de dados. Essa pré-verificação confirma que o cluster de banco de dados não tem um ou mais bancos de dados com gatilhos duplicados. Consulte mais informações em Identifier Case Sensitivity na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "description": "Error: You have one or more database containing duplicate
 triggers. Mysql 8.0 does not support case sensitive triggers within a database
 https://dev.mysql.com/doc/refman/8.0/en/identifier-case-sensitivity.html. To
 upgrade to MySQL 8.0, drop the triggers with case-insensitive duplicate names and
 recreate with distinct names.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_product"
      },
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_PRODUCT"
      }
  ]
```

```
}
```

A pré-verificação relata o erro de que o cluster de banco de dados tem dois gatilhos com o mesmo nome, mas com grafia diferente de maiúscula e minúscula: `test.before_insert_product` e `test.before_insert_PRODUCT`.

Antes de fazer a atualização, renomeie os gatilhos ou descarte-os e recrie-os com um novo nome.

Depois de renomear `test.before_insert_PRODUCT` para `test.before_insert_product_2`, a pré-verificação é bem-sucedida.

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "detectedProblems": []
}
```

getEventsWithNullDefiner

Nível de pré-verificação: erro

A coluna de definidor referente a **`mysql.event`** não pode ser nula nem estar em branco.

O atributo `DEFINER` especifica a conta do MySQL que possui uma definição de objeto armazenado, como um gatilho, um procedimento armazenado ou um evento. Esse atributo é útil principalmente em situações em que você quer controlar o contexto de segurança no qual o objeto armazenado é executado. Ao criar um objeto armazenado, se não for especificado um `DEFINER`, o padrão será o usuário que criou o objeto.

Ao fazer atualização para o MySQL 8.0, você não pode ter nenhum objeto armazenado que tenha um definidor `null` ou em branco no dicionário de dados do MySQL. Se você tiver esses objetos armazenados, será gerado um erro de pré-verificação. Você deve corrigi-lo antes de prosseguir com a atualização.

Exemplo de erro:

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
```

```
  "status": "OK",
  "description": "Error: Set definer column in mysql.event to a valid non-null
 definer.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.get_version",
        "description": "Set definer for event get_version in Schema test"
      }
  ]
}
```

A pré-verificação exibe um erro para o evento `test.get_version` porque ele tem um definidor `null`.

Para resolver isso, você pode verificar a definição do evento. Como você pode ver, o definidor é `null` ou está em branco.

```
mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+---------+
| db   | name        | definer |
+------+-------------+---------+
| test | get_version |         |
+------+-------------+---------+
1 row in set (0.00 sec)
```

Descarte ou recrie o evento com um definidor válido.

> **Note**
>
> Antes de descartar ou redefinir um `DEFINER`, analise e verifique cuidadosamente os requisitos da aplicação e de privilégios. Consulte mais informações em Stored Object Access Control na documentação do MySQL.

```
mysql> drop event test.get_version;
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> delimiter $$
mysql> CREATE EVENT get_version
```

```
    ->     ON SCHEDULE
    ->       EVERY 1 DAY
    ->     DO
    ->      ///DO SOMETHING //
    -> $$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+------------+
| db   | name        | definer    |
+------+-------------+------------+
| test | get_version | reinvent@% |
+------+-------------+------------+
1 row in set (0.00 sec)
```

Agora a pré-verificação é concluída com êxito.

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "detectedProblems": []},
```

getMismatchedMetadata

Nível de pré-verificação: erro

Incompatibilidade de definição de coluna entre o dicionário de dados do InnoDB e a definição real da tabela

De modo semelhante a schemaInconsistencyCheck, essa pré-verificação verifica se os metadados da tabela no MySQL são consistentes antes de prosseguir com a atualização. Nesse caso, a pré-verificação confere se as definições de coluna são correspondentes entre o dicionário de dados do InnoDB e a definição de tabela do MySQL. Se for detectada uma incompatibilidade, a atualização não prosseguirá.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

Exemplos de resultado:

```
{
  "id": "getMismatchedMetadata",
  "title": "Column definition mismatch between InnoDB Data Dictionary and actual
 table definition.",
  "status": "OK",
  "description": "Error: Your database has mismatched metadata. The upgrade to mysql
 8.0 will not succeed until this is fixed.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.mismatchTable",
        "description": "Table `test/mismatchTable` column names mismatch with InnoDb
 dictionary column names: iD <> id"
      }
  ]
}
```

A pré-verificação relata uma incompatibilidade nos metadados da coluna `id` na tabela `test.mismatchTable`. Especificamente, os metadados do MySQL têm o nome de coluna `iD`, enquanto o InnoDB tem o nome `id`.

getTriggersWithNullDefiner

Nível de pré-verificação: erro

A coluna de definidor referente a **`information_schema.triggers`** não pode ser **`null`** nem estar em branco.

A pré-verificação confere se o banco de dados não tem gatilhos definidos com definidores `null` ou em branco. Consulte mais informações sobre os requisitos de definidores para objetos armazenados em getEventsWithNullDefiner.

Exemplos de resultado:

```
{
  "id": "getTriggersWithNullDefiner",
  "title": "The definer column for information_schema.triggers cannot be null or
 blank.",
  "status": "OK",
  "detectedProblems": [
      {
```

```
        "level": "Error",
        "dbObject": "test.example_trigger",
        "description": "Set definer for trigger example_trigger in Schema test"
      }
  ]
}
```

A pré-verificação exibe um erro porque o gatilho `example_trigger` no esquema `test` tem um definidor `null`. Para corrigir esse problema, ajuste o definidor recriando o gatilho com um usuário válido ou descarte o gatilho. Consulte mais informações no exemplo em getEventsWithNullDefiner.

> **Note**
>
> Antes de descartar ou redefinir um `DEFINER`, analise e verifique cuidadosamente os requisitos da aplicação e de privilégios. Consulte mais informações em Stored Object Access Control na documentação do MySQL.

getValueOfVariablelower_case_table_names

Nível de pré-verificação: erro

Todos os nomes de bancos de dados ou tabelas devem estar em minúsculas quando o parâmetro **`lower_case_table_names`** é definido como **`1`**.

Antes do MySQL 8.0, os nomes de banco de dados, os nomes de tabela e outros objetos correspondiam a arquivos no diretório de dados, como metadados baseados em arquivos (.frm). A variável de sistema lower_case_table_names permite que os usuários controlem como o servidor lida com a distinção entre letras maiúsculas e minúsculas do identificador para objetos de banco de dados e o armazenamento desses objetos de metadados. Esse parâmetro podia ser alterado em um servidor já inicializado após uma reinicialização.

No entanto, no MySQL 8.0, embora esse parâmetro ainda controle como o servidor lida com a distinção entre letras maiúsculas e minúsculas do identificador, ele não pode ser alterado após a inicialização do dicionário de dados. Ao fazer a atualização ou criar um banco de dados do MySQL 8.0, o valor definido como `lower_case_table_names` na primeira vez em que o dicionário de dados é iniciado no MySQL é usado durante o ciclo de vida desse banco de dados. Essa restrição foi implementada como parte da implementação do Atomic Data Dictionary, em

que objetos de banco de dados são migrados de metadados baseados em arquivos para tabelas internas do InnoDB no esquema `mysql`.

Consulte mais informações em Data Dictionary Changes na documentação do MySQL.

Para evitar problemas durante a atualização de metadados baseados em arquivo para o novo Atomic Data Dictionary, essa pré-verificação confirma se, quando `lower_case_table_names = 1`, todas as tabelas são armazenadas em disco em letras minúsculas. Se não forem, um erro de pré-verificação será exibido e você deverá corrigir os metadados antes de prosseguir com a atualização.

Exemplos de resultado:

```
{
  "id": "getValueOfVariablelower_case_table_names",
  "title": "MySQL pre-checks that all database or table names are lowercase when the
 lower_case_table_names parameter is set to 1.",
  "status": "OK",
  "description": "Error: You have one or more databases or tables with uppercase
 letters in the names, but the lower_case_table_names parameter is set to 1. To
 upgrade to MySQL 8.0, either change all database or table names to lowercase, or
 set the parameter to 0.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.TEST",
        "description": "Table test.TEST contains one or more capital letters in name
 while lower_case_table_names = 1"
      }
  ]
}
```

Um erro é exibido porque a tabela `test.TEST` contém letras maiúsculas, mas `lower_case_table_names` está definido como `1`.

Para resolver esse problema, você pode renomear a tabela com letras minúsculas ou modificar o parâmetro `lower_case_table_names` no cluster de banco de dados antes de iniciar a atualização.

> **Note**
>
> Teste e examine cuidadosamente a documentação sobre a distinção entre letras maiúsculas e minúsculas no MySQL e como essas alterações podem afetar sua aplicação.
> Consulte também a documentação do MySQL 8.0 sobre como lower_case_table_names são tratados de forma diferente no MySQL 8.0.

groupByAscSyntaxCheck

Nível de pré-verificação: erro

Uso da sintaxe **`GROUP BY ASC/DESC`** removida

Desde o MySQL 8.0.13, a sintaxe obsoleta `ASC` ou `DESC` para cláusulas `GROUP BY` foi removida. As consultas que dependem da classificação `GROUP BY` agora podem produzir resultados diferentes. Para ter uma ordem de classificação específica, use uma cláusula `ORDER BY`. Se algum objeto no banco de dados estiver usando essa sintaxe, você deverá recriá-lo por meio de uma cláusula `ORDER BY` antes de tentar fazer a atualização novamente. Consulte mais informações em SQL Changes na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "groupbyAscSyntaxCheck",
  "title": "Usage of removed GROUP BY ASC/DESC syntax",
  "status": "OK",
  "description": "Error: The following DB objects use removed GROUP BY ASC/DESC
 syntax. They need to be altered so that ASC/DESC keyword is removed from GROUP BY
 clause and placed in appropriate ORDER BY clause.",
  "documentationLink": "https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
news-8-0-13.html#mysqld-8-0-13-sql-syntax",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.groupbyasc",
        "description": "PROCEDURE uses removed GROUP BY ASC syntax",
        "dbObjectType": "Routine"
      }
  ]
```

```
}
```

mysqlEmptyDotTableSyntaxCheck

Nível de pré-verificação: erro

Verifique a sintaxe obsoleta **`.<table>`** usada nas rotinas.

No MySQL 8.0, as rotinas não podem mais conter a sintaxe obsoleta do identificador (`\".<table>\"`). Se alguma rotina ou gatilho armazenado tiver esses identificadores, a atualização falhará. Por exemplo, a seguinte referência a `.dot_table` não é mais permitida:

```
mysql> show create procedure incorrect_procedure\G
*************************** 1. row ***************************
           Procedure: incorrect_procedure
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `incorrect_procedure`()
BEGIN delete FROM .dot_table; select * from .dot_table where 1=1; END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Depois que você recria as rotinas e os gatilhos para usar o escape e a sintaxe correta do identificador, a pré-verificação é concluída com êxito e a atualização pode prosseguir. Consulte mais informações sobre identificadores em Schema Object Names na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "mysqlEmptyDotTableSyntaxCheck",
  "title": "Check for deprecated '.<table>' syntax used in routines.",
  "status": "OK",
  "description": "Error: The following routines contain identifiers in deprecated
 identifier syntax (\".<table>\"), and should be corrected before upgrade:\n",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.incorrect_procedure",
        "description": " routine body contains deprecated identifiers."
```

```
      }
  ]
}
```

A pré-verificação exibe um erro referente à rotina `incorrect_procedure` no banco de dados `test` porque ela contém uma sintaxe obsoleta.

Depois que você corrige a rotina, a pré-verificação é bem-sucedida e você pode tentar fazer a atualização novamente.

mysqlIndexTooLargeCheck

Nível de pré-verificação: erro

Verifique se há índices que são muito grandes para funcionar em versões do MySQL posteriores à 5.7

Com relação a formatos de linha compactos ou redundantes, não deveria ser possível criar um índice com um prefixo com mais de 767 bytes. No entanto, antes do MySQL versão 5.7.35, isso era possível. Consulte mais informações em MySQL 5.7.35 Release Notes.

Todos os índices afetados por esse erro ficarão inacessíveis após a atualização para o MySQL 8.0. Essa pré-verificação identifica índices incorretos que precisam ser recriados antes que a atualização possa prosseguir.

```
{
 "id": "mysqlIndexTooLargeCheck",
 "title": "Check for indexes that are too large to work on higher versions of MySQL
Server than 5.7",
 "status": "OK",
 "description": "Error: The following indexes ware made too large for their format
in an older version of MySQL (older than 5.7.34). Normally those indexes within
tables with compact or redundant row formats shouldn't be larger than 767 bytes. To
fix this problem those indexes should be dropped before upgrading or those tables
will be inaccessible.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test.table_with_large_idx",
       "description": "IDX_2"
     }
 ]
```

```
}
```

A pré-verificação exibe um erro porque a tabela `test.table_with_large_idx` contém um índice em uma tabela que usa um formato de linha compacto ou redundante com mais de 767 bytes. Essas tabelas ficariam inacessíveis depois da atualização para o MySQL 8.0. Antes de prosseguir com a atualização, execute uma destas ações:

- Elimine o índice mencionado na pré-verificação.
- Adicione um índice mencionado na pré-verificação.
- Altere o formato da linha usado pela tabela.

Aqui, nós recriamos a tabela para resolver a falha de pré-verificação. Antes de recriar a tabela, verifique se o innodb_file_format está definido como `Barracuda` e o innodb_default_row_format está definido como `dynamic`. Esses são os padrões no MySQL 5.7. Consulte mais informações em InnoDB Row Formats e em InnoDB File-Format Management na documentação do MySQL.

> **Note**
>
> Antes de recriar espaços de tabela, consulte Online DDL Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

```
mysql > select @@innodb_file_format,@@innodb_default_row_format;
+----------------------+-----------------------------+
| @@innodb_file_format | @@innodb_default_row_format |
+----------------------+-----------------------------+
| Barracuda            | dynamic                     |
+----------------------+-----------------------------+
1 row in set (0.00 sec)

mysql> optimize table table_with_large_idx;
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| Table                     | Op       | Msg_type | Msg_text
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| test.table_with_large_idx | optimize | note     | Table does not support optimize,
 doing recreate + analyze instead |
```

```
| test.table_with_large_idx | optimize | status   | OK
                                  |
+---------------------------+----------+----------
+--------------------------------------------------------------------+
2 rows in set (0.02 sec)

# Verify FILE_FORMAT and ROW_FORMAT
mysql>  select * from information_schema.innodb_sys_tables where name like 'test/
table_with_large_idx';
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
| TABLE_ID | NAME                      | FLAG | N_COLS | SPACE | FILE_FORMAT |
 ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
|       43 | test/table_with_large_idx |   33 |      4 |    26 | Barracuda   |
 Dynamic    |             0 | Single     |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
1 row in set (0.00 sec)
```

Após a recriação da tabela, a pré-verificação é concluída com êxito e a atualização pode prosseguir.

```
{
  "id": "mysqlIndexTooLargeCheck",
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlInvalid57NamesCheck

Nível de pré-verificação: erro

Verifique se há nomes inválidos de tabelas e esquemas usados no MySQL 5.7

Ao migrar para o novo dicionário de dados no MySQL 8.0, a instância de banco de dados não pode conter esquemas ou tabelas com o prefixo `#mysql50#`. Se algum desses objetos existir, a atualização falhará. Para resolver esse problema, execute mysqlcheck nos esquemas e tabelas exibidos.

> **Note**
>
> Você deve usar uma versão do MySQL 5.7 de `mysqlcheck`, pois --fix-db-names e --fix-table-names foram removidos do MySQL 8.0.

Exemplos de resultado:

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "description": "The following tables and/or schemas have invalid names. In order
 to fix them use the mysqlcheck utility as follows:\n\n  $ mysqlcheck --check-
upgrade --all-databases\n  $ mysqlcheck --fix-db-names --fix-table-names --all-
databases\n\nOR via mysql client, for eg:\n\n  ALTER DATABASE `#mysql50#lost+found`
 UPGRADE DATA DIRECTORY NAME;",
  "documentationLink": "https://dev.mysql.com/doc/refman/5.7/en/identifier-
mapping.html https://dev.mysql.com/doc/refman/5.7/en/alter-database.html https://
dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "#mysql50#fix_db_names",
        "description": "Schema name"
      }
  ]
}
```

A pré-verificação informa que o esquema `#mysql50#fix_db_names` tem um nome inválido.

Após a correção do nome do esquema, a pré-verificação é concluída com êxito, permitindo que a atualização prossiga.

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlOrphanedRoutinesCheck

Nível de pré-verificação: erro

Verifique se há rotinas órfãs na versão 5.7

Ao migrar para o novo dicionário de dados no MySQL 8.0, se houver algum procedimento armazenado no banco de dados em que o esquema não exista mais, a atualização falhará. Essa pré-verificação confere se todos os esquemas referidos nos procedimentos armazenados na instância de banco de dados ainda existem. Para permitir que a atualização prossiga, descarte esses procedimentos armazenados.

Exemplos de resultado:

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "description": "Error: The following routines have been orphaned. Schemas that
 they are referencing no longer exists.\nThey have to be cleaned up or the upgrade
 will fail.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "dropped_db.get_version",
        "description": "is orphaned"
      }
  ]
},
```

A pré-verificação informa que o procedimento armazenado `get_version` no banco de dados `dropped_db` é órfão.

Para limpar esse procedimento, você pode recriar o esquema ausente.

```
mysql> create database dropped_db;
Query OK, 1 row affected (0.01 sec)
```

Depois que o esquema for recriado, você poderá descartar o procedimento para permitir que a atualização prossiga.

```
{
```

```
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlSchemaCheck

Nível de pré-verificação: erro

Nomes de tabela no esquema **`mysql`** conflitantes com novas tabelas no MySQL 8.0

O novo Atomic Data Dictionary, introduzido no MySQL 8.0, armazena todos os metadados em um conjunto de tabelas internas do InnoDB no esquema `mysql`. Durante a atualização, as novas tabelas do dicionário de dados interno são criadas no esquema `mysql`. Para evitar conflitos de nomenclatura, que resultariam em falhas de atualização, a pré-verificação examina todos os nomes de tabela no esquema `mysql` para garantir que nenhum dos novos nomes já esteja em uso. Se estiverem, um erro será exibido, e a atualização não poderá prosseguir.

Exemplos de resultado:

```
{
  "id": "mysqlSchema",
  "title": "Table names in the mysql schema conflicting with new tables in the
 latest MySQL.",
  "status": "OK",
  "description": "Error: The following tables in mysql schema have names that will
 conflict with the ones introduced in the latest version. They must be renamed or
 removed before upgrading (use RENAME TABLE command). This may also entail changes
 to applications that use the affected tables.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrade-before-you-
begin.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.tablespaces",
        "description": "Table name used in mysql schema.",
        "dbObjectType": "Table"
      }
  ]
}
```

Um erro é exibido porque há uma tabela chamada `tablespaces` no esquema `mysql`. Esse é um dos novos nomes de tabela do dicionário de dados interno no MySQL 8.0. Você deve renomear ou remover essas tabelas antes da atualização usando o comando `RENAME TABLE`.

nonNativePartitioningCheck

Nível de pré-verificação: erro

Tabelas particionadas usando mecanismos com particionamento não nativo

De acordo com a documentação do MySQL 8.0, no momento, dois mecanismos de armazenamento fornecem suporte nativo ao particionamento: InnoDB e NDB. Dentre eles, somente o InnoDB é compatível com o Aurora MySQL versão 3 compatível com o MySQL 8.0. Qualquer tentativa de criar tabelas particionadas no MySQL 8.0 usando qualquer outro mecanismo de armazenamento apresenta falha. Essa pré-verificação procura tabelas no cluster de banco de dados que estão usando particionamento não nativo. Se encontrar algum resultado, você deverá remover o particionamento ou realizar a conversão do mecanismo de armazenamento para o InnoDB.

Exemplos de resultado:

```
{
  "id": "nonNativePartitioning",
  "title": "Partitioned tables using engines with non native partitioning",
  "status": "OK",
  "description": "Error: In the latest MySQL storage engine is responsible for
 providing its own partitioning handler, and the MySQL server no longer provides
 generic partitioning support. InnoDB and NDB are the only storage engines that
 provide a native partitioning handler that is supported in the latest MySQL. A
 partitioned table using any other storage engine must be altered—either to convert
 it to InnoDB or NDB, or to remove its partitioning—before upgrading the server,
 else it cannot be used afterwards.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-configuration-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partMyisamTable",
         "description": "MyISAM engine does not support native partitioning",
         "dbObjectType": "Table"
      }
  ]
```

```
}
```

Aqui, uma tabela MyISAM está usando particionamento, o que requer uma ação para que a atualização possa prosseguir.

oldTemporalCheck

Nível de pré-verificação: erro

Uso do tipo temporal antigo

“Temporais antigos” referem-se às colunas do tipo temporal (como `TIMESTAMP` e`DATETIME`) criadas no MySQL versões 5.5 e anteriores. No MySQL 8.0, a compatibilidade com esses tipos de dados temporais antigos foi removida, o que significa que as atualizações no local do MySQL 5.7 para 8.0 não serão possíveis se o banco de dados tiver esses tipos temporais antigos. Para corrigir isso, você deve recriar todas as tabelas que contenham esses tipos de dados temporais antigos antes de prosseguir com a atualização.

Consulte mais informações sobre a descontinuação de tipos de dados temporais antigos no MySQL 5.7 neste blog. Consulte mais informações sobre a remoção de tipos de dados temporais antigos no MySQL 8.0 neste blog.

> **Note**
>
> Antes de recriar espaços de tabela, consulte Online DDL Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

Exemplos de resultado:

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "description": "Error: Following table columns use a deprecated and no longer
 supported temporal disk storage format. They must be converted to the new format
 before upgrading. It can by done by rebuilding the table using 'ALTER TABLE
 <table_name> FORCE' command",
  "documentationLink": "https://dev.mysql.com/blog-archive/mysql-8-0-removing-
support-for-old-temporal-datatypes/",
```

```
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.55_temporal_table.timestamp_column",
        "description": "timestamp /* 5.5 binary format */",
        "dbObjectType": "Column"
      }
  ]
},
```

Um erro é relatado para a coluna `timestamp_column` na tabela `test.55_temporal_table`, porque ela usa um formato de armazenamento em disco temporal antigo que não é mais compatível.

Para resolver esse problema e permitir que a atualização prossiga, recrie a tabela para converter o formato de armazenamento em disco temporal antigo no novo apresentado no MySQL 5.6. Antes de fazer isso, consulte mais informações e pré-requisitos em Converting Between 3-Byte And 4-Byte Unicode Character Sets na documentação do MySQL.

A execução do comando a seguir para recriar as tabelas mencionadas nessa pré-verificação converte os tipos temporais antigos no formato mais novo com precisão de fração de segundos.

```
ALTER TABLE ... ENGINE=InnoDB;
```

Consulte mais informações sobre como recriar tabelas em ALTER TABLE Statement na documentação do MySQL.

Depois que você recria a tabela em questão e reinicia a atualização, a verificação de compatibilidade é concluída com êxito, permitindo que a atualização prossiga.

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "detectedProblems": []
}
```

partitionedTablesInSharedTablespaceCheck

Nível de pré-verificação: erro

Uso de tabelas particionadas em espaços de tabela compartilhados

Desde o MySQL 8.0.13, não é mais possível colocar partições de tabela em espaços de tabela compartilhados. Antes da atualização, mova essas tabelas de espaços de tabela compartilhados para espaços de tabela de arquivo por tabela.

> **Note**
>
> Antes de recriar espaços de tabela, consulte Partitioning Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

Exemplos de resultado:

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "description": "Error: The following tables have partitions in shared tablespaces.
 They need to be moved to file-per-table tablespace before upgrading. You can do
 this by running query like 'ALTER TABLE table_name REORGANIZE PARTITION X INTO
 (PARTITION X VALUES LESS THAN (30) TABLESPACE=innodb_file_per_table);'",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partInnoDBTable",
        "description": "Partition p1 is in shared tablespace innodb",
        "dbObjectType": "Table"
      }
  ]
}
```

A pré-verificação falha porque a partição `p1` da tabela `test.partInnoDBTable` está no espaço de tabela do sistema.

Para resolver esse problema, recrie a tabela `test.partInnodbTable`, colocando a partição incorreta `p1` em um espaço de tabela de arquivo por tabela.

```
mysql > ALTER TABLE partInnodbTable REORGANIZE PARTITION p1
```

```
    ->   INTO (PARTITION p1 VALUES LESS THAN ('2014-01-01')
 TABLESPACE=innodb_file_per_table);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Depois disso, a pré-verificação é concluída com êxito.

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "detectedProblems": []
}
```

removedFunctionsCheck

Nível de pré-verificação: erro

Uso de funções que foram removidas da versão mais recente do MySQL

No MySQL 8.0, várias funções integradas foram removidas. Essa pré-verificação examina o banco de dados em busca de objetos que possam usar essas funções. Se algo for encontrado, será exibido um erro. Você deve resolver os problemas antes de tentar fazer a atualização novamente.

As funções removidas são em sua maioria funções espaciais, que foram substituídas por funções ST_* equivalentes. Nesses casos, você modifica os objetos do banco de dados para usar a nova nomenclatura do procedimento. Para obter mais informações, consulte Recursos removidos no MySQL 8.0 na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "description": "Error: The following DB objects use functions that were removed
 in the latest MySQL version. Please make sure to update them to use supported
 alternatives before upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
```

```
        "level": "Error",
        "dbObject": "test.GetLocationsInPolygon",
        "description": "PROCEDURE uses removed function POLYGONFROMTEXT (consider
 using ST_POLYGONFROMTEXT instead)",
        "dbObjectType": "Routine"
      },
      {
        "level": "Error",
        "dbObject": "test.InsertLocation",
        "description": "PROCEDURE uses removed function POINTFROMTEXT (consider
 using ST_POINTFROMTEXT instead)",
        "dbObjectType": "Routine"
      }
  ]
},
```

A pré-verificação informa que o procedimento armazenado `test.GetLocationsInPolygon` está usando duas funções removidas: POLYGONFROMTEXT e POINTFROMTEXT. Ela também sugere que você use como substituição ST_POLYGONFROMTEXT e ST_POINTFROMTEXT novos. Depois que você recria o procedimento usando as sugestões, a pré-verificação é concluída com êxito.

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "detectedProblems": []
},
```

> **Note**
>
> Embora, na maioria dos casos, as funções obsoletas tenham substituições diretas, teste a aplicação e examine a documentação em busca de quaisquer mudanças de comportamento resultantes da alteração.

routineSyntaxCheck

Nível de pré-verificação: erro

Verificação de sintaxe do MySQL para objetos de rotina

O MySQL 8.0 introduziu palavras-chave reservadas que não eram reservadas anteriormente. A pré-verificação de atualização avalia o uso de palavras-chave reservadas nos nomes de objeto de banco de dados e em suas definições e corpo. Se for detectado que estão sendo usadas palavras-chave reservadas em objetos de banco de dados, como procedimentos armazenados, funções, eventos e gatilhos, a atualização falhará e um erro será publicado no arquivo `upgrade-prechecks.log`. Para resolver o problema, você deve atualizar essas definições de objeto e colocar essas referências entre aspas simples (‘’) antes da atualização. Consulte mais informações sobre o escape de palavras reservadas no MySQL em String Literals na documentação do MySQL.

Ou é possível alterar o nome para um nome diferente, o que pode exigir alterações na aplicação.

Exemplos de resultado:

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "description": "The following objects did not pass a syntax check with the latest
 MySQL grammar. A common reason is that they reference names that conflict with new
 reserved keywords. You must update these routine definitions and `quote` any such
 references before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
         "level": "Error",
         "dbObject": "test.select_res_word",
         "description": "at line 2,18: unexpected token 'except'",
         "dbObjectType": "Routine"
      }
  ]
}
```

Para corrigir esse problema, confira a definição da rotina.

```
SHOW CREATE PROCEDURE test.select_res_word\G

*************************** 1. row ***************************
           Procedure: select_res_word
            sql_mode:
 ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
    Create Procedure: CREATE PROCEDURE 'select_res_word'()
```

```
BEGIN
    SELECT * FROM except;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

O procedimento usa uma tabela chamada `except`, que é uma nova palavra-chave no MySQL 8.0. Recrie o procedimento fazendo o escape do literal da string.

```
> drop procedure if exists select_res_word;
Query OK, 0 rows affected (0.00 sec)

> DELIMITER $$
 > CREATE PROCEDURE select_res_word()
    -> BEGIN
    ->     SELECT * FROM 'except';
    -> END$$
Query OK, 0 rows affected (0.00 sec)

 > DELIMITER ;
```

A pré-verificação agora é concluída com êxito.

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "detectedProblems": []
}
```

schemaInconsistencyCheck

Nível de pré-verificação: erro

Inconsistências em esquemas resultantes da remoção ou corrupção de arquivos

Como descrito anteriormente, o MySQL 8.0 introduziu o Atomic Data Dictionary, que armazena todos os metadados em um conjunto de tabelas internas do InnoDB no esquema `mysql`. Essa nova arquitetura oferece uma forma transacional compatível com ACID de gerenciar metadados de banco de dados, resolvendo o problema de “DDL atômica” com a antiga abordagem baseada

em arquivos. Antes do MySQL 8.0, era possível que objetos de esquema ficassem órfãos se uma operação de DDL fosse interrompida inesperadamente. A migração de metadados baseados em arquivos para as novas tabelas do Atomic Data Dictionary durante a atualização garante que não existam esses objetos de esquema órfãos na instância de banco de dados. Se algum objeto órfão for encontrado, a atualização falhará.

Para ajudar a detectar esses objetos órfãos antes de iniciar a atualização, a `schemaInconsistencyCheck` pré-verificação é executada para garantir que todos os objetos de metadados do dicionário de dados estejam sincronizados. Se algum objeto de metadados órfão for detectado, a atualização não prosseguirá. Para continuar com a a atualização, limpe esses objetos de metadados órfãos.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

Exemplos de resultado:

```
{
  "id": "schemaInconsistencyCheck",
  "title": "Schema inconsistencies resulting from file removal or corruption",
  "status": "OK",
  "description": "Error: Following tables show signs that either table datadir
 directory or frm file was removed/corrupted. Please check server logs, examine
 datadir to detect the issue and fix it before upgrade",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.schemaInconsistencyCheck_failure",
        "description": "present in INFORMATION_SCHEMA's INNODB_SYS_TABLES table but
 missing from TABLES table"
      }
  ]
}
```

A pré-verificação informa que a tabela `test.schemaInconsistencyCheck_failure` tem metadados inconsistentes. Nesse caso, a tabela existe nos metadados do mecanismo de armazenamento do InnoDB (`information_schema.INNODB_SYS_TABLES`), mas não nos metadados do MySQL (`information_schema.TABLES`).

Pré-verificações do Aurora MySQL que relatam erros

As pré-verificações a seguir são específicas do Aurora MySQL:

- auroraCheckDDLRecovery
- auroraCheckRdsUpgradePrechecksTable
- auroraFODUpgradeCheck
- auroraGetDanglingFulltextIndex
- auroraUpgradeCheckForDatafilePathInconsistency
- auroraUpgradeCheckForFtsSpaceIdZero
- auroraUpgradeCheckForIncompleteXATransactions
- auroraUpgradeCheckForInstanceLimit
- auroraUpgradeCheckForInternalUsers
- auroraUpgradeCheckForMasterUser
- auroraUpgradeCheckForPrefixIndexOnGeometryColumns
- auroraUpgradeCheckForSpecialCharactersInProcedures
- auroraUpgradeCheckForSysSchemaObjectTypeMismatch
- auroraUpgradeCheckForViewColumnNameLength
- auroraUpgradeCheckIndexLengthLimitOnTinyColumns
- auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable
- auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

auroraCheckDDLRecovery

Nível de pré-verificação: erro

Verifique se há artefatos relacionados ao recurso de recuperação de DDL do Aurora

Como parte da implementação de recuperação da linguagem de definição de dados (DDL) no Aurora MySQL, os metadados nas instruções de DDL em trânsito são mantidos nas tabelas `ddl_log_md_table` e `ddl_log_table` no esquema `mysql`. A implementação da recuperação de DDL do Aurora não é compatível com a versão 3 em diante, porque a funcionalidade faz parte da nova implementação do Atomic Data Dictionary no MySQL 8.0. Se você tiver alguma instrução de DDL em execução durante as verificações de compatibilidade, essa pré-verificação poderá

falhar. Recomendamos tentar fazer a atualização quando nenhuma instrução de DDL estiver em execução.

Se essa pré-verificação falhar sem que nenhuma instrução de DDL esteja em execução, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

Se alguma instrução DDL estiver em execução, a saída da pré-verificação imprimirá a seguinte mensagem:

```
“There are DDL statements in process. Please allow DDL statements to finish before
 upgrading.”
```

Exemplos de resultado:

```
{
  "id": "auroraCheckDDLRecovery",
  "title": "Check for artifacts related to Aurora DDL recovery feature",
  "status": "OK",
  "description": "Aurora implementation of DDL recovery is not supported from 3.x
 onwards. This check verifies that the database do not have artifacts realted to the
 feature",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_md_table",
        "description": "Table mysql.ddl_log_md_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_table",
        "description": "Table mysql.ddl_log_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "information_schema.processlist",
        "description": "There are DDL statements in process. Please allow DDL
 statements to finish before upgrading."
      }
```

```
  ]
}
```

A pré-verificação exibiu um erro devido à execução simultânea de um DDL em trânsito com as verificações de compatibilidade. Recomendamos que você tente fazer a atualização novamente sem DDLs em execução.

auroraCheckRdsUpgradePrechecksTable

Nível de pré-verificação: erro

Verifique a existência da tabela **`mysql.rds_upgrade_prechecks`**

Essa é uma pré-verificação somente interna realizada pelo serviço RDS. Os erros serão gerenciados automaticamente na atualização e podem ser ignorados com segurança.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

```
{
  "id": "auroraCheckRdsUpgradePrechecksTable",
  "title": "Check existence of mysql.rds_upgrade_prechecks table",
  "status": "OK",
  "detectedProblems": []
}
```

auroraFODUpgradeCheck

Nível de pré-verificação: erro

Verifique se há artefatos relacionados ao recurso de DDL rápida do Aurora

A otimização da DDL rápida foi introduzida no modo de laboratório no Aurora MySQL versão 2 para melhorar a eficiência de algumas operações de DDL. No Aurora MySQL versão 3, o modo de laboratório foi removido e a implementação da DDL rápida foi substituída pelo recurso do MySQL 8.0 chamado DDL instantânea.

Antes de fazer a atualização para o Aurora MySQL versão 3, todas as tabelas que usam a DDL rápida no modo de laboratório precisarão ser recriadas executando o comando `OPTIMIZE`

`TABLE` ou `ALTER TABLE ... ENGINE=InnoDB` para garantir a compatibilidade com o Aurora MySQL versão 3.

Essa pré-verificação exibe uma lista dessas tabelas. Depois que você recriar as tabelas exibidas, você poderá tentar fazer a atualização novamente.

Exemplos de resultado:

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "description": "Aurora fast DDL is not supported from 3.x onwards. This check
 verifies that the database does not have artifacts related to the feature",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/AuroraMySQL.Managing.FastDDL.html#AuroraMySQL.Managing.FastDDL-v2",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test",
        "description": "Your table has pending Aurora fast DDL operations. Run
 'OPTIMIZE TABLE <table name>' for the table to apply all the pending DDL updates.
 Then try the upgrade again."
      }
  ]
}
```

A pré-verificação informa que a tabela `test.test` tem operações de DDL rápida pendentes.

Para permitir que a atualização prossiga, você pode recriar a tabela e tentar fazer upload novamente.

> **Note**
>
> Antes de recriar espaços de tabela, consulte Online DDL Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

```
mysql> optimize table test.test;
+-----------+----------+----------
+-------------------------------------------------------------------+
```

```
| Table     | Op       | Msg_type | Msg_text
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
| test.test | optimize | note     | Table does not support optimize, doing recreate
 + analyze instead |
| test.test | optimize | status   | OK
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.04 sec)
```

Depois que você recria a tabela, a pré-verificação é bem-sucedida.

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "detectedProblems": []
}
```

auroraGetDanglingFulltextIndex

Nível de pré-verificação: erro

Tabelas com referência de índice **FULLTEXT** pendente

Antes do MySQL 5.6.26, era possível que, após a eliminação de um índice de pesquisa de texto completo, as colunas ocultas `FTS_DOC_ID` e `FTS_DOC_ID_INDEX` ficassem órfãs. Consulte mais informações em Bug #76012.

Se você tiver alguma tabela criada em versões anteriores do MySQL em que isso tenha ocorrido, as atualizações para o Aurora MySQL versão 3 poderão falhar. Essa pré-verificação confere se esses índices órfãos ou “pendentes” de texto completo não existem no cluster de banco de dados antes da atualização para o MySQL 8.0. Se essa pré-verificação falhar, recrie todas as tabelas que contenham esses índices de texto completo pendentes.

Exemplos de resultado:

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
```

```
  "status": "OK",
  "description": "Error: The following tables contain dangling FULLTEXT index which
 is not supported. It is recommended to rebuild the table before upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_fts_index",
        "description": "Table `test.table_with_fts_index` contains dangling FULLTEXT
 index. Kindly recreate the table before upgrade."
      }
  ]
},
```

A pré-verificação relata um erro na tabela `test.table_with_fts_index` porque ela contém um índice de texto completo pendente. Para permitir que a atualização prossiga, recrie a tabela para limpar as tabelas auxiliares de índice de texto completo. Use `OPTIMIZE TABLE test.table_with_fts_index` ou `ALTER TABLE test.table_with_fts_index, ENGINE=INNODB`.

Depois que você recria a tabela, a pré-verificação é concluída com êxito.

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForDatafilePathInconsistency

Nível de pré-verificação: erro

Verifique a inconsistência relacionada ao caminho do arquivo **`ibd`**

Essa pré-verificação se aplica apenas ao Aurora MySQL versão 3.03.0 e anterior. Se você encontrar um erro com essa pré-verificação, faça a atualização para o Aurora MySQL versão 3.04 ou posterior.

Exemplos de resultado:

```
{
```

```
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForFtsSpaceIdZero

Nível de pré-verificação: erro

Verifique o índice de texto completo com o ID do espaço como zero

No MySQL, quando você adiciona um índice de texto completo a uma tabela do InnoDB, vários espaços de tabela de índice auxiliares são criados. Devido a um erro nas versões anteriores do MySQL, que foi corrigido na versão 5.6.20, era possível que essas tabelas de índice auxiliares fossem criadas no espaço de tabela do sistema, em vez de em seu próprio espaço de tabela do InnoDB.

Se existirem espaços de tabela auxiliares desse tipo, a atualização falhará. Recrie os índices de texto completo mencionados no erro de pré-verificação e tente fazer a atualização novamente.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForFtsSpaceIdZero",
  "title": "Check for fulltext index with space id as zero",
  "status": "OK",
  "description": "The auxiliary tables of FTS indexes on the table are created
 in system table-space. Due to this DDL queries executed on MySQL8.0 shall cause
 database unavailability. To avoid that, drop and recreate all the FTS indexes on
 the table or rebuild the table using ALTER TABLE query before the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.fts_space_zero_check",
        "description": " The auxiliary tables of FTS indexes on the table
 'test.fts_space_zero_check' are created in system table-space due to https://
bugs.mysql.com/bug.php?id=72132. In MySQL8.0, DDL queries executed on this table
 shall cause database unavailability. To avoid that, drop and recreate all the
 FTS indexes on the table or rebuild the table using ALTER TABLE query before the
 upgrade."
      }
```

```
  ]
},
```

A pré-verificação relata um erro na tabela `test.fts_space_zero_check`, pois ela tem tabelas auxiliares de pesquisa de texto completo (FTS) no espaço de tabela do sistema.

Depois que eliminar e recriar os índices de FTS associados a essa tabela, a pré-verificação será bem-sucedida.

> **Note**
>
> Antes de recriar espaços de tabela, consulte Partitioning Operations na documentação do MySQL para entender os efeitos do bloqueio e da movimentação de dados nas transações em primeiro plano.

```
{
 "id": "auroraUpgradeCheckForFtsSpaceIdZero",
 "title": "Check for fulltext index with space id as zero",
 "status": "OK",
 "detectedProblems": []
}
```

auroraUpgradeCheckForIncompleteXATransactions

Nível de pré-verificação: erro

Verifique se há transações XA no estado preparado

Ao executar o processo de atualização da versão principal, é essencial que a instância de banco de dados do Aurora MySQL versão 2 passe por um desligamento limpo. Isso garante que todas as transações sejam confirmadas ou revertidas e que o InnoDB tenha eliminado todos os registros de logs undo. Como a reversão de transações é necessária, se o banco de dados tiver alguma transação XA em um estado preparado, ele poderá impedir que o desligamento limpo continue. Por esse motivo, se alguma transação XA preparada for detectada, a atualização não poderá continuar enquanto você não tomar medidas para confirmá-la ou revertê-la.

Consulte mais informações sobre como encontrar transações de XA em um estado preparado usando `XA RECOVER` em XA Transaction SQL Statements na documentação do MySQL.

Consulte mais informações sobre estados de transações de XA em XA Transaction States na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "description": "Your cluster currently has XA transactions in the prepared state.
 To proceed with the upgrade, commit or rollback these transactions.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your cluster currently has XA transactions in the prepared
 state. To proceed with the upgrade, commit or rollback these transactions."
      }
  ]
}
```

Essa pré-verificação informa um erro porque há transações em um estado preparado que devem ser confirmadas ou revertidas.

Depois de fazer login no banco de dados, você pode verificar mais informações na tabela information_schema.innodb_trx e na saída `XA RECOVER`.

> **Important**
>
> Antes de confirmar ou reverter uma transação, recomendamos que você consulte a documentação do MySQL e os requisitos da aplicação.

```
mysql> select trx_started,
    trx_mysql_thread_id,
    trx_id,trx_state,
    trx_operation_state,
    trx_rows_modified,
    trx_rows_locked
from
    information_schema.innodb_trx;
```

```
+---------------------+---------------------+---------+-----------
+---------------------+------------------+-----------------+
| trx_started         | trx_mysql_thread_id | trx_id  | trx_state |
 trx_operation_state | trx_rows_modified | trx_rows_locked |
+---------------------+---------------------+---------+-----------
+---------------------+------------------+-----------------+
| 2024-08-12 01:09:39 |                   0 | 2849470 | RUNNING   | NULL
    |                 1 |               0 |
+---------------------+---------------------+---------+-----------
+---------------------+------------------+-----------------+
1 row in set (0.00 sec)

mysql> xa recover;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        1 |            6 |            0 | xatest |
+----------+--------------+--------------+--------+
1 row in set (0.00 sec)
```

Nesse caso, revertemos a transação preparada.

```
mysql> XA ROLLBACK 'xatest';
Query OK, 0 rows affected (0.00 sec)
v
mysql> xa recover;
Empty set (0.00 sec)
```

Depois que a transação XA for revertida, a pré-verificação será bem-sucedida.

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInstanceLimit

Nível de pré-verificação: erro

Verifique se a atualização é compatível com a classe de instância atual

No momento, não é possível fazer atualização no local do Aurora MySQL versão 2.12.0 ou 2.12.1, em que a classe de instância de banco de dados do gravador é db.r6i.32xlarge. Nesse caso, a pré-verificação exibe um erro. Para permitir que a atualização continue, você pode alterar a classe de instância de banco de dados para db.r6i.24xlarge ou menor. Ou você pode fazer a atualização para o Aurora MySQL versão 2.12.2 ou posterior, na qual a atualização no local para o Aurora MySQL versão 3 é compatível com db.r6i.32xlarge.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForInstanceLimit",
  "title": "Checks if upgrade is supported on the current instance class",
  "status": "OK",
  "description": "Upgrade from Aurora Version 2.12.0 and 2.12.1 may fail for 32.xl
 and above instance class.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Upgrade is not supported on this instance size for Aurora
 MySql Version 2.12.1. Before upgrading to Aurora MySql 3, please consider either:
 1. Changing the instance class to 24.xl or lower. -or- 2. Upgrading to patch
 version 2.12.2 or higher."
      }
  ]
},
```

A pré-verificação exibe um erro porque a instância de banco de dados do gravador está usando a classe de instância db.r6i.32xlarge e está sendo executada no Aurora MySQL versão 2.12.1.

auroraUpgradeCheckForInternalUsers

Nível de pré-verificação: erro

Verifique se há usuários internos da versão 8.0

Essa pré-verificação se aplica apenas ao Aurora MySQL versão 3.03.0 e anterior. Se você encontrar um erro com essa pré-verificação, faça a atualização para o Aurora MySQL versão 3.04 ou posterior.

Exemplos de resultado:

```
{
```

```
  "id": "auroraUpgradeCheckForInternalUsers",
  "title": "Check for 8.0 internal users.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForMasterUser

Nível de pré-verificação: erro

Verifique se o usuário principal do RDS existe

O MySQL 8 adicionou um novo modelo de privilégios com suporte para perfil e privilégios dinâmicos a fim de tornar o gerenciamento de privilégios mais fácil e refinado. Como parte dessa mudança, o Aurora MySQL introduziu o novo `rds_superuser_role`, que é concedido automaticamente ao usuário principal do banco de dados na atualização do Aurora MySQL versão 2 para a versão 3.

Consulte mais informações sobre os perfis e privilégios atribuídos ao usuário principal no Aurora MySQL em Privilégios da conta de usuário mestre. Consulte mais informações sobre o modelo de privilégio baseado em perfil do Aurora MySQL versão 3 em Modelo de privilégios baseados em funções.

Essa pré-verificação confere se o usuário principal existe no banco de dados. Se o usuário principal não existir, a pré-verificação falhará. Para permitir que a atualização continue, recrie o usuário principal redefinindo a senha dele ou criando o usuário manualmente. Tente realizar a atualização novamente. Consulte mais informações sobre como redefinir a senha do usuário principal em Alterar a senha do usuário mestre do banco de dados.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
  "description": "Throws error if master user has been dropped!",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/UsingWithRDS.MasterAccounts.html",
  "detectedProblems": [
      {
        "level": "Error",
```

```
        "dbObject": "all",
        "description": "Your Master User on host '%' has been dropped. To proceed
 with the upgrade, recreate the master user `reinvent` on default host '%'"
      }
  ]
}
```

Depois de redefinir sua senha do usuário principal, a pré-verificação será concluída com êxito e você poderá tentar fazer a atualização novamente.

O exemplo a seguir usa a AWS CLI para redefinir a senha. As alterações de senha são aplicadas imediatamente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --master-user-password my-new-password
```

Depois, a pré-verificação é bem-sucedida.

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  title": "Check if master user exists",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForPrefixIndexOnGeometryColumns

Nível de pré-verificação: erro

Verifique as colunas de geometria nos índices de prefixo

Desde o MySQL 8.0.12, você não pode mais criar um índice prefixado em uma coluna usando o tipo de dados GEOMETRY. Consulte mais informações em WL#11808.

Se algum desses índices existir, a atualização falhará. Para resolver o problema, descarte os índices prefixados `GEOMETRY` nas tabelas mencionadas na falha de pré-verificação.

Exemplos de resultado:

```
{
```

```
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "description": "Consider dropping the prefix Indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.geom_index_prefix",
        "description": "Table `test`.`geom_index_prefix` has an index `LatLon` on
 geometry column/s. Mysql 8.0 does not support this type of index on a geometry
 column https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run
 'DROP INDEX `LatLon` ON `test`.`geom_index_prefix`;"
      }
  ]
}
```

A pré-verificação relata um erro porque a tabela `test.geom_index_prefix` contém um índice com um prefixo em uma coluna `GEOMETRY`.

Depois que você descartar esse índice, a pré-verificação será bem-sucedida.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForSpecialCharactersInProcedures

Nível de pré-verificação: erro

Verifique se há inconsistência relacionada a caracteres especiais em procedimentos armazenados

Antes do MySQL 8.0, nomes de bancos de dados, nomes de tabelas e outros objetos correspondiam a arquivos no diretório de dados, ou seja, metadados baseados em arquivos. Como parte da atualização para o MySQL 8.0, todos os objetos do banco de dados são migrados para as novas tabelas internas do dicionário de dados que são armazenadas no esquema `mysql` para dar suporte ao recém-implementado Atomic Data Dictionary. Como parte da migração

de procedimentos armazenados, a definição e o corpo de cada procedimento são validados à medida que são inseridos no novo dicionário de dados.

Antes do MySQL 8, em alguns casos era possível criar rotinas armazenadas ou inserir diretamente na tabela `mysql.proc` procedimentos que continham caracteres especiais. Por exemplo, você podia criar um procedimento armazenado que continha um comentário com o caractere de espaço rígido incompatível `\xa0`. Se algum desses procedimentos for encontrado, a atualização falhará.

Essa pré-verificação confirma se o corpo e a definição de seus procedimentos armazenados não contêm esses caracteres. Para que a atualização continue, recrie esses procedimentos armazenados sem nenhum caractere oculto ou especial.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "description": "Following procedure(s) has special characters inconsistency.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "information_schema.routines",
        "description": "Data Dictionary Metadata is inconsistent for the procedure
 `get_version_proc` in the database `test` due to usage of special characters in
 procedure body. To avoid that, drop and recreate the procedure without any special
 characters before proceeding with the Upgrade."
      }
  ]
}
```

A pré-verificação informa que o cluster de banco de dados contém um procedimento chamado `get_version_proc` no banco de dados `test` que contém caracteres especiais no corpo do procedimento.

Depois que você descarta e recria o procedimento armazenado, a pré-verificação é bem-sucedida, permitindo que a atualização continue.

```
{
```

```
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForSysSchemaObjectTypeMismatch

Nível de pré-verificação: erro

Verifique a incompatibilidade do tipo de objeto para o esquema **`sys`**

O esquema sys é um conjunto de objetos e visualizações em um banco de dados do MySQL que ajuda os usuários a solucionar problemas em instâncias de banco de dados, bem como a otimizá-las e monitorá-las. Quando é feita uma atualização de uma versão principal do Aurora MySQL versão 2 para a versão 3, as visualizações do esquema `sys` são recriadas e atualizadas para as novas definições do Aurora MySQL versão 3.

Se, na atualização, algum objeto no esquema `sys` for definido usando mecanismos de armazenamento (`sys_config/BASE TABLE` em INFORMATION_SCHEMA.TABLES), em vez de visualizações, a atualização falhará. Essas tabelas podem ser encontradas na tabela `information_schema.tables`. Esse não é um comportamento esperado, mas em alguns casos pode ocorrer devido a um erro do usuário.

Essa pré-verificação valida todos os objetos do esquema `sys` para garantir que eles usem as definições de tabela corretas e que as visualizações não sejam definidas erroneamente como tabelas do InnoDB ou MyISAM. Para resolver o problema, corrija manualmente todos os objetos exibidos renomeando-os ou descartando-os. Tente realizar a atualização novamente.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "description": "Database contains objects with type mismatch for sys schema.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "sys.waits_global_by_latency",
```

```
        "description": "Your object sys.waits_global_by_latency has a type mismatch.
 To fix the inconsistency we recommend to rename or remove the object before
 upgrading (use RENAME TABLE command). "
      }
  ]
}
```

A pré-verificação informa que a visualização sys.waits_global_by_latency no esquema sys tem uma incompatibilidade de tipo que está impedindo que a atualização continue.

Depois de fazer login na instância de banco de dados, você pode ver que esse objeto está definido como uma tabela do InnoDB, quando deveria ser uma visualização.

```
mysql> show create table sys.waits_global_by_latency\G
*************************** 1. row ***************************
       Table: waits_global_by_latency
Create Table: CREATE TABLE `waits_global_by_latency` (
  `events` varchar(128) DEFAULT NULL,
  `total` bigint(20) unsigned DEFAULT NULL,
  `total_latency` text,
  `avg_latency` text,
  `max_latency` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Para resolver esse problema, podemos descartar e recriar a visualização com a definição correta ou renomeá-la. Durante o processo de atualização, ela será criada automaticamente com a definição correta da tabela.

```
mysql> RENAME TABLE sys.waits_global_by_latency to sys.waits_global_by_latency_old;
Query OK, 0 rows affected (0.01 sec)
```

Após isso, a pré-verificação é concluída com êxito.

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForViewColumnNameLength

Nível de pré-verificação: erro

Verifique o limite superior do nome da coluna na visualização

A extensão máxima permitida para um nome de coluna no MySQL é 64 caracteres. Antes do MySQL 8.0, em alguns casos era possível criar uma visualização com um nome de coluna com mais de 64 caracteres. Se alguma dessas visualizações existir na instância de banco de dados, um erro de pré-verificação será retornado e a atualização falhará. Para que a atualização continue, você deve recriar as visualizações em questão, garantindo que as extensões de coluna tenham menos de 64 caracteres. Tente realizar a atualização novamente.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "description": "Following view(s) has column(s) with length greater than 64.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject":
 "test.colname_view_test.col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pa
        "description": "View `test`.`colname_view_test`has column
 `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` with
 invalid column name length. To avoid Upgrade errors, view should be altered by
 renaming the column name so that its length is not 0 and doesn't exceed 64."
      }
  ]
}
```

A pré-verificação informa que a visualização `test.colname_view_test` contém uma coluna `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` que excede o tamanho máximo permitido de 64 caracteres.

Observando a definição da visualização, podemos ver a coluna incorreta.

```
mysql> desc `test`.`colname_view_test`;
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
```

```
| Field                                                                 | Type        |
 Null | Key | Default | Extra |
+-----------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| col1                                                                  | varchar(20) |
 YES  |     | NULL    |       |
| col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad | int(11)     |
 YES  |     | NULL    |       |
+-----------------------------------------------------------------------+-------------
+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Para que a atualização continue, recrie a visualização, garantindo que o tamanho da coluna não exceda 64 caracteres.

```
mysql> drop view `test`.`colname_view_test`;
Query OK, 0 rows affected (0.01 sec)

mysql> create view `test`.`colname_view_test`(col1, col2_nopad) as select inf,
 fodcol from test;
Query OK, 0 rows affected (0.01 sec)

mysql> desc `test`.`colname_view_test`;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| col1       | varchar(20) | YES  |     | NULL    |       |
| col2_nopad | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Após isso, a pré-verificação é bem-sucedida.

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckIndexLengthLimitOnTinyColumns

Nível de pré-verificação: erro

Verifique se há tabelas com índices definidos com uma extensão de prefixo com mais de 255 bytes em colunas minúsculas

Ao criar um índice em uma coluna usando um tipo de dados binário no MySQL, você deve adicionar uma extensão de prefixo na definição do índice. Antes do MySQL 8.0, em alguns casos era possível especificar uma extensão de prefixo superior ao tamanho máximo permitido desses tipos de dados. Um exemplo são as colunas `TINYTEXT` e `TINYBLOB`, em que o tamanho máximo de dados permitido é 255 bytes, mas prefixos de índice maiores eram permitidos. Consulte mais informações em InnoDB Limits na documentação do MySQL.

Se essa pré-verificação falhar, descarte o índice incorreto ou reduza a extensão do prefixo das colunas `TINYTEXT` e `TINYBLOB` do índice para menos de 255 bytes. Tente realizar a atualização novamente.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "description": "Prefix length of the indexes defined on tiny columns cannot
 exceed 255 bytes. With utf8mb4 char set, this limits the prefix length supported
 upto 63 characters only. A larger prefix length was allowed in MySQL5.7 using
 innodb_large_prefix parameter. This parameter is deprecated in MySQL 8.0.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html,
 https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.tintxt_prefixed_index.col1",
        "description": "Index 'PRIMARY' on tinytext/tinyblob column `col1` of table
 `test.tintxt_prefixed_index` is defined with prefix length exceeding 255 bytes.
 Reduce the prefix length to <=255 bytes depending on character set used. For
 utf8mb4, it should be <=63."
      }
  ]
}
```

A pré-verificação relata um erro referente à tabela `test.tintxt_prefixed_index`, porque ela tem um índice `PRIMARY` com um prefixo com mais de 255 bytes em uma coluna TINYTEXT ou TINYBLOB.

Observando a definição dessa tabela, podemos ver que a chave primária tem um prefixo de 65 na coluna TINYTEXT `col1`. Como a tabela é definida usando o conjunto de caracteres `utf8mb4`, que armazena 4 bytes por caractere, o prefixo excede o limite de 255 bytes.

```
mysql> show create table `test`.`tintxt_prefixed_index`\G
*************************** 1. row ***************************
       Table: tintxt_prefixed_index
Create Table: CREATE TABLE `tintxt_prefixed_index` (
  `col1` tinytext NOT NULL,
  `col2` tinytext,
  `col_id` tinytext,
  PRIMARY KEY (`col1`(65))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC
1 row in set (0.00 sec)
```

Modificar o prefixo do índice para 63 ao usar o conjunto de caracteres `utf8mb4` permitirá que a atualização continue.

```
mysql> alter table `test`.`tintxt_prefixed_index` drop PRIMARY KEY, ADD  PRIMARY KEY
 (`col1`(63));
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Após isso, a pré-verificação é bem-sucedida.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

Nível de pré-verificação: erro

Verifique a inconsistência dos metadados do InnoDB ausentes na tabela **`mysql.host`**

Essa é uma pré-verificação somente interna realizada pelo serviço RDS. Os erros serão gerenciados automaticamente na atualização e podem ser ignorados com segurança.

Se você encontrar algum erro nessa pré-verificação, abra um caso no AWS Support para solicitar que a inconsistência de metadados seja resolvida. Ou você pode tentar fazer a atualização novamente executando um despejo lógico e, depois, restaurando para um novo cluster de banco de dados do Aurora MySQL versão 3.

auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

Nível de pré-verificação: erro

Verifique se há comentários de coluna utf8mb3 inválidos

Essa pré-verificação identifica tabelas que contêm comentários de coluna com codificação de caracteres utf8mb3 inválida. No MySQL 8.0, uma validação mais rigorosa é aplicada à codificação de caracteres nos metadados, incluindo comentários de coluna. Se algum comentário de coluna contiver caracteres que não sejam válidos no conjunto de caracteres utf8mb3, a atualização falhará.

A causa mais comum desse problema é a presença de caracteres fora do plano multilíngue básico (BMP) nos comentários de coluna. Os caracteres fora do BMP exigem codificação UTF-8 de 4 bytes (utf8mb4), mas utf8mb3 aceita apenas sequências de 3 bytes, que não podem representar esses caracteres.

Para resolver esse problema, antes de tentar fazer a atualização, você deve modificar os comentários de coluna para remover ou substituir quaisquer caracteres fora do BMP. Você pode usar a instrução `ALTER TABLE` para atualizar os comentários de coluna.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "description": "Following table(s) has/have invalid utf8mb3 comments on the
 column/columns.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.t2",
        "description": "Table test.t2 has invalid utf8mb3 comments in it's column/
columns. This is due to non-BMP characters in the comment field. To fix the
 inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
```

```
      }
  ]
}
```

A pré-verificação informa que a tabela `test.t2` contém caracteres utf8mb3 inválidos em um ou mais comentários de coluna, especificamente devido à presença de caracteres fora do BMP.

Para resolver esse problema, você pode identificar as colunas problemáticas e atualizar os respectivos comentários. Primeiro, examine a estrutura da tabela para identificar colunas com comentários:

```
mysql> SHOW CREATE TABLE test.t2\G
```

Depois de identificar as colunas com comentários problemáticos, atualize-as usando a instrução `ALTER TABLE`. Por exemplo:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT 'Updated
 comment without non-BMP characters';
```

Também é possível remover o comentário completamente:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT '';
```

Depois de atualizar todos os comentários problemáticos da coluna, a pré-verificação será aprovada e a atualização poderá prosseguir:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "detectedProblems": []
}
```

> **Note**
>
> Antes de modificar os comentários da coluna, você deve atualizar adequadamente qualquer código ou documentação da aplicação que dependa desses comentários. Considere migrar para o conjunto de caracteres utf8mb4 para obter melhor suporte a Unicode se sua aplicação exigir caracteres fora do BMP.

Avisos

As pré-verificações a seguir geram avisos quando a pré-verificação falha e a atualização pode continuar.

Tópicos

- Pré-verificações do MySQL que exibem avisos
- Pré-verificações do Aurora MySQL que exibem avisos

Pré-verificações do MySQL que exibem avisos

As seguintes pré-verificações são do MySQL Community:

- defaultAuthenticationPlugin
- maxdbFlagCheck
- mysqlDollarSignNameCheck
- reservedKeywordsCheck
- utf8mb3Check
- zeroDatesCheck

defaultAuthenticationPlugin

Nível de pré-verificação: aviso

Considerações sobre o novo plug-in de autenticação padrão

No MySQL 8.0, o plug-in de autenticação `caching_sha2_password` foi introduzido, fornecendo criptografia de senha mais segura e melhor desempenho do que o plug-in obsoleto `mysql_native_password`. No Aurora MySQL versão 3, o plug-in de autenticação padrão usado para usuários do banco de dados é o `mysql_native_password`.

Essa pré-verificação avisa que esse plug-in será removido e o padrão alterado em uma futura versão principal. Considere avaliar a compatibilidade dos clientes e usuários da aplicação antes dessa mudança.

Consulte mais informações em caching_sha2_password Compatibility Issues and Solutions na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "defaultAuthenticationPlugin",
  "title": "New default authentication plugin considerations",
  "description": "Warning: The new default authentication plugin
 'caching_sha2_password' offers more secure password hashing than previously
 used 'mysql_native_password' (and consequent improved client connection
 authentication). However, it also has compatibility implications that
 may affect existing MySQL installations. If your MySQL installation
 must serve pre-8.0 clients and you encounter compatibility issues after
 upgrading, the simplest way to address those issues is to reconfigure
 the server to revert to the previous default authentication plugin
 (mysql_native_password). For example, use these lines in the server option file:
\n\n[mysqld]\ndefault_authentication_plugin=mysql_native_password\n\nHowever, the
 setting should be viewed as temporary, not as a long term or permanent solution,
 because it causes new accounts created with the setting in effect to forego the
 improved authentication security.\nIf you are using replication please take time to
 understand how the authentication plugin changes may impact you.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-caching-sha2-password-compatibility-issues\nhttps://
dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-
sha2-password-replication"
},
```

maxdbFlagCheck

Nível de pré-verificação: aviso

Uso do sinalizador obsoleto **`MAXDB` `sql_mode`**

No MySQL 8.0, várias opções obsoletas de variáveis de sistema sql_mode foram removidas, como a `MAXDB`. Essa pré-verificação examina todas as sessões atualmente conectadas, bem como rotinas e gatilhos, para garantir que nenhuma tenha `sql_mode` definido para qualquer combinação que inclua `MAXDB`.

Exemplos de resultado:

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
```

```
  "description": "Warning: The following DB objects have the obsolete MAXDB
 option persisted for sql_mode, which will be cleared during the upgrade. It can
 potentially change the datatype DATETIME into TIMESTAMP if it is used inside
 object's definition, and this in turn can change the behavior in case of dates
 earlier than 1970 or later than 2037. If this is a concern, please redefine these
 objects so that they do not rely on the MAXDB flag before running the upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.maxdb_stored_routine",
        "description": "PROCEDURE uses obsolete MAXDB sql_mode",
        "dbObjectType": "Routine"
      }
  ]
}
```

A pré-verificação informa que a rotina `test.maxdb_stored_routine` contém uma opção `sql_mode` incompatível.

Depois de fazer login no banco de dados, você pode ver na definição de rotina que `sql_mode` contém `MAXDB`.

```
 > SHOW CREATE PROCEDURE test.maxdb_stored_routine\G

*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode:
 PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,MAXDB,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPT
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Para resolver o problema, recrie o procedimento depois de definir o `sql_mode` correto no cliente.

> **Note**
>
> De acordo com a documentação do MySQL, o MySQL armazena a configuração `sql_mode` em vigor quando uma rotina é criada ou alterada. Ele sempre executa a rotina com essa configuração, independentemente da configuração `sql_mode` quando a rotina é iniciada.
> Antes de alterar `sql_mode`, consulte Server SQL Modes na documentação do MySQL. Avalie cuidadosamente qualquer possível impacto de fazer isso na aplicação.

Recrie o procedimento sem o `sql_mode` incompatível.

```
mysql > set sql_mode='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql > DROP PROCEDURE test.maxdb_stored_routine\G
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER $$
mysql >
mysql > CREATE PROCEDURE test.maxdb_stored_routine()
    -> SQL SECURITY DEFINER
    -> BEGIN
    ->     SELECT * FROM test;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER ;
mysql > show create procedure test.maxdb_stored_routine\G
*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

```
1 row in set (0.00 sec)
```

A pré-verificação é bem-sucedida.

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "detectedProblems": []
}
```

mysqlDollarSignNameCheck

Nível de pré-verificação: aviso

Verifique se estão sendo usados cifrões únicos obsoletos em nomes de objeto

Desde o MySQL 8.0.32, o uso do cifrão ($) como o primeiro caractere de um identificador sem aspas foi descontinuado. Se você tiver esquemas, tabelas, visualizações, colunas ou rotinas contendo $ como primeiro caractere, essa pré-verificação exibirá um aviso. Embora esse aviso não impeça a continuidade da atualização, recomendamos que você tome medidas o quanto antes para resolver isso. A partir do MySQL 8.4, qualquer um desses identificadores retornará um erro de sintaxe em vez de um aviso.

Exemplos de resultado:

```
{
  "id": "mysqlDollarSignNameCheck",
  "title": "Check for deprecated usage of single dollar signs in object names",
  "status": "OK",
  "description": "Warning: The following objects have names with deprecated usage
 of dollar sign ($) at the begining of the identifier. To correct this warning,
 ensure, that names starting with dollar sign, also end with it, similary to quotes
 ($example$). ",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.$deprecated_syntx",
        "description": " name starts with $ sign."
      }
  ]
},
```

A pré-verificação exibe um aviso porque a tabela `$deprecated_syntx` no esquema `test` contém um $ como o primeiro caractere.

reservedKeywordsCheck

Nível de pré-verificação: aviso

Uso de objetos de banco de dados com nomes conflitantes com novas palavras-chave reservadas

Essa verificação é semelhante à routineSyntaxCheck, pois verifica o uso de objetos de banco de dados com nomes conflitantes com novas palavras-chave reservadas. Embora eles não impeçam as atualizações, você precisa avaliar os avisos com cuidado.

Exemplos de resultado:

Usando o exemplo anterior com a tabela chamada `except`, a pré-verificação exibe um aviso:

```
{
  "id": "reservedKeywordsCheck",
  "title": "Usage of db objects with names conflicting with new reserved keywords",
  "status": "OK",
  "description": "Warning: The following objects have names that conflict with
 new reserved keywords. Ensure queries sent by your applications use `quotes` when
 referring to them or they will result in errors.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.except",
        "description": "Table name",
        "dbObjectType": "Table"
      }
  ]
}
```

Esse aviso permite que você saiba que pode haver algumas consultas de aplicação a serem analisadas. Se as consultas da aplicação não tiverem o escape dos literais de string correto, elas poderão apresentar erros após a atualização para o MySQL 8.0. Analise as aplicações para confirmar, testando com base em um clone ou snapshot do cluster de banco de dados do Aurora MySQL em execução na versão 3.

Exemplo de uma consulta de aplicação sem escape que falhará após a atualização:

```
SELECT * FROM escape;
```

Exemplo de uma consulta de aplicação com escape correto que será bem-sucedida após a atualização:

```
SELECT * FROM 'escape';
```

utf8mb3Check

Nível de pré-verificação: aviso

Uso do conjunto de caracteres **`utf8mb3`**

No MySQL 8.0, o conjunto de caracteres `utf8mb3` foi descontinuado e será removido em uma futura versão principal do MySQL. Essa pré-verificação é implementada para gerar um aviso se algum objeto de banco de dados que usa esse conjunto de caracteres for detectado. Embora isso não impeça a continuidade de uma atualização, é altamente recomendável que você avalie a possibilidade de migrar as tabelas para o conjunto de caracteres `utf8mb4`, que é o padrão a partir do MySQL 8.0. Consulte mais informações sobre utf8mb3 e utf8mb4 em Converting Between 3-Byte and 4-Byte Unicode Character Sets na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "utf8mb3",
  "title": "Usage of utf8mb3 charset",
  "status": "OK",
  "description": "Warning: The following objects use the deprecated utf8mb3
 character set. It is recommended to convert them to use utf8mb4 instead, for
 improved Unicode support. The utf8mb3 character is subject to removal in the
 future.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-
utf8mb3.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.t1.col1",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      },
      {
```

```
        "level": "Warning",
        "dbObject": "test.t1.col2",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      }
  ]
}
```

Para resolver esse problema, recrie as tabelas e os objetos referidos. Antes de fazer isso, consulte mais informações e pré-requisitos em Converting Between 3-Byte And 4-Byte Unicode Character Sets na documentação do MySQL.

zeroDatesCheck

Nível de pré-verificação: aviso

Valores de data zero, data e hora e carimbo de data/hora

O MySQL agora impõe regras mais rígidas em relação ao uso de valores zero nas colunas de data, data e hora e carimbo de data/hora. Recomendamos que você use os modos `NO_ZERO_IN_DATE` e `NO_ZERO_DATE` SQL em conjunto com o modo `strict`, pois eles serão mesclados com o modo `strict` em uma versão futura do MySQL.

Se a configuração `sql_mode` de qualquer uma de suas conexões de banco de dados, no momento da execução da pré-verificação, não incluir esses modos, um aviso será gerado na pré-verificação. Talvez os usuários ainda consigam inserir valores de data, data e hora e carimbo de data/hora contendo valores zero. No entanto, é altamente recomendável substituir valores zero por valores válidos, pois no futuro eles podem mudar de comportamento e não funcionar corretamente. Como é um aviso, isso não impede as atualizações, mas recomendamos que você comece a pensar em agir.

Exemplos de resultado:

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
```

```
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
  ]
}
```

Pré-verificações do Aurora MySQL que exibem avisos

As pré-verificações a seguir são específicas do Aurora MySQL:

- auroraUpgradeCheckForRollbackSegmentHistoryLength
- auroraUpgradeCheckForUncommittedRowModifications

auroraUpgradeCheckForRollbackSegmentHistoryLength

Nível de pré-verificação: aviso

Verifica se a extensão da lista de histórico do segmento de reversão para o cluster é alto

Como mencionado em auroraUpgradeCheckForIncompleteXATransactions, ao executar o processo de atualização da versão principal, é essencial que a instância de banco de dados do Aurora MySQL versão 2 passe por um desligamento limpo. Isso garante que todas as transações sejam confirmadas ou revertidas e que o InnoDB tenha eliminado todos os registros de logs undo.

Se o tamanho da lista de histórico (HLL) de segmentos de reversão do cluster de banco de dados for grande, isso poderá prolongar o tempo que o InnoDB leva para concluir a limpeza dos registros de log undo, aumentando o tempo de inatividade durante o processo de atualização da versão principal. Se a pré-verificação detectar que o valor de HLL no cluster de banco de

dados está alto, ela emitirá um aviso. Embora isso não impeça que a atualização continue, recomendamos que você monitore de perto o HLL no cluster de banco de dados. Mantê-lo em níveis baixos reduz o tempo de inatividade necessário durante a atualização de uma versão principal. Consulte mais informações sobre o monitoramento de HLL em O tamanho da lista de histórico do InnoDB aumentou significativamente.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForRollbackSegmentHistoryLength",
  "title": "Checks if the rollback segment history length for the cluster is high",
  "status": "OK",
  "description": "Rollback Segment History length is greater than 1M. Upgrade may
 take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_metrics",
        "description": "The InnoDB undo history list length('trx_rseg_history_len')
 is 82989114. Upgrade may take longer due to purging of undo information for old row
 versions."
      }
  ]
}
```

A pré-verificação exibe um aviso porque detectou que o valor de HLL de undo do InnoDB estava alto no cluster de banco de dados (82989114). Embora a atualização continue, dependendo da quantidade de undo a limpar, isso pode estender o tempo de inatividade necessário durante o processo de atualização.

Recomendamos que você investigue as transações abertas no cluster de banco de dados antes de executar atualização na produção, para garantir que o HLL seja mantido em um valor gerenciável.

auroraUpgradeCheckForUncommittedRowModifications

Nível de pré-verificação: aviso

Verifica se há muitas modificações de linha não confirmadas

Como mencionado em auroraUpgradeCheckForIncompleteXATransactions, ao executar o processo de atualização da versão principal, é essencial que a instância de banco de dados do

Aurora MySQL versão 2 passe por um desligamento limpo. Isso garante que todas as transações sejam confirmadas ou revertidas e que o InnoDB tenha eliminado todos os registros de logs undo.

Se o cluster de banco de dados tiver transações que modificaram um grande número de linhas, isso poderá prolongar o tempo que o InnoDB leva para concluir uma reversão dessa transação como parte do processo de desligamento limpo. Se a pré-verificação encontrar transações de longa execução, com um grande número de linhas modificadas no cluster de banco de dados, ela gerará um aviso. Embora isso não impeça que a atualização continue, recomendamos que você monitore de perto o tamanho das transações ativas no cluster de banco de dados. Mantê-lo em níveis baixos reduz o tempo de inatividade necessário durante a atualização de uma versão principal.

Exemplos de resultado:

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "description": "Database contains uncommitted row changes greater than 10M.
 Upgrade may take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_trx",
        "description": "The database contains 11000000 uncommitted row change(s) in
 1 transaction(s). Upgrade may take longer due to transaction rollback."
      }
  ]
},
```

A pré-verificação informa que o cluster de banco de dados contém uma transação com 11 milhões de alterações de linha não confirmadas que precisarão ser revertidas durante o processo de desligamento limpo. A atualização continuará, mas para reduzir o tempo de inatividade durante o processo, recomendamos que você monitore e investigue isso antes de realizar a atualização nos clusters de produção.

Para visualizar transações ativas na instância de banco de dados do gravador, você pode usar a tabela information_schema.innodb_trx. A consulta a seguir na instância de banco de dados do gravador mostra as transações atuais, o tempo de execução, o estado e as linhas modificadas do cluster de banco de dados.

```
# Example of uncommitted transaction
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state | rows_modified |
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
| 2024-08-12 18:32:52 |                          1592 |
 20041 | 52866130 | RUNNING   |      11000000 |
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
1 row in set (0.01 sec)

# Example of transaction rolling back
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+------------------------------
+---------------------------------+----------+--------------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state    | rows_modified |
+---------------------+------------------------------
+---------------------------------+----------+--------------+---------------+
| 2024-08-12 18:32:52 |                          1719 |
 20041 | 52866130 | ROLLING BACK |      10680479 |
+---------------------+------------------------------
+---------------------------------+----------+--------------+---------------+
1 row in set (0.01 sec)
```

Depois que a transação é confirmada ou revertida, a pré-verificação não exibe mais um aviso. Consulte a documentação do MySQL e sua equipe de aplicação antes de reverter qualquer

transação grande, pois a conclusão da reversão pode levar algum tempo, dependendo do tamanho da transação.

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "detectedProblems": []
},
```

Consulte mais informações sobre como otimizar o gerenciamento de transações do InnoDB e o possível impacto da execução e reversão de grandes transações em instâncias do banco de dados do MySQL em Optimizing InnoDB Transaction Management na documentação do MySQL.

Avisos

A pré-verificação a seguir gera um aviso quando a pré-visualização falha e a atualização pode continuar.

sqlModeFlagCheck

Nível de pré-verificação: aviso

Uso dos sinalizadores obsoletos **`sql_mode`**

Além de `MAXDB`, várias outras opções de `sql_mode` foram removidas: `DB2`, `MSSQL`, `MYSQL323`, `MYSQL40`, `ORACLE`, `POSTGRESQL`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS` e `NO_TABLE_OPTIONS`. A partir do MySQL 8.0, nenhum desses valores pode ser atribuído à variável de sistema `sql_mode`. Se essa pré-verificação encontrar alguma sessão aberta usando essas configurações de `sql_mode`, garanta que os grupos de parâmetros da instância e do cluster de banco de dados, bem como as configurações e aplicações cliente, estejam atualizados para desativá-las. Consulte mais informações na documentação do MySQL.

Exemplos de resultado:

```
{
  "id": "sqlModeFlagCheck",
  "title": "Usage of obsolete sql_mode flags",
  "status": "OK",
  "detectedProblems": []
```

```
}
```

Para resolver qualquer uma dessas falhas de pré-verificação, consulte maxdbFlagCheck.

Erros, avisos ou alertas

A pré-verificação a seguir pode exibir um erro, aviso ou alerta, dependendo da saída da pré-verificação.

checkTableOutput

Nível de pré-verificação: erro, aviso ou alerta

Problemas relatados pelo comando **`check table x for upgrade`**

Antes de iniciar a atualização para o Aurora MySQL versão 3, `check table for upgrade` é executado em cada tabela nos esquemas do usuário no cluster de banco de dados. Essa pré-verificação não é a mesma que checkTableMysqlSchema.

O comando `check table for upgrade` examina as tabelas em busca de possíveis problemas que possam surgir durante uma atualização para uma versão mais recente do MySQL. Executar esse comando antes de tentar fazer a atualização pode ajudar a identificar e resolver quaisquer incompatibilidades com antecedência, facilitando o respectivo processo.

Esse comando executa várias verificações em cada tabela, como as seguintes:

- Verificar se os metadados e a estrutura da tabela são compatíveis com a versão de destino do MySQL
- Verificar se há algum recurso obsoleto ou removido usado pela tabela
- Garantir que a atualização da tabela seja feito adequadamente sem perda de dados

Ao contrário de outras pré-verificações, ela pode exibir um erro, aviso ou alerta, dependendo da saída `check table`. Se essa pré-verificação exibir alguma tabela, examine-a cuidadosamente, assim como o código de retorno e a mensagem, antes de iniciar a atualização. Consulte mais informações em CHECK TABLE Statement na documentação do MySQL.

Aqui, fornecemos um exemplo de erro e um exemplo de aviso.

Exemplo de erro:

```
{
```

```
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.parent",
        "description": "Table 'test.parent' doesn't exist"
      }
  ]
},
```

A pré-verificação relata um erro de que a tabela `test.parent` não existe.

O arquivo `mysql-error.log` da instância de banco de dados do gravador mostra que há um erro de chave externa.

```
2024-08-13T15:32:10.676893Z 62 [Warning] InnoDB: Load table `test`.`parent` failed,
 the table has missing foreign key indexes. Turn off 'foreign_key_checks' and try
 again.
2024-08-13T15:32:10.676905Z 62 [Warning] InnoDB: Cannot open table test/parent from
 the internal data dictionary of InnoDB though the .frm file for the table exists.
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
 for how to resolve the issue.
```

Faça login na instância de banco de dados do gravador e execute `show engine innodb status\G` para ter mais informações sobre o erro de chave externa.

```
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2024-08-13 15:33:33 0x14ef7b8a1700 INNODB MONITOR OUTPUT
=====================================
.
.
.
------------------------
LATEST FOREIGN KEY ERROR
------------------------
```

```
2024-08-13 15:32:10 0x14ef6dbbb700 Error in foreign key constraint of table test/
child:
there is no index in referenced table which would contain
the columns as the first columns, or the data types in the
referenced table do not match the ones in table. Constraint:
,
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
The index in the foreign key in table is p_name_idx
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-
constraints.html for correct foreign key definition.
.
.
```

A mensagem `LATEST FOREIGN KEY ERROR` informa que a restrição de chave externa `fk_pname` na tabela `test.child`, que faz referência à tabela `test.parent`, tem um índice ausente ou uma incompatibilidade de tipo de dados. A documentação do MySQL sobre restrições de chave externa menciona que as colunas referidas em uma chave externa devem ter um índice associado e que as colunas principal/secundárias devem usar o mesmo tipo de dados.

Para verificar se isso está relacionado a um índice ausente ou a uma incompatibilidade de tipo de dados, faça login no banco de dados e verifique as definições da tabela desativando temporariamente a variável de sessão foreign_key_checks. Depois de fazer isso, podemos ver que a restrição secundária em questão (`fk_pname`) usa `p_name varchar(20) CHARACTER SET latin1 DEFAULT NULL` para se referir à tabela principal `name varchar(20) NOT NULL`. A tabela principal usa `DEFAULT CHARSET=utf8`, mas a coluna `p_name` da tabela secundária usa `latin1`, então o erro de incompatibilidade do tipo de dados é gerado.

```
mysql> show create table parent\G
ERROR 1146 (42S02): Table 'test.parent' doesn't exist

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

```
mysql> set foreign_key_checks=0;
Query OK, 0 rows affected (0.00 sec)

mysql> show create table parent\G
*************************** 1. row ***************************
       Table: parent
Create Table: CREATE TABLE `parent` (
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Para resolver esse problema, podemos alterar a tabela secundária para que use o mesmo conjunto de caracteres da tabela principal ou alterar a tabela principal para que use o mesmo conjunto de caracteres da tabela secundária. Aqui, como a tabela secundária usa explicitamente `latin1` na definição da coluna `p_name`, executamos `ALTER TABLE` para modificar o conjunto de caracteres para `utf8`.

```
mysql> alter table child modify p_name varchar(20) character set utf8 DEFAULT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> flush tables;
Query OK, 0 rows affected (0.01 sec)
```

Depois disso, a pré-verificação é concluída com êxito e a atualização pode continuar.

Exemplo de aviso:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.orders",
        "description": "Trigger test.orders.delete_audit_trigg does not have CREATED
 attribute."
      }
  ]
}
```

A pré-verificação exibe um aviso para o gatilho `delete_audit_trigg` na tabela `test.orders` porque ela não tem um atributo `CREATED`. De acordo com Checking Version Compatibility na documentação do MySQL, essa mensagem é informativa e é impressa para gatilhos criados antes do MySQL 5.7.2.

Como é um aviso, isso não impede que a atualização continue. No entanto, se quiser resolver o problema, você pode recriar o gatilho em questão e a pré-verificação será bem-sucedida sem nenhum aviso.

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": []
},
```

Como realizar uma atualização local

Recomendamos revisar o material de base em Como funciona a atualização da versão principal no local Aurora MySQL.

Execute qualquer planejamento e teste de pré-atualização, conforme descrito em Planejando uma atualização de versão principal para um cluster de Aurora MySQL.

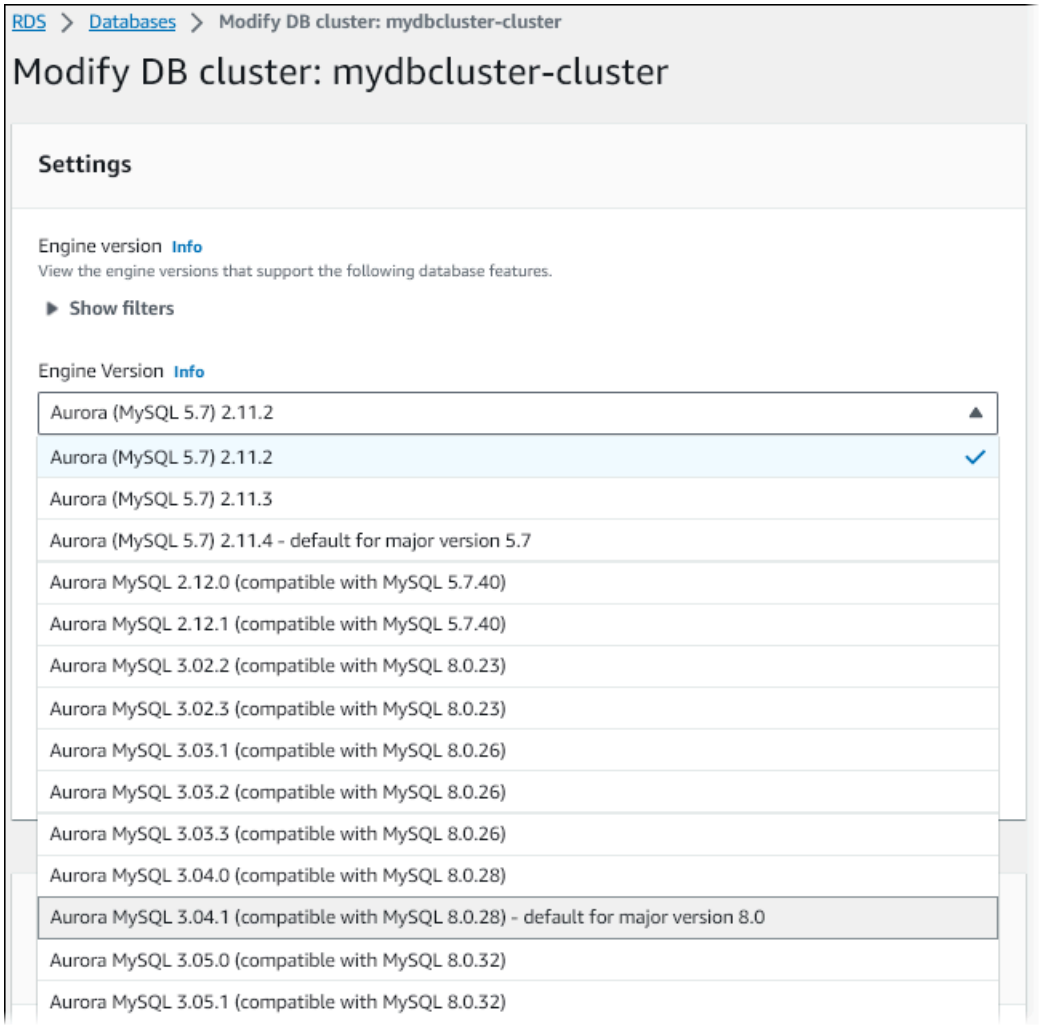
Console

O exemplo a seguir atualiza o cluster de banco de dados `mydbcluster-cluster` para o Aurora MySQL versão 3.04.1.

Para atualizar a versão principal de um cluster de bancos de dados Aurora MySQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Se você usou um grupo de parâmetros personalizado com o cluster de banco de dados original, crie um grupo de parâmetros correspondente compatível com a nova versão principal. Faça todos os ajustes necessários aos parâmetros de configuração nesse novo grupo de parâmetro. Para ter mais informações, consulte Como as atualizações locais afetam os grupos de parâmetros de um cluster.
3. No painel de navegação, escolha Databases (Bancos de dados).
4. Na lista, escolha o cluster de banco de dados que deseja modificar.
5. Selecione Modify.
6. Em Version (Versão), selecione uma versão principal do Aurora MySQL.

   Recomendamos usar a versão secundária mais recente da versão principal. Aqui, escolhemos a versão padrão atual.



7. Escolha Continue.
8. Na próxima página, especifique quando executar a atualização. Escolha During the next scheduled maintenance window (Durante a próxima janela de manutenção programada) ou Immediately (Imediatamente).
9. (Opcional) Examine periodicamente a página Events (Eventos) no console do RDS durante a atualização. Isso ajuda você a monitorar o progresso da atualização e identificar quaisquer problemas. Se a atualização encontrar quaisquer problemas, consulte Solução de problemas para atualização no local de Aurora MySQL para as etapas a serem realizadas.
10. Se você criou um grupo de parâmetros no início deste procedimento, associe o grupo de parâmetros personalizado ao cluster atualizado. Para ter mais informações, consulte Como as atualizações locais afetam os grupos de parâmetros de um cluster.

> **Note**
>
> A execução desta etapa exige que você reinicie o cluster novamente para aplicar o novo grupo de parâmetro.

11. (Opcional) Depois de concluir qualquer teste pós-atualização, exclua o snapshot manual que Aurora criado no início da atualização.

AWS CLI

Para atualizar a versão principal de um cluster de bancos de dados Aurora MySQL, use o comando da AWS CLI modify-db-cluster com os seguintes parâmetros obrigatórios:

- `--db-cluster-identifier`
- `--engine-version`
- `--allow-major-version-upgrade`
- `--apply-immediately` ou `--no-apply-immediately`

Se o cluster usar quaisquer grupos de parâmetros personalizados, inclua também uma ou ambas as opções a seguir:

- `--db-cluster-parameter-group-name`, se o cluster usar um grupo de parâmetros de cluster personalizado
- `--db-instance-parameter-group-name`, se alguma instância no cluster usar um grupo de parâmetro de banco de dados personalizado

O exemplo a seguir atualiza o cluster de banco de dados `sample-cluster` para o Aurora MySQL versão 3.04.1. A atualização acontece imediatamente, em vez de aguardar a próxima janela de manutenção.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --engine-version 8.0.mysql_aurora.3.04.1 \
```

```
          --allow-major-version-upgrade \
          --apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
          --db-cluster-identifier sample-cluster ^
          --engine-version 8.0.mysql_aurora.3.04.1 ^
          --allow-major-version-upgrade ^
          --apply-immediately
```

Você pode combinar outros comandos CLI com `modify-db-cluster` para criar um processo completo automatizado para executar e verificar atualizações. Para ter mais informações e exemplos, consulte Tutorial de atualização no local de Aurora MySQL.

> **Note**
>
> Se o cluster fizer parte de um banco de dados Aurora global, o procedimento de atualização local será ligeiramente diferente. Você chama a operação de comando modify-global-cluster em vez de `modify-db-cluster`. Para ter mais informações, consulte Principais atualizações no local para bancos de dados globais.

API do RDS

Para atualizar a versão principal de um cluster de banco de dados do Aurora MySQL, use a operação ModifyDBCluster da API do RDS com os seguintes parâmetros necessários:

- `DBClusterIdentifier`
- `Engine`
- `EngineVersion`
- `AllowMajorVersionUpgrade`
- `ApplyImmediately` (definido como `true` ou `false`)

> **Note**
>
> Se o cluster fizer parte de um banco de dados Aurora global, o procedimento de atualização local será ligeiramente diferente. Você chama a operação ModifyGlobalCluster em vez de

`ModifyDBCluster`. Para ter mais informações, consulte Principais atualizações no local para bancos de dados globais.

Como as atualizações locais afetam os grupos de parâmetros de um cluster

Os grupos de parâmetros do Aurora têm diferentes conjuntos de configurações para clusters que são compatíveis com MySQL 5.7 ou 8.0. Quando você realiza uma atualização no local, o cluster atualizado e todas as instâncias devem usar grupos de parâmetros de instância e cluster correspondentes:

Seu cluster e suas instâncias podem usar os grupos de parâmetros padrão compatíveis com 5.7. Em caso afirmativo, o cluster atualizado e a instância iniciarão com os grupos de parâmetros padrão compatíveis com 8.0. Se o cluster e as instâncias usarem quaisquer grupos de parâmetros personalizados, crie grupos de parâmetros correspondentes ou compatíveis com 8.0. Além disso, especifique-os durante o processo de atualização.

Note

Para a maioria das configurações de parâmetros, é possível selecionar o grupo de parâmetros personalizado em dois pontos. É quando você cria o cluster ou associa o grupo de parâmetros ao cluster mais tarde.
No entanto, se você utilizar uma configuração não padrão para o parâmetro `lower_case_table_names`, deverá configurar o grupo de parâmetros personalizado com essa configuração com antecedência. Em seguida, especifique o grupo de parâmetros ao realizar a restauração do snapshot para criar o cluster. Qualquer alteração no parâmetro `lower_case_table_names` não terá efeito após a criação do cluster.
Recomendamos que você use a mesma configuração para `lower_case_table_names` ao atualizar do Aurora MySQL versão 2 para a versão 3.
Com um banco de dados Aurora global baseado no Aurora MySQL, você não poderá executar uma atualização no local do Aurora MySQL versão 2 para a versão 3 se o parâmetro `lower_case_table_names` estiver ativado. Para ter mais informações sobre os métodos que você pode usar, consulte Atualizações de versão principal.

> **Important**
>
> Se você especificar qualquer grupo de parâmetros personalizado durante o processo de atualização, será necessário reinicializar manualmente o cluster após a conclusão da atualização. Isso faz com que o cluster comece a usar suas configurações de parâmetros personalizadas.

Alterações nas propriedades do cluster entre as versões do Aurora MySQL

Ao realizar o upgrade do Aurora MySQL versão 2 para a versão 3, verifique todas as aplicações ou scripts utilizados para configurar ou gerenciar clusters e instâncias de banco de dados do Aurora MySQL.

Além disso, altere seu código que manipula grupos de parâmetros para explicar o fato de que os nomes dos grupos de parâmetros padrão são diferentes para os clusters compatíveis com as versões 5.7 e 8.0. Os nomes dos grupos de parâmetros padrão para clusters do Aurora MySQL versão 2 e 3 são `default.aurora-mysql5.7` e `default.aurora-mysql8.0`, respectivamente.

Por exemplo, você pode ter um código como abaixo que se aplica ao cluster antes de uma atualização.

```
# Check the default parameter values for MySQL 5.7-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql5.7 --
region us-east-1
```

Depois de atualizar a versão principal do cluster, altere esse código da seguinte forma.

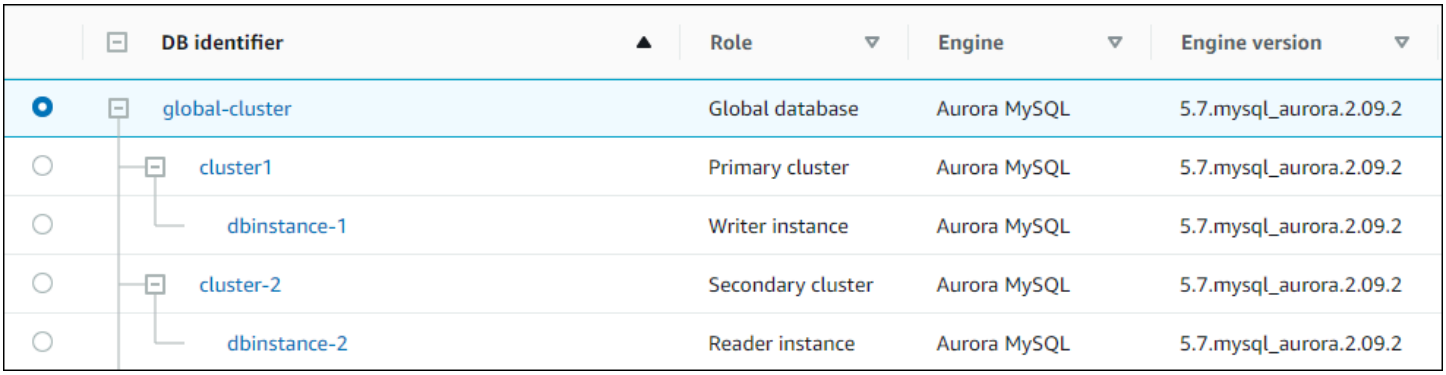
```
# Check the default parameter values for MySQL 8.0-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql8.0 --
region us-east-1
```

Principais atualizações no local para bancos de dados globais

Para um Aurora Global Database, você atualiza o cluster de banco de dados global. O Aurora atualiza automaticamente todos os clusters ao mesmo tempo e garante que todos eles executem a mesma versão do mecanismo. Esse requisito ocorre porque todas as alterações nas tabelas do sistema, formatos de arquivo de dados e outros são replicadas automaticamente para todos os clusters secundários.

Siga as instruções em Como funciona a atualização da versão principal no local Aurora MySQL. Quando você especificar o que deve ser atualizado, escolha o cluster de banco de dados global em vez de um dos clusters que ele contém.

Se você usar o Console de gerenciamento da AWS, escolha o item com a função Global database (Banco de dados global).



Se você usar a AWS CLI ou a API do RDS, inicie o processo de atualização chamando o comando modify-global-cluster ou a operação ModifyGlobalCluster. Você usa um desses em vez de `modify-db-cluster` ou `ModifyDBCluster`.

> **Note**
>
> Você não pode especificar um grupo de parâmetros personalizado para o cluster de banco de dados global enquanto estiver realizando uma atualização de versão principal desse banco de dados global do Aurora. Crie seus grupos de parâmetros personalizados em cada região do cluster global. Depois, aplique-os manualmente aos clusters regionais após a atualização.

Para atualizar a versão principal de um cluster de bancos de dados do Aurora MySQL, use o comando modify-global-cluster da AWS CLI com os seguintes parâmetros obrigatórios:

- `--global-cluster-identifier`
- `--engine aurora-mysql`
- `--engine-version`
- `--allow-major-version-upgrade`

O exemplo a seguir atualiza o cluster de banco de dados global para a Aurora MySQL versão 2.10.2.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-global-cluster \
          --global-cluster-identifier global_cluster_identifier \
          --engine aurora-mysql \
          --engine-version 5.7.mysql_aurora.2.10.2 \
          --allow-major-version-upgrade
```

Para Windows:

```
aws rds modify-global-cluster ^
          --global-cluster-identifier global_cluster_identifier ^
          --engine aurora-mysql ^
          --engine-version 5.7.mysql_aurora.2.10.2 ^
          --allow-major-version-upgrade
```

Considerações sobre retrocesso

Se o cluster que você atualizou tiver o recurso de retrocesso ativado, você não poderá retroceder o cluster atualizado para antes da atualização.

Tutorial de atualização no local de Aurora MySQL

Os exemplos do Linux a seguir mostram como você pode executar as etapas gerais do procedimento de atualização no local usando AWS CLI.

Este primeiro exemplo cria um cluster de banco de dados do Aurora que está executando uma versão 2.x do Aurora MySQL. O cluster inclui uma instância de banco de dados de gravador e uma instância de banco de dados de leitor. O comando `wait db-instance-available` pausa até que a instância de banco de dados do gravador esteja disponível. Esse é o ponto em que o cluster está pronto para ser atualizado.

```
aws rds create-db-cluster --db-cluster-identifier mynewdbcluster --engine aurora-mysql
 \
  --db-cluster-version 5.7.mysql_aurora.2.11.2
...
aws rds create-db-instance --db-instance-identifier mynewdbcluster-instance1 \
  --db-cluster-identifier mynewdbcluster --db-instance-class db.t4g.medium --engine
 aurora-mysql
...
```

```
aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

As versões 3.x do Aurora MySQL para as quais é possível fazer upgrade do cluster dependem da versão 2.x em que o cluster está sendo executado no momento e da Região da AWS em que o cluster está localizado. O primeiro comando com `--output text` apenas mostra a versão de destino disponível. O segundo comando mostra a saída JSON completa da resposta. Nessa resposta, você pode ver detalhes, como o valor `aurora-mysql` que você usa para o parâmetro `engine`. Você também pode ver o fato de que fazer upgrade para a versão 3.04.0 representa um upgrade de versão principal.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.11.2 \
  --query '*[].[ValidUpgradeTarget]'
...
{
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.0",
    "Description": "Aurora MySQL 3.04.0 (compatible with MySQL 8.0.28)",
    "AutoUpgrade": false,
    "IsMajorVersionUpgrade": true,
    "SupportedEngineModes": [
        "provisioned"
    ],
    "SupportsParallelQuery": true,
    "SupportsGlobalDatabases": true,
    "SupportsBabelfish": false,
    "SupportsIntegrations": false
},
...
```

Este exemplo mostra que, se você inserir um número de versão de destino que não seja um destino de atualização válido para o cluster, o Aurora não realizará a atualização. O Aurora também não realiza uma atualização de versão principal, a menos que você inclua o parâmetro `--allow-major-version-upgrade`. Dessa forma, você não pode executar acidentalmente uma atualização que precise exigir testes e alterações amplas no código do aplicativo.

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
```

```
  --engine-version 5.7.mysql_aurora.2.09.2 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: Cannot find upgrade target from 5.7.mysql_aurora.2.11.2 with requested
 version 5.7.mysql_aurora.2.09.2.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --region us-east-1 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: The AllowMajorVersionUpgrade flag must be present when upgrading to a new
 major version.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --apply-immediately --allow-major-version-
upgrade
{
  "DBClusterIdentifier": "mynewdbcluster",
  "Status": "available",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.11.2"
}
```

Demora alguns momentos para que o status do cluster e das instâncias de banco de dados associadas mudem para `upgrading`. Os números de versão para as instâncias de cluster e banco de dados só mudam quando a atualização for concluída. Novamente, você pode usar o comando `wait db-instance-available` para que a instância de banco de dados do gravador aguarde até que a atualização seja concluída antes de prosseguir.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[Status,EngineVersion]' --output text
upgrading 5.7.mysql_aurora.2.11.2

aws rds describe-db-instances --db-instance-identifier mynewdbcluster-instance1 \
  --query '*[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceStatus:DBInstanceStatus} | [0]'
{
    "DBInstanceIdentifier": "mynewdbcluster-instance1",
    "DBInstanceStatus": "upgrading"
}

aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

Neste ponto, o número da versão para o cluster corresponde ao que foi especificado para a atualização.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[EngineVersion]' --output text

8.0.mysql_aurora.3.04.0
```

O exemplo anterior fez uma atualização imediata especificando o parâmetro `--apply-immediately`. Para que a atualização aconteça em um momento conveniente quando o cluster não estiver ocupado, você pode especificar o parâmetro `--no-apply-immediately`. Isso faz com que a atualização inicie durante a próxima janela de manutenção para o cluster. A janela de manutenção define o período durante o qual as operações de manutenção podem ser iniciadas. Uma operação de longa duração pode não ser concluída durante a janela de manutenção. Assim, você não precisa definir uma janela de manutenção maior mesmo se esperar que a atualização possa levar muito tempo.

O exemplo a seguir faz upgrade de um cluster que está executando inicialmente o Aurora MySQL versão 2.11.2. Na saída `describe-db-engine-versions`, os valores `False` e `True` representam a propriedade `IsMajorVersionUpgrade`. A partir da versão 2.11.2, é possível fazer upgrade para algumas outras versões 2.*. Essas atualizações não são consideradas atualizações de versão principais e, portanto, não exigem uma atualização no local. O upgrade no local só está disponível para upgrades para as versões 3.* que são mostradas na lista.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.10.2 \
  --query '*[].[ValidUpgradeTarget|[0][0]|[*].[EngineVersion,IsMajorVersionUpgrade]'
 --output text

5.7.mysql_aurora.2.11.3 False
5.7.mysql_aurora.2.11.4 False
5.7.mysql_aurora.2.11.5 False
5.7.mysql_aurora.2.11.6 False
5.7.mysql_aurora.2.12.0 False
5.7.mysql_aurora.2.12.1 False
5.7.mysql_aurora.2.12.2 False
5.7.mysql_aurora.2.12.3 False
```

```
8.0.mysql_aurora.3.04.0 True
8.0.mysql_aurora.3.04.1 True
8.0.mysql_aurora.3.04.2 True
8.0.mysql_aurora.3.04.3 True
8.0.mysql_aurora.3.05.2 True
8.0.mysql_aurora.3.06.0 True
8.0.mysql_aurora.3.06.1 True
8.0.mysql_aurora.3.07.1 True

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --no-apply-immediately --allow-major-
version-upgrade
...
```

Quando um cluster é criado sem uma janela de manutenção especificada, Aurora seleciona um dia aleatório da semana. Nesse caso, o comando `modify-db-cluster` é enviado em uma segunda-feira. Assim, mudamos a janela de manutenção para terça-feira de manhã. Todos os horários são representados no fuso horário UTC. A janela `tue:10:00-tue:10:30` corresponde ao intervalo das 2h às 2h30 no horário do Pacífico. A alteração na janela de manutenção entra em vigor imediatamente.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "sat:08:20-sat:08:50"
    ]
]

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster --preferred-
maintenance-window tue:10:00-tue:10:30"
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "tue:10:00-tue:10:30"
    ]
]
```

O exemplo a seguir mostra como obter um relatório dos eventos gerados pela atualização. O argumento `--duration` representa o número de minutos para recuperar as informações do evento. Ele é necessário porque, por padrão, `describe-events` só exibe eventos da última hora.

```
aws rds describe-events --source-type db-cluster --source-identifier mynewdbcluster --
duration 20160
{
    "Events": [
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "DB cluster created",
            "EventCategories": [
                "creation"
            ],
            "Date": "2022-11-17T01:24:11.093000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing online pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:08.450000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing offline pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:59.519000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-
mynewdbcluster-5-7-mysql-aurora-2-10-2-to-8-0-mysql-aurora-3-02-0-2022-11-18-22-55].",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:00:22.318000+00:00",
```

```
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Cloning volume.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:01:45.428000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:02:25.141000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:23.036000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Upgrading database objects.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:48.208000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
```

```
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster major version has been upgraded",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:10:28.999000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        }
    ]
}
```

Descobrir os motivos das falhas de atualização da versão principal do Aurora MySQL

No tutorial, a atualização do Aurora MySQL versão 2 para a versão 3 foi bem-sucedida. Mas, se a atualização tivesse falhado, seria útil saber o motivo.

É possível começar usando o comando `describe-events` da AWS CLI para examinar os eventos do cluster de banco de dados. Este exemplo mostra os eventos de `mydbcluster` nas últimas dez horas.

```
aws rds describe-events \
    --source-type db-cluster \
    --source-identifier mydbcluster \
    --duration 600
```

Nesse caso, tivemos uma falha na pré-verificação da atualização.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:22.846000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        },
        {
```

```
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster is in a state that cannot be upgraded: Upgrade
 prechecks failed. For more details, see the
             upgrade-prechecks.log file. For more information on troubleshooting the
 cause of the upgrade failure, see
             https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
AuroraMySQL.Upgrading.Troubleshooting.html",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:24.373000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        }
    ]
}
```

Para diagnosticar a causa exata do problema, examine os logs do banco de dados para a instância de banco de dados de gravador. Quando uma atualização para o Aurora MySQL versão 3 falha, a instância de gravador contém um arquivo de log com o nome do arquivo `upgrade-prechecks.log`. Este exemplo mostra como detectar a presença desse log e, em seguida, baixá-lo para um arquivo local para exame.

```
aws rds describe-db-log-files --db-instance-identifier mydbcluster-instance \
    --query '*[].[LogFileName]' --output text

error/mysql-error-running.log
error/mysql-error-running.log.2024-04-11.20
error/mysql-error-running.log.2024-04-11.21
error/mysql-error.log
external/mysql-external.log
upgrade-prechecks.log

aws rds download-db-log-file-portion --db-instance-identifier mydbcluster-instance \
    --log-file-name upgrade-prechecks.log \
    --starting-token 0 \
    --output text >upgrade_prechecks.log
```

O arquivo `upgrade-prechecks.log` está no formato JSON. Nós o baixamos usando a opção `--output text` para evitar codificar a saída JSON em outro wrapper JSON. Para atualizações do Aurora MySQL versão 3, esse log sempre inclui determinadas mensagens informativas e de

aviso. Ele só inclui mensagens de erro se a atualização falhar. Se a atualização for bem-sucedida, o arquivo de log não será produzido.

Para resumir todos esses erros e exibir os campos de objeto e descrição associados, é possível executar o comando `grep -A 2 '"level": "Error"'` no conteúdo do arquivo `upgrade-prechecks.log`. Isso exibe cada linha de erro e as duas linhas depois dela. Elas contêm o nome do objeto de banco de dados correspondente e orientações sobre como corrigir o problema.

```
$ cat upgrade-prechecks.log | grep -A 2 '"level": "Error"'

"level": "Error",
"dbObject": "problematic_upgrade.dangling_fulltext_index",
"description": "Table `problematic_upgrade.dangling_fulltext_index` contains dangling
 FULLTEXT index. Kindly recreate the table before upgrade."
```

Neste exemplo, é possível executar o comando SQL a seguir na tabela com problemas para tentar corrigi-los ou recriar a tabela sem o índice pendente.

```
OPTIMIZE TABLE problematic_upgrade.dangling_fulltext_index;
```

Tente realizar a atualização novamente.

Solução de problemas para atualização no local de Aurora MySQL

Use as dicas a seguir para ajudar a solucionar problemas com as atualizações no local do Aurora MySQL. Estas dicas não se aplicam a clusters de banco de dados do Aurora Serverless.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção
A réplica associada entre regiões do Aurora ainda não foi corrigida	Aurora cancela a atualização.	Atualize a réplica entre regiões do Aurora e tente novamente.
Cluster tem transações XA no estado preparado	Aurora cancela a atualização.	Confirmar ou reverter todas as transações XA preparadas.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção
O cluster está processando qualquer instrução DDL (Data Definition Language)	Aurora cancela a atualização.	Considere esperar e executar a atualização depois que todas as instruções DDL estiverem concluídas.
Cluster tem alterações não confirmadas para muitas linhas	A atualizaç ão pode levar muito tempo.	O processo de atualização reverte as alteraçõe s não confirmadas. O indicador para esta condição é o valor de `TRX_ROWS_MODIFIED` na tabela `INFORMATION_SCHEMA .INNODB_TRX` . Considere executar a atualização somente depois que todas as grandes transações forem confirmadas ou revertidas.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção
Cluster tem um número elevado de registros para desfazer	A atualizaç ão pode levar muito tempo.	Mesmo que as transações não confirmad as não afetem um grande número de linhas, podem envolver um grande volume de dados. Por exemplo, você pode inserir BLOBs grandes. O Aurora não detecta nem gera automaticamente um evento para esse tipo de atividade de transação. O indicador para essa condição é o tamanho da lista de histórico (HLL). O processo de atualização reverte as alterações não confirmadas. É possível conferir o HLL na saída do comando `SHOW ENGINE INNODB STATUS` SQL ou diretamente usando a seguinte consulta SQL: `SELECT count FROM information_schema .innodb_metrics WHERE name = 'trx_rseg_history_len';` Também é possível monitorar a métrica `RollbackSegmentHistoryListLength` no Amazon CloudWatch. Pense em realizar a atualização somente quando o HLL for menor.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção
O cluster está no processo de confirmação de uma grande transação de log binário	A atualizaç ão pode levar muito tempo.	O processo de atualização aguarda até que as alterações de log binário sejam aplicadas. Mais transações ou instruções DDL podem começar durante esse período, retardando ainda mais o processo de atualização. Agende o processo de atualização quando o cluster não estiver ocupado gerando alteraçõe s de replicação de log binário. O Aurora não detecta nem gera automaticamente um evento para esta condição.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção
Inconsistências em esquemas resultantes da remoção ou corrupção de arquivos	Aurora cancela a atualização.	Altere o mecanismo de armazenamento padrão para tabelas temporárias do MyISAM para o InnoDB. Siga estas etapas: 1. Modifique o parâmetro de banco de dados `default_tmp_storage_engine` para `InnoDB`. 2. Reinicialize o cluster de banco de dados. 3. Após a reinicialização, confirme se o parâmetro de banco de dados `default_tmp_storage_engine` está definido como `InnoDB`. Use o seguinte comando: `show global variables like 'default_tmp_storage_engine';` 4. Certifique-se de não criar nenhuma tabela temporária que use o mecanismo de armazenamento MyISAM. Recomendamos que você pause todas as workloads de banco de dados e não crie nenhuma conexão de banco de dados, porque você fará upgrade em breve. 5. Tente realizar o upgrade local novamente.

Motivo para a atualização no local ser cancelada ou lenta	Efeito	Solução para permitir que a atualização no local seja concluída dentro da janela de manutenção

O usuário mestre foi excluído.	Aurora cancela a atualização.	**Important**
		Não exclua o usuário mestre.
		
		No entanto, se por algum motivo você excluir o usuário mestre, restaure-o usando os seguintes comandos SQL:
		
		`CREATE USER 'master_username '@'% IDENTIFIED BY ' master_user_password ' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;`
		
		`GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, LOAD FROM S3, SELECT INTO S3, INVOKE LAMBDA, INVOKE SAGEMAKER, INVOKE COMPREHEND ON *.* TO 'master_username '@'% WITH GRANT OPTION;`

Para ver mais detalhes sobre a solução de problemas que fazem com que as verificações prévias de atualização falhem, consulte os seguintes blogs:

- Amazon Aurora MySQL version 2 (with MySQL 5.7 compatibility) to version 3 (with MySQL 8.0 compatibility) upgrade checklist, Part 1

- Amazon Aurora MySQL version 2 (with MySQL 5.7 compatibility) to version 3 (with MySQL 8.0 compatibility) upgrade checklist, Part 2

Você pode usar as etapas a seguir para executar suas próprias verificações para algumas das condições na tabela anterior. Dessa forma, você pode agendar a atualização em um momento em que você sabe que o banco de dados está em um estado onde a atualização pode ser concluída com sucesso e rapidamente.

- Você pode verificar se há transações XA abertas executando a instrução `XA RECOVER`. Você pode então confirmar ou reverter as transações XA antes de iniciar a atualização.
- Você pode verificar se há instruções DDL executando uma instrução `SHOW PROCESSLIST` e procurando `CREATE`, `DROP`, `ALTER`, `RENAME` e `TRUNCATE` na saída. Permita que todas as instruções DDL terminem antes de iniciar a atualização.
- Você pode verificar o número total de linhas não confirmadas consultando a tabela `INFORMATION_SCHEMA.INNODB_TRX`. A tabela contém uma linha para cada transação. A coluna `TRX_ROWS_MODIFIED` contém o número de linhas modificadas ou inseridas pela transação.
- Você pode verificar o comprimento da lista de histórico do InnoDB executando a instrução `SHOW ENGINE INNODB STATUS SQL` e procurando o `History list length` na saída. Você também pode verificar o valor diretamente executando a seguinte consulta:

  ```
  SELECT count FROM information_schema.innodb_metrics WHERE name =
   'trx_rseg_history_len';
  ```

  O comprimento da lista de histórico corresponde à quantidade de informações desfeitas armazenadas pelo banco de dados para implementar o controle de simultaneidade de várias versões (MVCC).

Limpeza pós-upgrade do Aurora MySQL versão 3

Depois de concluir o upgrade de qualquer cluster do Aurora MySQL versão 2 para o Aurora MySQL versão 3, você poderá executar essas outras ações de limpeza:

- Crie novas versões compatíveis com o MySQL 8.0 de qualquer grupo de parâmetros personalizado. Aplique todos os valores de parâmetros personalizados necessários aos novos grupos de parâmetros.

- Atualize todos os alarmes do CloudWatch, scripts de configuração e assim por diante para utilizar os novos nomes de qualquer métrica cujos nomes tenham sido afetados por alterações inclusivas de idioma. Para obter uma lista dessas métricas, consulte Alterações de linguagem inclusiva do Aurora MySQL versão 3.
- Atualize qualquer template do CloudFormation para utilizar os novos nomes para quaisquer parâmetros de configuração cujos nomes tenham sido afetados por alterações inclusivas de linguagem. Para obter uma lista completa desses parâmetros, consulte Alterações de linguagem inclusiva do Aurora MySQL versão 3.

Índices espaciais

Depois de atualizar para o Aurora MySQL versão 3, verifique se você precisa descartar ou recriar objetos e índices relacionados a índices espaciais. Antes do MySQL 8.0, o Aurora podia otimizar consultas espaciais utilizando índices que não continham um identificador de recursos espaciais (SRID). O Aurora MySQL versão 3 utiliza apenas índices espaciais contendo SRIDs. Durante um upgrade, o Aurora descarta automaticamente todos os índices espaciais sem SRIDs e imprime mensagens de aviso no log do banco de dados. Se você observar essas mensagens de aviso, crie novos índices espaciais com SRIDs após o upgrade. Para obter mais informações sobre alterações em funções espaciais e tipos de dados no MySQL 8.0, consulte o tópico sobre Alterações feitas no MySQL 8.0, no Guia de referência do MySQL.

Atualizações e correções feitas no mecanismo de banco de dados do Amazon Aurora MySQL

É possível encontrar as seguintes informações em Release notes for Amazon Aurora MySQL-Compatible Edition:

- Atualizações feitas no mecanismo de banco de dados do Amazon Aurora MySQL versão 3
- Atualizações feitas no mecanismo de banco de dados do Amazon Aurora MySQL versão 2
- Atualizações feitas no mecanismo de banco de dados do Amazon Aurora MySQL versão 1 (obsoleta)
- Bugs do MySQL corrigidos por atualizações do mecanismo de banco de dados do Aurora MySQL
- Vulnerabilidades de segurança corrigidas no Amazon Aurora MySQL

Trabalho com Amazon Aurora PostgreSQL

O Amazon Aurora PostgreSQL é um mecanismo de banco de dados relacional totalmente gerenciado e compatível com PostgreSQL e ACID, que combina a velocidade, a confiabilidade e a capacidade de gerenciamento do Amazon Aurora com a simplicidade e a eficiência econômica dos bancos de dados de código aberto. O Aurora PostgreSQL é uma substituição de entrada do PostgreSQL que torna mais simples e econômico configurar, operar e escalar suas implantações do PostgreSQL novas e existentes, permitindo que você se concentre em seus negócios e aplicações. Para saber mais sobre o Aurora em geral, consulte O que é o Amazon Aurora?.

Além dos benefícios do Aurora, o Aurora PostgreSQL oferece um caminho de migração conveniente do Amazon RDS para o Aurora, com ferramentas de migração tão simples quanto botões que convertem suas aplicações do RDS para PostgreSQL em Aurora PostgreSQL. As tarefas de rotina de bancos de dados, como provisionamento, aplicação de patches, backup, recuperação, detecção de falhas e reparo, também são fáceis de gerenciar com o Aurora PostgreSQL.

O Aurora PostgreSQL pode funcionar com vários padrões do setor. Por exemplo, você pode usar os bancos de dados Aurora PostgreSQL para criar aplicações compatíveis com a HIPAA e para armazenar informações relacionadas à saúde, incluindo informações de saúde protegidas, sob um Acordo de Associação Comercial (BAA) celebrado com a AWS.

O Aurora PostgreSQL é ALTAMENTE elegível para o FedRAMP Para obter detalhes sobre a AWS e os esforços de conformidade, consulte os serviços AWS no escopo pelo programa de conformidade.

Important

Se você encontrar algum problema com o cluster de banco de dados do Aurora PostgreSQL, o atendente do AWS Support poderá precisar de mais informações sobre a integridade dos bancos de dados. O objetivo é garantir que o AWS Support receba o máximo possível das informações necessárias no menor tempo possível.
É possível usar o PG Collector para ajudar a reunir informações valiosas do banco de dados em um arquivo HTML consolidado. Para ter mais informações sobre o PG Collector, como executá-lo e como baixar o relatório HTML, consulte PG Collector.
Após a conclusão bem-sucedida e salvo indicação em contrário, o script exibe a saída em um formato HTML legível. O script foi projetado para excluir desse HTML quaisquer dados ou detalhes de segurança que possam comprometer sua empresa. Ele também não faz

modificações no banco de dados ou no ambiente. No entanto, se houver informações no HTML que você não se sinta confortável para compartilhar, fique à vontade para removê-las antes de fazer upload do HTML. Quando o HTML estiver aceitável, faça upload dele utilizando a seção de anexos nos detalhes do caso de suporte.

Tópicos

- Trabalhar com o ambiente de visualização de banco de dados
- Segurança com o Amazon Aurora PostgreSQL
- Atualizar aplicações para conexão com clusters de banco de dados PostgreSQL do Aurora usando novos certificados SSL/TLS
- Usar a autenticação Kerberos com o Aurora PostgreSQL
- Migrar dados para o Amazon Aurora com compatibilidade com o PostgreSQL
- Otimizar a performance de consultas no Aurora PostgreSQL
- Trabalhar com tabelas não registradas em log no Aurora PostgreSQL
- Trabalhar com o autovacuum do PostgreSQL no Amazon Aurora PostgreSQL
- Gerenciar a contenção de TOAST OID no Amazon Aurora PostgreSQL
- Usar o Babelfish para Aurora PostgreSQL
- Performance e escalabilidade do Amazon Aurora PostgreSQL
- Ajustar com eventos de espera do Aurora PostgreSQL
- Ajustar o Aurora PostgreSQL com insights proativos do Amazon DevOps Guru
- Práticas recomendadas do Amazon Aurora PostgreSQL
- Replicação com Amazon Aurora PostgreSQL
- Encaminhamento de gravação local no Aurora PostgreSQL
- Usar o Aurora PostgreSQL como base de conhecimento para o Amazon Bedrock
- Integração do Amazon Aurora PostgreSQL com outros produtos da AWS
- Monitorar planos de execução de consultas e pico de memória do Aurora PostgreSQL
- Gerenciar planos de execução de consultas do Aurora PostgreSQL
- Trabalhar com extensões e invólucros de dados externos
- Trabalhar com Trusted Language Extensions para PostgreSQL

- Referência do Amazon Aurora PostgreSQL
- Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL

Trabalhar com o ambiente de visualização de banco de dados

A comunidade do PostgreSQL lança uma nova versão principal do PostgreSQL anualmente. Da mesma forma, o Amazon Aurora disponibiliza versões principais do PostgreSQL como versões de pré-visualização. Isso permite que você crie um cluster de banco de dados usando a versão de pré-visualização e teste os recursos no Ambiente de Pré-visualização do Banco de Dados.

Os clusters de banco de dados do Aurora PostgreSQL no Ambiente de Pré-visualização do Banco de Dados são semelhantes a outros clusters de banco de dados do Aurora PostgreSQL em termos de funcionalidade. No entanto, você não pode usar uma versão prévia para produção.

Lembre-se das seguintes limitações importantes:

- Todas as instâncias e os clusters de banco de dados são excluídos sessenta dias após a criação, além dos backups e dos snapshots.
- Só é possível criar uma instância de banco de dados em uma virtual private cloud (VPC) com base no serviço da Amazon VPC.
- Não é possível copiar um snapshot de uma instância de banco de dados para um ambiente de produção.

As opções a seguir são compatíveis com a visualização.

- É possível criar instâncias de banco de dados somente com o uso dos tipos r5, r6g, r6i, r7g, r7i, r8g, x2g, t3 e t4g. Para ter mais informações sobre as classes da instância, consulte Classes de instâncias de banco de dados do Amazon Aurora.
- Você pode usar implantações single-AZ e multi-AZ.
- Você pode usar funções padrão de despejo e carregamento do PostgreSQL para exportar ou importar bancos de dados para o Database Preview Environment.

Tipos de classe de instância de banco de dados compatíveis

O Amazon Aurora PostgreSQL é compatível com as seguintes classes de instância de banco de dados na região de pré-visualização:

Classes otimizadas para memória

- db.r5: classes de instância otimizada para memória
- db.r6g: classes de instância otimizada para memória com processadores Graviton2 da AWS.
- db.r6i: classes de instância otimizada para memória
- db.r7g: classes de instância otimizada para memória com processadores Graviton3 da AWS.
- db.r7i: classes de instância otimizada para memória otimizadas para memória
- db.r8g: classes de instância otimizada para memória com processadores Graviton4 da AWS
- db.x2g: classes de instância otimizada para memória desenvolvidas por processadores Graviton2 da AWS.

> **Note**
>
> Para ter mais informações sobre a lista de classes de instância, consulte Tipos de classe de instância de banco de dados.

Classes expansíveis

- db.t3.medium
- db.t3.large
- db.t4g.medium
- db.t4g.large

Recursos não compatíveis no ambiente de pré-visualização

Os recursos a seguir não estão disponíveis no ambiente de visualização:

- Aurora Serverless v1 e v2
- Atualizações de versão principal (MVU)
- Nenhuma nova versão secundária será lançada na região de pré-visualização
- Replicação de entrada do RDS para PostgreSQL para o Aurora PostgreSQL
- Implantação azul/verde do Amazon RDS

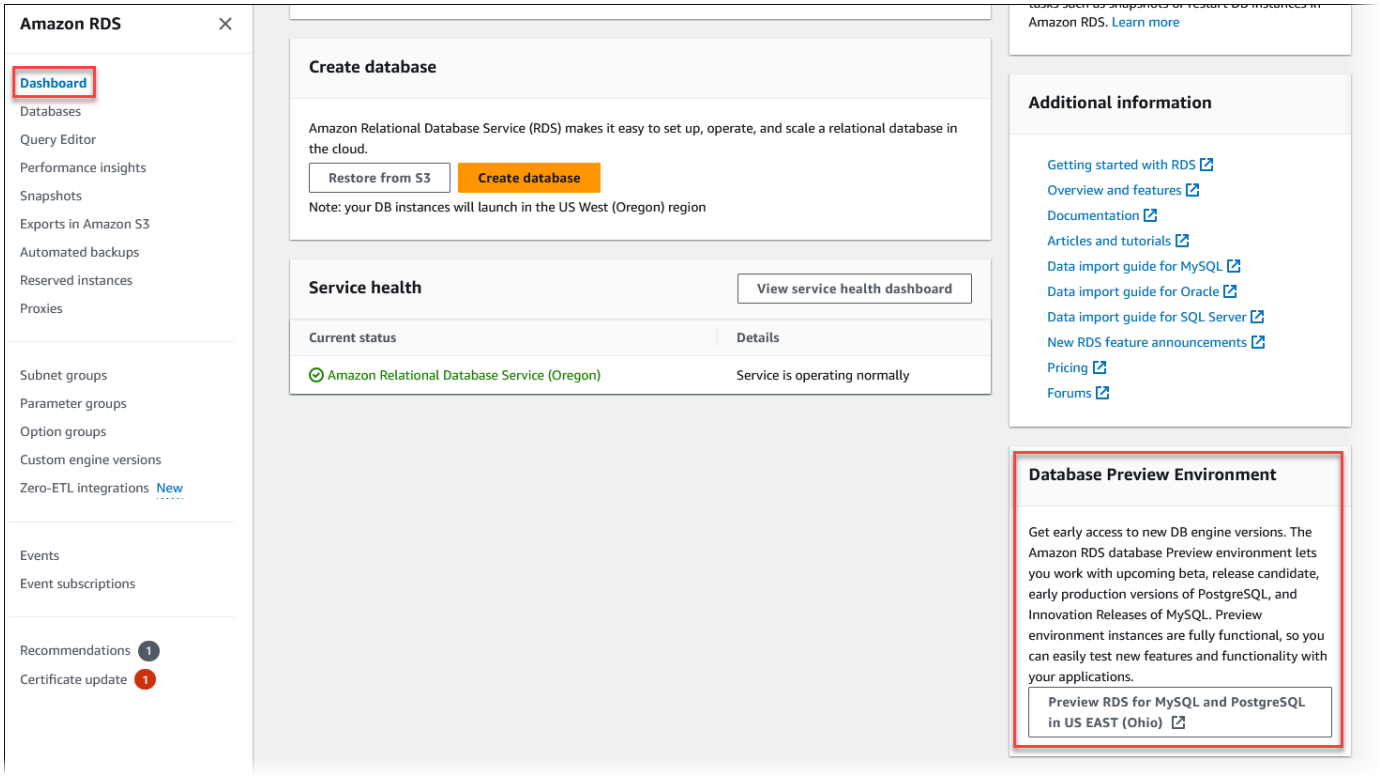
- Cópia de snapshots entre regiões
- Banco de dados global Aurora.
- Fluxos de atividades do banco de dados (DAS), RDS Proxy e AWS DMS
- Réplicas de leitura com ajuste de escala automático
- AWS Bedrock
- Exportação do RDS
- Insights de Performance
- Endpoints personalizados
- Cópia do snapshot
- ETL zero
- Babelfish
- Todas as extensões

Criar um cluster de banco de dados no ambiente de pré-visualização

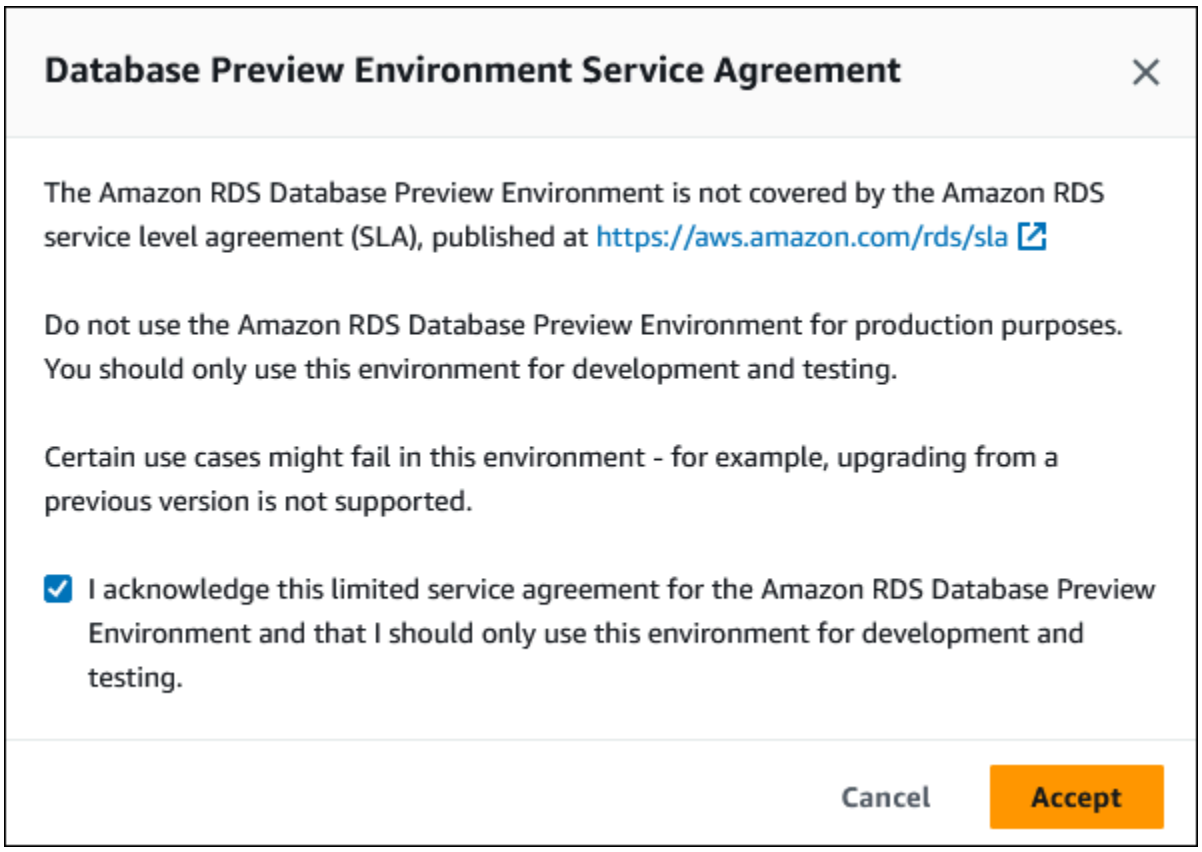
Use o procedimento a seguir para criar um cluster de banco de dados no ambiente de pré-visualização.

Como criar um cluster de banco de dados no ambiente de pré-visualização

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Dashboard (Painel) no painel de navegação.
3. Na página Dashboard (Painel), localize a seção Database Preview Environment (Ambiente de visualização do banco de dados) na página Dashboard (Painel), conforme mostrado na imagem a seguir.



Você também pode navegar diretamente para o Ambiente de visualização do banco de dados. Antes de continuar, você deve reconhecer e aceitar as limitações.



4. Para criar o cluster de banco de dados do Aurora PostgreSQL, siga o mesmo processo de criação de qualquer cluster de banco de dados do Aurora. Para obter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Para criar uma instância no Ambiente de Pré-visualização do Banco de Dados usando a API do Aurora ou a AWS CLI, use o endpoint a seguir.

```
rds-preview.us-east-2.amazonaws.com
```

PostgreSQL versão 17 no Ambiente de Pré-visualização do Banco de Dados

⚠ Esta é a documentação de pré-visualização do Aurora PostgreSQL versão 17. Está sujeita a alteração.

O PostgreSQL versão 17.0 já está disponível no Ambiente de Pré-visualização do Banco de Dados do Amazon RDS. O PostgreSQL versão 17 contém várias melhorias que estão descritas na seguinte documentação do PostgreSQL:

- O PostgreSQL 17 está disponível

Para obter informações sobre o ambiente de visualização de banco de dados, consulte Trabalhar com o ambiente de visualização de banco de dados. Para acessar o ambiente de visualização do console, selecione https://console.aws.amazon.com/rds-preview/.

Segurança com o Amazon Aurora PostgreSQL

Para obter uma visão geral da segurança do Aurora, consulte Segurança no Amazon Aurora. É possível gerenciar a segurança do Amazon Aurora PostgreSQL em alguns níveis diferentes:

- Para controlar quem pode realizar ações de gerenciamento do Amazon RDS em clusters de banco de dados e instâncias de banco de dados Aurora PostgreSQL, use o AWS Identity and Access Management (IAM). O IAM processa a autenticação da identidade do usuário antes que o usuário possa acessar o serviço. Ele também processa a autorização, ou seja, se o usuário tem permissão para fazer o que está tentando fazer. A autenticação de banco de dados do IAM é um método de autenticação adicional que pode ser escolhido ao criar o cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

  Ao usar o IAM com o cluster de banco de dados do Aurora PostgreSQL, primeiro faça login no Console de gerenciamento da AWS com suas credenciais do IAM, antes de abrir o console do Amazon RDS em https://console.aws.amazon.com/rds/.
- Crie clusters de banco de dados do Aurora em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Para controlar quais dispositivos e instâncias do Amazon EC2 podem abrir conexões para o endpoint e a porta da instância de banco de dados dos clusters de banco de dados Aurora em uma VPC, use um grupo de segurança da VPC. Você pode fazer essas conexões de endpoint e de porta usando o Secure Sockets Layer (SSL). Além disso, as regras de firewall em sua empresa podem controlar se dispositivos sendo executados nela podem abrir conexões em uma instância de banco de dados. Para ter mais informações sobre VPCs, consulte Amazon VPC e Amazon Aurora.

A locação da VPC compatível depende da classe de instância de banco de dados usada pelos seus clusters de bancos de dados Aurora PostgreSQL. Com a locação de VPC `default`, o cluster de banco de dados é executado em hardware compartilhado. Com a locação de VPC `dedicated`, o cluster de banco de dados é executado em uma instância de hardware dedicado. As classes de instâncias de banco de dados com performance intermitente apenas oferecem suporte para locação de VPC padrão. As classes de instância de banco de dados de performance intermitente incluem db.t3 e db.t4g. Todas as outras classes de instâncias de banco de dados Aurora PostgreSQL oferecem suporte a locações de VPC padrão e dedicada.

Para ter mais informações sobre as classes da instância, consulte Classes de instâncias de banco de dados do Amazon Aurora. Para ter mais informações sobre a locação de VPC `default` e `dedicated`, consulte Instâncias dedicadas no Guia do usuário do Amazon Elastic Compute Cloud.

- Para conceder permissões aos bancos de dados do PostgreSQL em execução no cluster de banco de dados do Amazon Aurora, é possível usar a mesma abordagem geral usada em instâncias autônomas do PostgreSQL. Comandos, como `CREATE ROLE`, `ALTER ROLE`, `GRANT` e `REVOKE`, funcionam exatamente como em bancos de dados on-premises, assim como a modificação direta de bancos de dados, esquemas e tabelas.

  O PostgreSQL gerencia privilégios usando perfis. O perfil `rds_superuser` é o mais privilegiado em um cluster de banco de dados do Aurora PostgreSQL. Esse perfil é criado automaticamente e concedido ao usuário que cria o cluster de banco de dados (a conta de usuário mestre, `postgres` por padrão). Para saber mais, consulte Noções básicas de perfis e permissões do PostgreSQL.

Todas as versões disponíveis do Aurora PostgreSQL, incluindo as versões 10, 11, 12, 13, 14 e posteriores, são compatíveis com o SCRAM (Salted Challenge Response Authentication Mechanism) para senhas como uma alternativa ao resumo de mensagens (MD5). Recomendamos que você use o SCRAM porque ele é mais seguro que o MD5. Para ter mais informações, inclusive como migrar senhas de usuários do banco de dados do MD5 para o SCRAM, consulte Usar criptografia de senha SCRAM para PostgreSQL.

Noções básicas de perfis e permissões do PostgreSQL

Ao criar um cluster de banco de dados do Aurora PostgreSQL usando o Console de gerenciamento da AWS, uma conta de administrador é criada ao mesmo tempo. Por padrão, o nome é `postgres`, conforme mostrado na captura de tela a seguir:

É possível escolher outro nome em vez de aceitar esse padrão (`postgres`). Se você fizer isso, o nome escolhido deverá começar com uma letra e ter 1 a 16 caracteres alfanuméricos. Para simplificar, nós nos referimos a essa conta de usuário principal pelo seu valor padrão (`postgres`) ao longo deste guia.

Ao usar `create-db-cluster` na AWS CLI, em vez de usar o Console de gerenciamento da AWS, você cria o nome do usuário ao passá-lo com o parâmetro `master-username`. Para obter mais informações, consulte Etapa 2: Criar um cluster de banco de dados do Aurora PostgreSQL.

Se você usar o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS e se usar o nome `postgres` padrão ou escolher um nome diferente, essa primeira conta de usuário do banco de dados será membro do grupo `rds_superuser` e terá privilégios de `rds_superuser`.

Tópicos

- Noções básicas sobre o perfil rds_superuser
- Controlar o acesso de usuários ao banco de dados PostgreSQL
- Delegar e controlar o gerenciamento de senhas de usuários
- Usar criptografia de senha SCRAM para PostgreSQL

Noções básicas sobre o perfil rds_superuser

No PostgreSQL, um perfil pode definir um usuário, um grupo ou um conjunto de permissões específicas concedidas a um grupo ou usuário a vários objetos no banco de dados. Os comandos do PostgreSQL para `CREATE USER` e `CREATE GROUP` foram substituídos pelo comando mais geral `CREATE ROLE` com propriedades específicas para distinguir usuários de banco de dados. Um usuário de banco de dados pode ser considerado um perfil com o privilégio LOGIN.

> **Note**
>
> Os comandos `CREATE USER` e `CREATE GROUP` ainda podem ser usados. Para obter mais informações, consulte Database Roles (Perfis de banco de dados) na documentação do PostgreSQL.

O usuário `postgres` é o usuário de banco de dados mais privilegiado no cluster de banco de dados do Aurora PostgreSQL. Ele tem as características definidas pela seguinte instrução `CREATE ROLE`.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

As propriedades `NOSUPERUSER`, `NOREPLICATION`, `INHERIT` e `VALID UNTIL 'infinity'` são as opções padrão da instrução CREATE ROLE, a menos que especificado de outra forma.

Por padrão, `postgres` tem privilégios concedidos à função `rds_superuser` e permissões para criar funções e bancos de dados. O perfil `rds_superuser` permite que o usuário `postgres` faça o seguinte:

- Adicione as extensões que estão disponíveis para uso com o Aurora PostgreSQL. Para obter mais informações, consulte Trabalhar com extensões e invólucros de dados externos.
- Crie funções para usuários e conceda privilégios aos usuários. Para obter mais informações, consulte CREATE ROLE e GRANT na documentação do PostgreSQL.
- Crie bancos de dados. Para obter mais informações, consulte CREATE DATABASE na documentação do PostgreSQL.
- Conceda privilégios de `rds_superuser` a outras funções de usuário que não têm esses privilégios e revogue esses privilégios conforme necessário. Recomendamos que você conceda esse perfil somente aos usuários que executam tarefas de superusuário. Em outras palavras, você pode conceder esse perfil a administradores de banco de dados (DBAs) ou administradores de sistema.
- Conceda (e revogue) o perfil `rds_replication` a usuários de banco de dados que não têm o perfil `rds_superuser`.
- Conceda (e revogue) o perfil `rds_password` a usuários de banco de dados que não têm o perfil `rds_superuser`.

- Obtenha informações de status sobre todas as conexões de banco de dados usando a visualização `pg_stat_activity`. Quando necessário, `rds_superuser` pode interromper qualquer conexão usando `pg_terminate_backend` ou `pg_cancel_backend`.

Na instrução `CREATE ROLE postgres...`, é possível ver que o perfil do usuário `postgres` não autoriza especificamente as permissões de `superuser` do PostgreSQL. O Aurora PostgreSQL é um serviço gerenciado, portanto, você não pode acessar o sistema operacional host nem se conectar usando a conta `superuser` do PostgreSQL. Muitas das tarefas que exigem o acesso de `superuser` em um PostgreSQL autônomo são gerenciadas automaticamente pelo Aurora.

Para obter mais informações sobre como conceder privilégios, consulte GRANT na documentação do PostgreSQL.

O perfil `rds_superuser` é um dos vários perfis predefinidos em um cluster de banco de dados do Aurora PostgreSQL.

> **Note**
>
> No PostgreSQL 13 e em versões anteriores, os perfis predefinidos são conhecidos como perfis padrão.

Na lista a seguir, você encontra alguns dos outros perfis predefinidos que são criados automaticamente para um novo cluster de banco de dados do Aurora PostgreSQL. Os perfis predefinidos e seus privilégios não podem ser alterados. Não é possível descartar, renomear ou modificar os privilégios desses perfis predefinidos. Qualquer tentativa de fazer isso gerará um erro.

- rds_password: um perfil que pode alterar senhas e configurar restrições de senha para usuários de bancos de dados. O perfil `rds_superuser` recebe esse perfil por padrão e pode concedê-lo aos usuários do banco de dados. Para obter mais informações, consulte Controlar o acesso de usuários ao banco de dados PostgreSQL.
  - Para versões do RDS para PostgreSQL anteriores à 14, o perfil `rds_password` pode alterar senhas e configurar restrições de senha para usuários de bancos de dados e usuários com o perfil `rds_superuser`. Para versões do RDS para PostgreSQL 14 e posteriores, o perfil `rds_password` pode alterar senhas e configurar restrições de senha somente para usuários de banco de dados. Somente usuários com o perfil `rds_superuser` podem realizar essas ações em outros usuários com o perfil `rds_superuser`.

- rdsadmin: um perfil criado para lidar com muitas das tarefas de gerenciamento que o administrador com privilégios de `superuser` executaria em um banco de dados PostgreSQL autônomo. Esse perfil é usado internamente pelo Aurora PostgreSQL para várias tarefas de gerenciamento.

Visualizar perfis e os respectivos privilégios

Você pode visualizar perfis predefinidos e os respectivos privilégios na instância de banco de dados do RDS para PostgreSQL usando comandos diferentes, dependendo da versão do PostgreSQL. Para ver todos os perfis predefinidos, você pode se conectar à sua instância de banco de dados do RDS para PostgreSQL e executar os comandos a seguir usando o `psql`.

Para o **`psql`** 15 e versões anteriores

Conecte-se à instância de banco de dados do RDS para PostgreSQL e use o comando `\du` no psql:

```
postgres=> \du
                                                   List of roles
    Role name    |                         Attributes                         |
                  Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                    +|
 {rds_superuser}
                 | Password valid until infinity                              |
 rds_ad          | Cannot login                                               | {}
 rds_iam         | Cannot login                                               | {}
 rds_password    | Cannot login                                               | {}
 rds_replication | Cannot login                                               | {}
 rds_superuser   | Cannot login                                               |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                              |
```

Para o **`psql`** 16 e versões posteriores

```
postgres=> \drg+
                            List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
```

```
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

Para verificar a associação sem dependência de versão, você pode usar a seguinte consulta SQL:

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

Na saída, é possível ver que `rds_superuser` não é um perfil de usuário de banco de dados (não pode fazer login), mas tem os privilégios de muitos outros perfis. Também é possível ver que esse usuário do banco de dados `postgres` é membro do perfil `rds_superuser`. Como mencionado anteriormente, `postgres` é o valor padrão na página Create database (Criar banco de dados) do console do Amazon RDS. Se você escolheu outro nome, esse nome será mostrado na lista de perfis.

> **Note**
>
> O Aurora PostgreSQL versões 15.2 e 14.7 introduziram um comportamento restritivo do perfil de `rds_superuser`. Um usuário do Aurora PostgreSQL precisa receber o privilégio `CONNECT` no banco de dados correspondente para se conectar, mesmo que ele receba o perfil de `rds_superuser`. Antes das versões 14.7 e 15.2 do Aurora PostgreSQL, um usuário podia se conectar a qualquer banco de dados e tabela do sistema se recebesse o perfil de `rds_superuser`. Esse comportamento restritivo alinha-se com a AWS e os compromissos do Amazon Aurora com a melhoria contínua da segurança.
> Atualize a respectiva lógica nas aplicações se o aprimoramento acima tiver algum impacto.

Controlar o acesso de usuários ao banco de dados PostgreSQL

Os novos bancos de dados no PostgreSQL são sempre criados com um conjunto padrão de privilégios no esquema `public` do banco de dados que permite que todos os usuários e perfis do banco de dados criem objetos. Esses privilégios permitem que os usuários do banco de dados se conectem ao banco de dados, por exemplo, e criem tabelas temporárias durante a conexão.

Para controlar melhor o acesso dos usuários às instâncias de bancos de dados que você cria no nó primário do cluster de banco de dados do Aurora PostgreSQL , recomendamos que você revogue esses privilégios de `public` padrão. Depois disso, conceda privilégios específicos aos usuários do banco de dados de forma mais granular, conforme mostrado no procedimento a seguir.

Como configurar perfis e privilégios para uma nova instância de banco de dados

Suponha que você esteja configurando um banco de dados em um cluster de banco de dados do Aurora PostgreSQL recém-criado para uso por vários pesquisadores que precisam de acesso de leitura-gravação ao banco de dados.

1. Use o `psql` (ou o pgAdmin) para se conectar à instância do banco de dados primário no cluster de banco de dados do Aurora PostgreSQL:

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Insira sua senha quando for solicitado. O cliente `psql` conecta-se e exibe o banco de dados de conexão administrativa padrão, `postgres=>`, como o prompt.

2. Para evitar que os usuários do banco de dados criem objetos no esquema `public`, faça o seguinte:

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   REVOKE
   ```

3. Em seguida, crie uma nova instância de banco de dados:

   ```
   postgres=> CREATE DATABASE lab_db;
   CREATE DATABASE
   ```

4. Revogue todos os privilégios do esquema `PUBLIC` nesse novo banco de dados.

   ```
   postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
   REVOKE
   ```

5. Crie um perfil para os usuários do banco de dados.

   ```
   postgres=> CREATE ROLE lab_tech;
   CREATE ROLE
   ```

6. Permita que os usuários do banco de dados que têm esse perfil conectem-se ao banco de dados.

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. Conceda a todos os usuários com o perfil `lab_tech` todos os privilégios nesse banco de dados.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. Crie os usuários do banco de dados, da seguinte forma:

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. Conceda a esses dois usuários os privilégios associados ao perfil lab_tech:

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

Neste ponto, o `lab_user1` e o `lab_user2` podem conectar-se ao banco de dados `lab_db`. Este exemplo não segue as práticas recomendadas para uso corporativo, o que pode incluir a criação de várias instâncias de banco de dados, esquemas diferentes e concessão de permissões limitadas. Para obter informações mais completas e cenários adicionais, consulte Managing PostgreSQL Users and Roles (Gerenciar usuários e perfis do PostgreSQL).

Para obter mais informações sobre privilégios nos bancos de dados PostgreSQL, consulte o comando GRANT na documentação do PostgreSQL.

Delegar e controlar o gerenciamento de senhas de usuários

Como DBA, talvez você queira delegar o gerenciamento de senhas de usuários. Ou talvez você queira impedir que os usuários do banco de dados alterem senhas ou reconfigurem restrições de senha, como o tempo de vida da senha. Para garantir que somente os usuários do banco de dados escolhidos possam alterar as configurações de senha, é possível ativar o recurso de gerenciamento

restrito de senhas. Quando você ativa esse recurso, somente os usuários do banco de dados que receberam o perfil `rds_password` podem gerenciar senhas.

> **Note**
>
> Para usar o gerenciamento restrito de senhas, o cluster de banco de dados do Aurora PostgreSQL deve estar executando o Amazon Aurora PostgreSQL 10.6 ou superior.

Por padrão, esse recurso está `off`, conforme mostrado a seguir:

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

Para ativar esse recurso, use um grupo de parâmetros personalizado e altere a configuração de `rds.restrict_password_commands` para 1. Reinicialize a instância do banco de dados primário do Aurora PostgreSQL para que a configuração entre em vigor.

Com esse recurso ativo, os privilégios de `rds_password` são necessários para os seguintes comandos SQL:

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

A renomeação de um perfil (`ALTER ROLE myrole RENAME TO newname`) também será restrita se a senha usar o algoritmo de hash MD5.

Com esse recurso ativo, qualquer tentativa de executar um desses comandos SQL sem as permissões do perfil `rds_password` gerará o seguinte erro:

```
ERROR: must be a member of rds_password to alter passwords
```

Recomendamos conceder a `rds_password` a apenas alguns perfis usados exclusivamente para o gerenciamento de senhas. Se você conceder privilégios de `rds_password` a usuários de banco de dados que não têm privilégios de `rds_superuser`, também precisará conceder o atributo `CREATEROLE` a eles.

Verifique os requisitos de senha, como expiração e complexidade necessária, no lado do cliente. Se você usar seu próprio utilitário do lado do cliente para alterações relacionadas a senhas, o utilitário precisará ser membro de `rds_password` e ter privilégios de `CREATE ROLE`.

Usar criptografia de senha SCRAM para PostgreSQL

O Salted Challenge Response Authentication Mechanism (SCRAM) é uma alternativa ao algoritmo de resumo de mensagens padrão (MD5) do PostgreSQL para senhas de criptografia. O mecanismo de autenticação SCRAM é considerado mais seguro que o MD5. Para saber mais sobre essas duas abordagens diferentes para proteger senhas, consulte Autorização com senha na documentação do PostgreSQL.

Recomendamos que você use o SCRAM em vez de o MD5 como o esquema de criptografia de senha para seu cluster de banco de dados do Aurora PostgreSQL. O SCRAM é compatível com o Aurora PostgreSQL versão 10 e todas as versões principais e secundárias posteriores. É um mecanismo criptográfico de resposta a desafios que usa o algoritmo scram-sha-256 para autenticação e criptografia de senha.

Convém atualizar as bibliotecas das aplicações cliente para oferecer compatibilidade com o SCRAM. Por exemplo, versões do JDBC anteriores à 42.2.0 não são compatíveis com o SCRAM. Para obter mais informações, consulte Driver JDBC do PostgreSQL na documentação do driver JDBC do PostgreSQL. Para obter uma lista de outros drivers do PostgreSQL e compatibilidade com o SCRAM, consulte Lista de drivers na documentação do PostgreSQL.

A versão 14 do Aurora PostgreSQL e versões posteriores são compatíveis com scram-sha-256 para criptografia de senha por padrão para novos clusters de banco de dados. Para essas versões, o grupo de parâmetros de cluster de banco de dados padrão (`default.aurora-postgresql14`) tem seu valor `password_encryption` definido como scram-sha-256. Não é possível usar o SCRAM com o Aurora Serverless v1.

Configurar o cluster de banco de dados do Aurora PostgreSQL para exigir o SCRAM

Para o Aurora PostgreSQL 14.3 e versões superiores, você pode exigir que o cluster de banco de dados do Aurora PostgreSQL aceite apenas senhas que usem o algoritmo scram-sha-256.

> **Important**
>
> Para proxies RDS existentes com bancos de dados PostgreSQL, se você modificar a autenticação do banco de dados para usar somente `SCRAM`, o proxy ficará indisponível por até 60 segundos. Para evitar o problema, faça o seguinte:
>
> - O banco de dados deve permitir tanto a autenticação `SCRAM` quanto a `MD5`.
> - Para usar somente a autenticação `SCRAM`, crie um proxy, migre o tráfego da aplicação para o novo proxy e exclua o proxy anteriormente associado ao banco de dados.

Antes de fazer alterações em seu sistema, entenda o processo completo da seguinte forma:

- Obtenha informações sobre todos os perfis e criptografia de senha para todos os usuários do banco de dados.
- Confira novamente as configurações dos parâmetros do cluster de banco de dados do Aurora PostgreSQL para verificar os parâmetros que controlam a criptografia de senha.
- Se o cluster de banco de dados do Aurora PostgreSQL usar um grupo de parâmetros padrão, será necessário criar um grupo de parâmetros de cluster de banco de dados personalizado e aplicá-lo ao cluster de banco de dados do Aurora PostgreSQL para poder modificar parâmetros quando necessário. Se seu cluster de banco de dados do Aurora PostgreSQL usa um grupo de parâmetros personalizado, você poderá modificar os parâmetros necessários posteriormente no processo, conforme necessário.
- Altere o parâmetro `password_encryption` para `scram-sha-256`.
- Avise a todos os usuários do banco de dados que eles precisam atualizar as senhas. Faça o mesmo para a conta `postgres`. As novas senhas são criptografadas e armazenadas usando o algoritmo scram-sha-256.
- Verifique se todas as senhas são criptografadas usando o tipo de criptografia.
- Se todas as senhas utilizarem scram-sha-256, você poderá alterar o parâmetro de `rds.accepted_password_auth_method` para `md5+scram`.

> **Warning**
>
> Depois de alterar `rds.accepted_password_auth_method` para scram-sha-256, os usuários (perfis) com senhas criptografadas com `md5` não conseguirão se conectar.

Preparar-se para exigir o SCRAM para seu cluster de banco de dados do Aurora PostgreSQL

Antes de fazer qualquer alteração em seu cluster de banco de dados do Aurora PostgreSQL, confira todas as contas de usuário do banco de dados existentes. Além disso, verifique o tipo de criptografia usada para senhas. Você pode realizar essas tarefas usando a extensão `rds_tools`. Consulte quais versões do PostgreSQL são compatíveis com `rds_tools` em Extension versions for Amazon RDS for PostgreSQL.

Como obter uma lista de usuários (perfis) de banco de dados e métodos de criptografia de senha

1. Use o `psql` para conectar-se à instância primária de seu cluster de banco de dados do Aurora PostgreSQL conforme mostrado a seguir.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Instale a extensão `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. Obtenha uma lista de perfis e criptografias.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

   Você verá uma saída semelhante à seguinte.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

Criar um grupo de parâmetros de cluster de banco de dados personalizado

> **Note**
>
> Se seu cluster de banco de dados do Aurora PostgreSQL já usa um grupo de parâmetros personalizado, não é necessário criar outro.

Para obter uma visão geral dos grupos de parâmetros do Aurora, consulte Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

O tipo de criptografia de senha usada para senhas é definido em um parâmetro, `password_encryption`. A criptografia permitida pelo cluster de banco de dados do Aurora PostgreSQL é definida em outro parâmetro, `rds.accepted_password_auth_method`. Alterar qualquer um desses valores padrão exige que você crie um grupo de parâmetros de cluster de banco de dados personalizado e aplique-o ao seu cluster.

Você também pode usar o Console de gerenciamento da AWS ou a API do RDS para criar um grupo de parâmetros de cluster de banco de dados personalizado. Consulte mais informações em Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Associe o grupo de parâmetros personalizado à sua instância de banco de dados.

Como criar um grupo de parâmetros de cluster de banco de dados personalizado

1. Use o comando `create-db-cluster-parameter-group` da CLI para criar o grupo de parâmetros personalizado para o cluster. O exemplo a seguir usa `aurora-postgresql13` como a origem desse grupo de parâmetros personalizado.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family aurora-postgresql13  --description 'Custom DB cluster parameter group for SCRAM'
   ```

   Para Windows:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-lab-scram-passwords" ^
   ```

```
  --db-parameter-group-family aurora-postgresql13  --description "Custom DB cluster
 parameter group for SCRAM"
```

Depois, associe o grupo de parâmetros personalizado ao seu cluster.

2. Use o comando `modify-db-cluster` da CLI para aplicar esse grupo de parâmetros personalizado ao cluster de banco de dados do Aurora PostgreSQL.

   Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster --db-cluster-identifier 'your-instance-name' \
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

   Para Windows:

```
aws rds modify-db-cluster --db-cluster-identifier "your-instance-name" ^
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

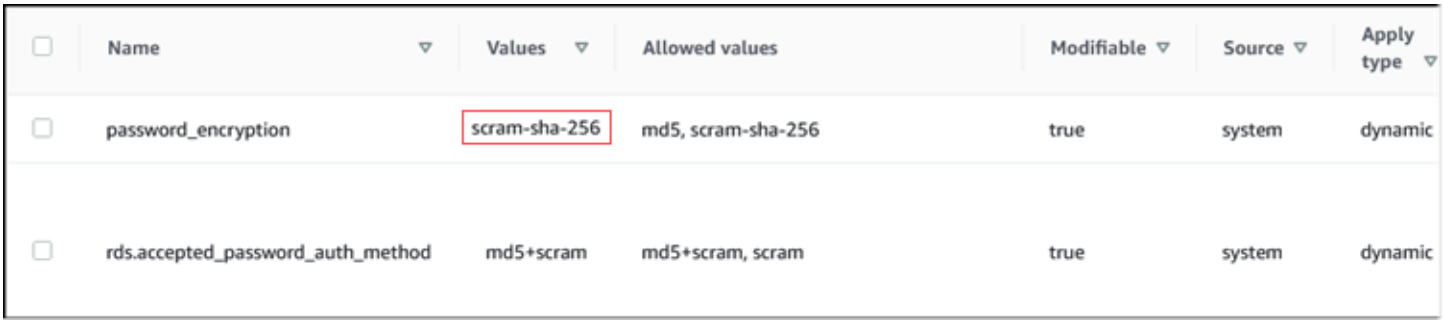
   Para sincronizar novamente seu cluster de banco de dados do Aurora PostgreSQL com seu grupo de parâmetros de cluster de banco de dados personalizado, reinicie a instância primária e todas as outras instâncias do cluster.

Configurar a criptografia de senha para usar o SCRAM

O mecanismo de criptografia de senha usado por um cluster de banco de dados do Aurora PostgreSQL é definido no grupo de parâmetros de cluster de banco de dados no parâmetro `password_encryption`. Os valores permitidos não estão definidos, `md5` ou `scram-sha-256`. O valor padrão depende da versão do Aurora PostgreSQL da seguinte forma:

- Aurora PostgreSQL 14: o padrão é `scram-sha-256`
- Aurora PostgreSQL 13: o padrão é `md5`

Com um grupo de parâmetros de cluster de banco de dados personalizado anexado ao cluster de banco de dados do Aurora PostgreSQL, é possível modificar valores do parâmetro de criptografia de senha.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Como alterar a configuração de criptografia de senha para scram-sha-256

- Altere o valor da criptografia de senha para scram-sha-256, conforme mostrado a seguir. A alteração pode ser aplicada imediatamente porque o parâmetro é dinâmico, portanto, não é necessário reiniciar para que a alteração seja implementada.

  Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name \
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

  Para Windows:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

Migrar senhas para funções de usuário para o SCRAM

Você pode migrar senhas de perfis de usuário para o SCRAM conforme descrito a seguir.

Como migrar senhas de usuário (função) do banco de dados do MD5 para o SCRAM

1. Faça login como usuário administrador (nome de usuário padrão, `postgres`) conforme mostrado a seguir.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Confira a configuração do parâmetro `password_encryption` em sua instância de banco de dados do RDS para PostgreSQL usando o comando a seguir.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. Altere o valor desse parâmetro para scram-sha-256. Para obter mais informações, consulte Configurar a criptografia de senha para usar o SCRAM.
4. Confira o valor novamente para garantir que agora ele esteja definido como `scram-sha-256` da seguinte forma.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

5. Avise a todos os usuários do banco de dados para alterar senhas. Altere também sua própria senha para a conta `postgres` (o usuário do banco de dados com privilégios de `rds_superuser`).

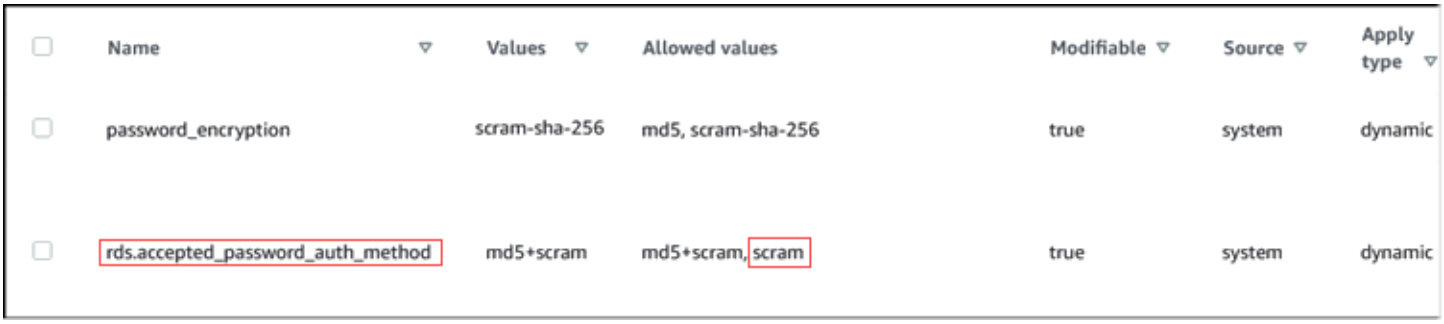
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

6. Repita o processo para todos os bancos de dados no cluster de banco de dados do Aurora PostgreSQL.

Alterar o parâmetro para exigir o SCRAM

Esta é a etapa final do processo. Depois de fazer a alteração no procedimento a seguir, as contas de usuário (perfis) que ainda usam a criptografia `md5` para senhas não poderão fazer login no cluster de banco de dados do Aurora PostgreSQL.

O `rds.accepted_password_auth_method` especifica o método de criptografia que o cluster de banco de dados do Aurora PostgreSQL aceita para uma senha de usuário durante o processo de login. O valor padrão é `md5+scram`, o que significa que qualquer método é aceito. Na imagem a seguir, você pode encontrar a configuração padrão para esse parâmetro.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Os valores permitidos para esse parâmetro são `md5+scram` ou `scram`. Alterar esse valor de parâmetro para `scram` torna isso um requisito.

Como alterar o valor do parâmetro para exigir autenticação SCRAM para senhas

1. Verifique se todas as senhas de usuários para todos os bancos de dados no cluster de banco de dados do Aurora PostgreSQL usam `scram-sha-256` para criptografia de senha. Para fazer isso, consulte `rds_tools` para obter o perfil (usuário) e o tipo de criptografia, da seguinte forma.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+-----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. Repita a consulta para todas as instâncias de banco de dados em seu cluster de banco de dados do Aurora PostgreSQL.

   Se todas as senhas usam scram-sha-256, você pode prosseguir.

3. Altere o valor da autenticação de senha aceita para scram-sha-256 da seguinte forma.

   Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-
lab-scram-passwords' \
```

```
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-
lab-scram-passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Como proteger dados do Aurora PostgreSQL com SSL/TLS

O Amazon RDS oferece suporte à criptografia de Secure Socket Layer (SSL) e Transport Layer Security (TLS) para clusters de bancos de dados Aurora PostgreSQL. Usando o SSL/TLS, você pode criptografar uma conexão entre as suas aplicações e clusters de banco de dados do PostgreSQL do Aurora. Você também pode forçar todas as conexões do seu cluster de bancos de dados Aurora PostgreSQL a usar SSL/TLS. O Amazon Aurora PostgreSQL oferece suporte ao Transport Layer Security (TLS) versões 1.1 e 1.2. Recomendamos usar o TLS 1.2 para conexões criptografadas. Adicionamos suporte ao TLSv1.3 nas seguintes versões do Aurora PostgreSQL:

- 15.3 e todas as versões posteriores
- 14.8 e versões 14 posteriores
- 13.11 e versões 13 posteriores
- 12.15 e versões 12 posteriores
- 11.20 e versões 11 posteriores

Para obter informações gerais sobre o suporte para SSL/TLS e bancos de dados do PostgreSQL, consulte Suporte para SSL na documentação do PostgreSQL. Para obter informações sobre como usar uma conexão SSL/TLS via JDBC, consulte Como configurar o cliente na documentação do PostgreSQL.

Tópicos

- Como exigir uma conexão SSL/TLS de um cluster de bancos de dados Aurora PostgreSQL
- Como determinar o status de conexão SSL/TLS
- Configurar conjuntos de cifras para conexões a clusters de banco de dados PostgreSQL do Aurora

O suporte a SSL/TLS está disponível em todas as regiões da AWS para o Aurora PostgreSQL. O Amazon RDS criará um certificado SSL/TLS para o seu cluster de bancos de dados Aurora PostgreSQL quando o cluster de banco de dados for criado. Se você habilitar a verificação de certificado SSL/TLS, o certificado SSL/TLS incluirá o endpoint do cluster de banco de dados como o nome comum (CN) do certificado SSL/TLS para se proteger contra ataques de falsificação.

Para se conectar a um cluster de bancos de dados Aurora PostgreSQL via SSL/TLS

1. Baixe o certificado.

   Para obter informações sobre como baixar certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.
2. Importe o certificado no sistema operacional.
3. Conecte-se ao seu cluster de bancos de dados Aurora PostgreSQL via SSL/TLS.

   Quando você se conectar usando o SSL/TLS, seu cliente poderá optar por verificar ou não a cadeia de certificados. Se os seus parâmetros de conexão especificarem `sslmode=verify-ca` ou `sslmode=verify-full`, seu cliente exigirá que os certificados de CA do RDS estejam no armazenamento confiável ou sejam referenciados no URL da conexão. Esse requisito tem o objetivo de verificar a cadeia de certificados que assina o seu certificado de banco de dados.

   Quando um cliente, como psql ou JDBC, for configurado com o suporte para SSL/TLS, primeiro ele tentará se conectar ao banco de dados via SSL/TLS por padrão. Se esse cliente não puder se conectar via SSL/TLS, ele retomará a conexão sem SSL/TLS. Por padrão, a opção `sslmode` para clientes baseados em JDBC e libpq é definida como `prefer`.

   Use o parâmetro `sslrootcert` para fazer referência ao certificado, por exemplo `sslrootcert=rds-ssl-ca-cert.pem`.

Veja a seguir um exemplo de como usar psql para se conectar a um cluster de banco de dados PostgreSQL do Aurora.

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Como exigir uma conexão SSL/TLS de um cluster de bancos de dados Aurora PostgreSQL

Para exigir conexões SSL/TLS com o cluster de bancos de dados do Aurora PostgreSQL, use o parâmetro `rds.force_ssl`.

- Para exigir conexões SSL/TLS, defina o valor do parâmetro `rds.force_ssl` como 1 (ativado).
- Para desativar as conexões SSL/TLS obrigatórias, defina o valor do parâmetro `rds.force_ssl` como 0 (desativado).

O valor padrão desse parâmetro depende da versão do Aurora PostgreSQL:

- Para o Aurora PostgreSQL versões 17 e posteriores: o valor padrão é 1 (ativado).
- Para o Aurora PostgreSQL versões 16 e posteriores: o valor padrão é 0 (desativado).

> **Note**
>
> Quando você realiza uma atualização de versão principal do Aurora PostgreSQL versão 16 ou anterior para a versão 17 ou posterior, o valor padrão do parâmetro muda de 0 (desativado) para 1 (ativado). Essa alteração pode causar falhas de conectividade em aplicações que não estão configuradas para SSL. Você pode reverter para o comportamento padrão anterior definindo esse parâmetro como 0 (desativado).

Para ter mais informações como lidar com parâmetros, consulte Grupos de parâmetros para Amazon Aurora.

Atualizar o parâmetro `rds.force_ssl` também define o parâmetro `ssl` do PostgreSQL para 1 (ativado) e modifica o arquivo `pg_hba.conf` do seu cluster de banco de dados para oferecer suporte à nova configuração de SSL/TLS.

Quando o parâmetro `rds.force_ssl` for definido como 1 para um cluster de banco de dados, você verá um resultado semelhante ao seguinte ao se conectar, indicando a exigência do SSL/TLS:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
```

```
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Como determinar o status de conexão SSL/TLS

O status criptografado da conexão é mostrado no banner de login quando você se conecta ao cluster de banco de dados:

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Você também pode carregar a extensão `sslinfo` e, então, chamar a função `ssl_is_used()` para determinar se o SSL/TLS está em uso. A função retornará `t` se a conexão estiver usando o SSL/TLS; caso contrário, retornará `f`.

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

Você pode usar o comando `select ssl_cipher()` para determinar a criptografia de SSL/TLS:

```
postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

Se você habilitar `set rds.force_ssl` e reiniciar seu cluster de banco de dados, as conexões sem SSL serão recusadas com a seguinte mensagem:

```
$ export PGSSLMODE=disable
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database
 "postgres", SSL off
$
```

Para obter informações sobre a opção `sslmode`, consulte Funções de controle de conexão com o banco de dados na documentação do PostgreSQL.

Configurar conjuntos de cifras para conexões a clusters de banco de dados PostgreSQL do Aurora

Usando conjuntos de cifras configuráveis, você pode ter mais controle sobre a segurança de suas conexões de banco de dados. Você pode especificar uma lista de conjuntos de cifras que deseja permitir para proteger conexões SSL/TLS do cliente com seu banco de dados. Com conjuntos de cifras configuráveis, você pode controlar a criptografia de conexão aceita pelo servidor de banco de dados. Fazer isso ajuda a evitar o uso de cifras inseguras ou obsoletas.

Os conjuntos de cifras configuráveis são compatíveis com as versões 11.8 e posteriores do Aurora PostgreSQL.

Para especificar a lista de cifras permitidas para criptografia de conexões, modifique o parâmetro de cluster `ssl_ciphers`. Defina o parâmetro `ssl_ciphers` como uma string de valores de criptografia separados por vírgulas em um grupo de parâmetros do cluster usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Para definir parâmetros de cluster, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

A tabela a seguir mostra as criptografias compatíveis para as versões do mecanismo do Aurora PostgreSQL

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
9.6, 10.20 e anteriores, 11.15 e anterior, 12.10 e anterior, 13.6 e anterior	DHE-RSA-AES128-SHA	Sim	Não	Não
	DHE-RSA-AES128-SHA256	Não	Sim	Não
	DHE-RSA-AES128-GCM-SHA256	Não	Sim	Não
	DHE-RSA-AES256-SHA	Não	Sim	Não
	DHE-RSA-AES256-SHA256	Não	Sim	Não
	DHE-RSA-AES256-GCM-SHA384	Não	Sim	Não
	ECDHE-ECDSA-AES256-SHA	Não	Sim	Não
	ECDHE-ECDSA-AES256-GCM-SHA384	Não	Sim	Não
	ECDHE-RSA-AES256-SHA384	Não	Sim	Não
	ECDHE-RSA-AES128-SHA	Não	Sim	Não
	ECDHE-RSA-AES128-SHA256	Não	Sim	Não
	ECDHE-RSA-AES128-GCM-SHA256	Não	Sim	Não

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
	ECDHE-RSA-AES256-SHA ECDHE-RSA-AES256-GCM-SHA384			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
10.21, 11.16, 12.11, 13.7, 14.3 e 14.4	ECDHE-RSA-AES128-SHATLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sim	Sim	Não
		Não	Sim	Não
		Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sim	Sim	Não
		Não	Sim	Não
		Sim	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Não	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Sim	Sim	Não
		Não	Sim	Não
		Sim	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA			
	TLS_ECDHE_RSA_WITH			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
	_AES_256_ GCM_SHA384 TLS_RSA_W ITH_AES_2 56_GCM_SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_W ITH_AES_1 28_GCM_SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_WITH _CHACHA20 _POLY1305_SHA256			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
10.22, 11.17, 12.12, 13.8, 14.5 e 15.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Não	Sim	Não
		Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sim	Sim	Não
		Não	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Não	Sim	Não
		Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sim	Sim	Não
		Sim	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
11.20, 12.15, 13.11, 14.8, 15.3, 16.1 e posterior	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Não	Sim	Não
		Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sim	Sim	Não
		Não	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Não	Sim	Não
		Sim	Sim	Não
		Não	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sim	Sim	Não
		Sim	Sim	Não
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Não	Sim	Não
		Não	Não	Sim
		Não	Não	Sim
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256			

Versões de mecanismo do Aurora PostgreSQL	Criptografias compatíveis	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_AES_256_GCM_SHA384			

Você também pode usar o comando da CLI describe-engine-default-cluster-parameters para determinar quais conjuntos de cifras são atualmente compatíveis com uma família de grupos de parâmetros específica. O exemplo a seguir mostra como obter os valores permitidos para o parâmetro de cluster `ssl_cipher` para o Aurora PostgreSQL 11.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
postgresql11

    ...some output truncated...
 {
  "ParameterName": "ssl_ciphers",
  "Description": "Sets the list of allowed TLS ciphers to be used on secure
 connections.",
  "Source": "engine-default",
  "ApplyType": "dynamic",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,
  ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-
SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-
SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,

 TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,T

 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AE
  "IsModifiable": true,
  "MinimumEngineVersion": "11.8",
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
    ...some output truncated...
```

O parâmetro `ssl_ciphers` é o padrão para todos os conjuntos de criptografia permitidos. Para ter mais informações sobre cifras, consulte a variável ssl_ciphers na documentação do PostgreSQL.

Usar o mascaramento dinâmico do Aurora PostgreSQL

O mascaramento dinâmico de dados é um recurso de segurança que protege dados sensíveis em bancos de dados do Aurora PostgreSQL, controlando como os dados aparecem para os usuários no momento da consulta. O Aurora o implementa por meio da extensão `pg_columnmask`. `pg_columnmask` oferece proteção de dados em nível de coluna que complementa a segurança por linha nativa e os mecanismos granulares de controle de acesso do PostgreSQL.

Com `pg_columnmask`, você cria políticas de mascaramento que determinam a visibilidade dos dados com base em perfis de usuários. Quando os usuários consultam tabelas com políticas de mascaramento, o Aurora PostgreSQL aplica a função de mascaramento apropriada no momento da consulta, com base no perfil do usuário e no peso da política. Os dados subjacentes permanecem inalterados no armazenamento.

`pg_columnmask` comporta os seguintes recursos:

- Funções de mascaramento integradas e personalizadas: use funções pré-integradas para padrões comuns, como mascaramento de e-mail e texto, ou crie suas próprias funções personalizadas para proteger dados sensíveis (PII) por meio de políticas de mascaramento baseadas em SQL.
- Várias estratégias de mascaramento: oculte completamente as informações, substitua valores parciais por curingas ou defina abordagens de mascaramento personalizadas.
- Priorização de políticas: defina várias políticas para uma única coluna. Use pesos para determinar qual política de mascaramento deve ser usada quando várias políticas se aplicam a uma coluna. O Aurora PostgreSQL aplica políticas com base no peso e na associação do perfil do usuário.

`pg_columnmask` está disponível no Aurora PostgreSQL versão 16.10 e posterior e na versão 17.6 e posterior. As informações estão disponíveis sem custo adicional.

Conceitos básicos de mascaramento dinâmico

Para mascarar dados dinamicamente, você deve instalar a extensão `pg_columnmask` em seu banco de dados e cria políticas de mascaramento para suas tabelas. O processo de configuração envolve verificação de pré-requisitos, instalação de extensões, configuração de perfis, criação de políticas e testes de validação.

Instalação e configuração da extensão

Conecte-se ao seu cluster do Aurora PostgreSQL usando o Editor de consultas do console do RDS ou um cliente do PostgreSQL, como psql, com credenciais rds_superuser (usuário mestre).

Execute o comando de criação da extensão para habilitar a funcionalidade `pg_columnmask`:

```
CREATE EXTENSION pg_columnmask;
```

Esse comando instala a extensão `pg_columnmask`, cria as tabelas de catálogo necessárias e registra as funções de mascaramento integradas. A instalação da extensão é específica do banco de dados, o que significa que você deve instalá-la separadamente em cada banco de dados em que a funcionalidade é necessária.

> **Note**
>
> As conexões estabelecidas antes da instalação dessa extensão ainda mostrarão dados não mascarados. Feche e restabeleça a conexão para corrigir isso.

Verifique a instalação da extensão conferindo as funções de mascaramento disponíveis:

```
SELECT proname FROM pg_proc
    WHERE pronamespace = 'pgcolumnmask'::regnamespace AND proname LIKE 'mask_%';
    proname
--------Output --------
 mask_email
 mask_text
 mask_timestamp
(3 rows)
```

Procedimentos para gerenciar políticas de mascaramento de dados

Você pode gerenciar políticas de mascaramento usando os procedimentos fornecidos pela extensão `pg_columnmask`. Para criar, modificar ou eliminar políticas de mascaramento, é necessário ter um dos seguintes privilégios:

- Proprietário da tabela na qual você está criando a política `pg_columnmask`.
- Membro do `rds_superuser`.
- Membro do perfil de gerente de políticas `pg_columnmask` definido pelo parâmetro `pgcolumnmask.policy_admin_rolname`.

O seguinte comando cria uma tabela que é usada nas seções seguintes:

```
CREATE TABLE public.customers (
    id SERIAL PRIMARY KEY,
    name TEXT,
    phone TEXT,
    address TEXT,
    email TEXT
);
```

CREATE_MASKING_POLICY

O seguinte procedimento cria uma política de mascaramento para uma tabela de usuários:

Sintaxe

```
create_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Argumentos

Parâmetro	DataType	Descrição
`policy_name`	NAME	O nome da política de mascaramento. Deve ser exclusivo por tabela.
`table_name`	REGCLASS	O nome ou oid qualificado/não qualificado da tabela à qual aplicar a política de mascaramento.
`masking_expressions`	JSONB	Objeto JSON que contém o nome da coluna e pares de funções de mascaramento. Cada chave é um nome de coluna e seu valor é a

Parâmetro	DataType	Descrição
		expressão de mascaramento a ser aplicada nessa coluna.
`roles`	NAME[]	Os perfis aos quais essa política de mascaramento se aplica. O padrão é PUBLIC.
`weight`	INT	Peso da política de mascaramento. Quando várias políticas são aplicáveis à consulta de determinado usuário, a política com o maior peso (maior número inteiro) será aplicada a cada coluna mascarada. O padrão é 0. Não há duas políticas de mascaramento na tabela com o mesmo peso.

Tipo de retorno

Nenhum

Example da criação de uma política de mascaramento que mascare a coluna de e-mail do perfil **`test_user`**:

```
CALL pgcolumnmask.create_masking_policy(
    'customer_mask',
    'public.customers',
    JSON_OBJECT('{
        "email", "pgcolumnmask.mask_email(email)"
    }')::JSONB,
    ARRAY['test_user'],
    100
);
```

ALTER_MASKING_POLICY

Esse procedimento modifica uma política de mascaramento existente. `ALTER_MASKING_POLICY` pode modificar as expressões de mascaramento da política, o conjunto de perfis aos quais a política se aplica e o peso da política de mascaramento. Quando um desses parâmetros é omitido, a parte correspondente da política permanece inalterada.

Sintaxe

```
alter_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Argumentos

Parâmetro	DataType	Descrição
`policy_name`	NAME	Nome da política de mascaramento existente.
`table_name`	REGCLASS	O oid do nome qualificado/não qualificado da tabela que contém a política de mascaramento.
`masking_expressions`	JSONB	Novo objeto JSON que contém pares de função de mascaramento e nome da coluna ou, caso contrário, NULL.
`roles`	NAME[]	A lista de novos perfis aos quais essa política de mascaramento se aplica ou, caso contrário, NULL.

Parâmetro	DataType	Descrição
`weight`	INT	Novo peso para a política de mascaramento ou, caso contrário, NULL.

Tipo de retorno

Nenhum

Example de adicionar o perfil de analista a uma política de mascaramento existente sem alterar outros atributos da política.

```
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'public.customers',
    NULL,
    ARRAY['test_user', 'analyst'],
    NULL
);

-- Alter the weight of the policy without altering other details
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'customers',
    NULL,
    NULL,
    4
);
```

DROP_MASKING_POLICY

Esse procedimento remove uma política de mascaramento existente.

Sintaxe

```
drop_masking_policy(
        policy_name,
        table_name)
```

Argumentos

Parâmetro	DataType	Descrição
`policy_name`	NAME	Nome da política de mascaramento existente.
`table_name`	REGCLASS	O oid do nome qualificado/não qualificado da tabela que contém a política de mascaramento.

Tipo de retorno

Nenhum

Example de descartar a política de mascaramento customer_mask

```
-- Drop a masking policy
    CALL pgcolumnmask.drop_masking_policy(
        'customer_mask',
        'public.customers',
    );
```

Inserir caractere de escape em identificadores no procedimento DDL da política de mascaramento

Quando você cria políticas de mascaramento de dados com identificadores entre aspas, o escape adequado é necessário para garantir referências de objetos e aplicação de políticas corretas. Para usar identificadores entre aspas nos procedimentos de gerenciamento da política de mascaramento `pg_columnmask`:

- Nome da política: deve ser entre aspas duplas.
- Nome da tabela: tanto o nome do esquema quanto o nome da tabela devem ser colocados entre aspas duplas individualmente quando necessário.
- Expressões de mascaramento: os nomes de colunas e funções nas expressões de mascaramento devem estar entre aspas duplas e as próprias aspas devem ser excluídas usando uma barra invertida.

- Funções: a matriz de nomes de perfil é automaticamente citada. O nome do perfil deve corresponder exatamente ao nome, conforme visto em `pg_roles` incluindo a distinção entre maiúsculas e minúsculas.

Example da sintaxe de escape e aspas

Este exemplo mostra a sintaxe adequada de escape e aspas na criação de políticas de mascaramento para tabelas, colunas, funções e perfis que usam nomes em maiúsculas e minúsculas ou exigem identificadores entre aspas no Aurora PostgreSQL.

```
-- Create a table and columns with mixed case name
CREATE TABLE public."Employees" (
    "Name" TEXT,
    "Email" TEXT,
    ssn VARCHAR(20)
);

-- Create a role with mixed case name
CREATE ROLE "Masked_user";

-- Create a function with mixed case name
CREATE OR REPLACE FUNCTION public."MaskEmail"(text)
    RETURNS character varying
    LANGUAGE plpgsql
    IMMUTABLE PARALLEL SAFE
    AS $$ BEGIN
        RETURN 'XXXXXXXX'::text;
    END $$;

-- Now use these objects with mixed case names in
-- masking policy management procedures
CALL pgcolumnmask.create_masking_policy(
    '"Policy1"',  -- policy name should be surrounded with double quotes for quoting
    'public."Employees"', -- table and schema name should be individually
                          -- surrounded with double quotes for quoting
    JSON_OBJECT('{
        "\"Email\"", "\"MaskEmail\"(\"Email\")"
    }')::JSONB, -- masking expression should have double quotes around function names
                -- and columns names etc when needed. Also the double quotes itself
                -- should be escaped using \ (backslash) since this is a JSON string
    ARRAY['Masked_user'], -- Rolename do not need quoting
                          -- (this behaviour may change in future release)
```

```
    100
);

SELECT * FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'Employees';
-[ RECORD 1 ]-----+-------------------------------------
schemaname        | public
tablename         | Employees
policyname        | Policy1
roles             | {Masked_user}
masked_columns    | {Email}
masking_functions | {"(\"MaskEmail\"(\"Email\")):text"}
weight            | 100
```

Visualizações administrativas

Você pode revisar toda a política `pg_columnmask` usando a visualização administrativa `pgcolumnmask.pg_columnmask_policies` acessível ao público. As informações a seguir estão disponíveis usando essa visualização. A visualização exibe somente as políticas de mascaramento pertencentes ao usuário atual.

Nome da coluna	Tipo de dados	Descrição
schemaname	NAME	Esquema da relação à qual a política está anexada
tablename	NAME	Nome da relação à qual a política está anexada
policyname	NAME	Nome da política de mascaramento, todas as políticas de mascaramento têm nomes exclusivos
perfis	TEXT[]	Perfil ao qual a política se aplica.
masked_columns	TEXT[]	Colunas mascaradas
masking_functions	TEXT[]	Funções de mascaramento

Nome da coluna	Tipo de dados	Descrição
weight	INT	Peso da política anexada

Funções de mascaramento de dados predefinidas

A extensão `pg_columnmask` fornece funções utilitárias integradas escritas em linguagem C (para execução mais rápida) que podem ser usadas como expressão de mascaramento para políticas `pg_columnmask`.

mask_text

Uma função para mascarar dados de texto com opções de visibilidade configuráveis.

Argumentos

Parâmetro	DataType	Descrição
`input`	TEXT	A string de texto original a ser mascarada.
`mask_char`	CHAR(1)	Caractere usado para mascarar (padrão: “X”).
`visible_prefix`	INT	Número de caracteres no início do texto de entrada que permanecerão sem máscara (padrão: 0).
`visible_suffix`	INT	Número de caracteres no fim do texto de entrada que permanecerão sem máscara (padrão: 0).
`use_hash_mask`	BOOLEAN	Se TRUE, usa um mascaramento baseado em hash em vez de mask_char (padrão: FALSE).

Example de usar diferentes opções de mascaramento

Mascare toda a string de entrada com o caractere “X” padrão.

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World');
  mask_text
-------------
 XXXXXXXXXXX
```

Use o argumento `mask_char` para mascarar a entrada de texto usando um caractere diferente.

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*');
  mask_text
-------------
 ***********
```

Use os parâmetros `visible_prefix` e `visible_suffix` para controlar quantos caracteres permanecem sem máscara no início e no final do texto.

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 5, 1);
  mask_text
-------------
 Hello*****d
```

Quando `use_hash_mask` é true, a string de entrada é mascarada usando caracteres aleatórios, o argumento `mask_char` é ignorado, mas `visible_prefix` ainda `visible_suffix` ainda são respeitados

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 2, 2, true);
  mask_text
-------------
 Hex36dOHild
```

mask_timestamp

Parâmetro	DataType	Descrição
`ts_to_mask`	TIMESTAMP	O carimbo de data/hora original a ser mascarado.

Parâmetro	DataType	Descrição
`mask_part`	TEXT	Especifica qual parte do carimbo de data/hora deve ser mascarada (padrão: “tudo”). Valores válidos: “ano”, “mês”, “dia”, “hora”, “minuto”, “segundo”, “tudo”.
`mask_value`	TIMESTAMP	O valor do carimbo de data/hora a ser usado para mascaramento (padrão: “1900-01-01 00:00:00”).

Example de usar **`mask_timestamps`**

Esses exemplos demonstram o mascaramento completo do carimbo de data/hora em um valor padrão, o mascaramento parcial de componentes específicos do carimbo de data/hora (somente ano) e o mascaramento com um valor de substituição personalizado.

Mascarar completamente o valor de entrada para o carimbo de data/hora padrão

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00');
   mask_timestamp
---------------------
 1900-01-01 00:00:00
```

Como mascarar apenas uma parte do carimbo de data/hora, por exemplo, apenas o ano

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'year');
   mask_timestamp
---------------------
 1900-06-15 14:30:00
```

Para alterar o valor mascarado do carimbo de data/hora, use o argumento `mask_value`.

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'all', '2012-12-12
 12:12:12');
   mask_timestamp
```

```
---------------------
 2012-12-12 12:12:12
```

mask_timestamp

Uma função para mascarar endereços de e-mail, mas preservar a estrutura.

Parâmetro	DataType	Descrição
`input`	TEXT	O endereço de e-mail original a ser mascarado.
`mask_char`	CHAR(1)	Caractere usado para mascarar (padrão: “X”).
`mask_local`	BOOLEAN	Se TRUE, mascara a parte local do e-mail (antes de @) (padrão: TRUE).
`mask_domain`	BOOLEAN	Se TRUE, mascara a parte do domínio do e-mail (após @) (padrão: TRUE).

Example de usar **`mask_email`**

Esses exemplos demonstram o mascaramento de e-mail completo, os caracteres de máscara personalizados e o mascaramento seletivo da parte local ou da parte do domínio do endereço de e-mail.

Mascaramento completo

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com');
    mask_email
------------------
 XXXX@XXXXXXX.com
```

Usar `mask_char` para alterar o caractere usado para mascarar

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*');
```

```
    mask_email
------------------
 ****@*******.com
```

Usar `mask_local` e `mask_domain` para controlar o mascaramento no local e no domínio

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', true, false);
    mask_email
------------------
 ****@example.com

postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', false, true);
    mask_email
------------------
 user@*******.com
```

Implementar pg_columnmask em um fluxo de trabalho completo

Esta seção demonstra uma implementação completa do `pg_columnmask` utilizando um exemplo de tabela de funcionários com dados sensíveis. Você vai aprender a criar funções de mascaramento personalizadas, definir várias políticas de mascaramento com diferentes níveis de peso para vários perfis (estagiário, suporte, analista) e observará como usuários com uma ou várias associações de perfil veem diferentes níveis de dados mascarados. Os exemplos também abrangem o comportamento de mascaramento em instruções do DML com cláusulas RETURNING, gatilhos em tabelas em comparação a visualizações e operações de gerenciamento de políticas, incluindo renomeação, alteração de pesos e limpeza.

1. Crie uma tabela de exemplo com alguns dados sensíveis:

```
CREATE SCHEMA hr;

CREATE TABLE hr.employees (
    id INT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT,
    ssn TEXT,
    salary NUMERIC(10,2)
 );

INSERT INTO hr.employees VALUES
    (1, 'John Doe', 'john.doe@example.com', '123-45-6789', 50000.00),
```

```
    (2, 'Jane Smith', 'jane.smith@example.com', '987-65-4321', 60000.00);
```

2. Crie funções de mascaramento personalizadas:

```
CREATE OR REPLACE FUNCTION public.mask_ssn(ssn TEXT)
    RETURNS TEXT AS $$
    BEGIN
        RETURN 'XXX-XX-' || RIGHT(ssn, 4);
    END;
    $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION public.mask_salary(salary NUMERIC, multiplier NUMERIC
 DEFAULT 0.0)
    RETURNS NUMERIC AS $$
    BEGIN
        RETURN salary * multiplier;
    END;
    $$ LANGUAGE plpgsql;
```

3. Crie várias políticas com diferentes níveis de mascaramento com base em perfis de usuários:

```
-- Create different roles
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE intern_role;

GRANT USAGE ON SCHEMA hr TO analyst_role, support_role, intern_role;
GRANT SELECT ON hr.employees TO analyst_role, support_role, intern_role;
-----------------------------------------------------------------------

-- Low-Weight Policy (Intern)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    JSON_BUILD_OBJECT('name', 'pgcolumnmask.mask_text(name, ''*'')',
                      'email', 'pgcolumnmask.mask_email(email)',
                      'ssn', 'pgcolumnmask.mask_text(ssn, ''*'')',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['intern_role'],
    10  -- Lowest weight
);

-----------------------------------------------------------------------
```

```
-- Medium-Weight Policy (Support)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_moderate',
    'hr.employees',
    JSON_BUILD_OBJECT('email', 'pgcolumnmask.mask_email(email, ''#'')',
                      'ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['support_role'],
    50   -- Medium weight
);

-------------------------------------------------------------------------
-- High-Weight Policy (Analyst)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_light',
    'hr.employees',
    JSON_BUILD_OBJECT('ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary, 0.9)')::JSONB,
    ARRAY['analyst_role'],
    100   -- Highest weight
);
```

4. Os exemplos a seguir demonstram como diferentes usuários veem os dados com base em sua associação de perfil e nos pesos das políticas.

```
-- Create users
CREATE USER sarah_intern;
GRANT intern_role TO sarah_intern;

CREATE USER lisa_support;
GRANT support_role TO lisa_support;

CREATE USER mike_analyst;
GRANT analyst_role TO mike_analyst;

CREATE USER ethan_support_intern;
GRANT support_role, intern_role TO ethan_support_intern;

CREATE USER john_analyst_intern;
GRANT analyst_role, intern_role TO john_analyst_intern;
```

Como estagiário (mascaramento mais rigoroso):

```
SET ROLE sarah_intern;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | ********** |   0.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | ********** |   0.00
```

Como usuário de suporte (mascaramento moderado):

```
SET ROLE lisa_support;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | John Doe   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | Jane Smith | ##########@#######.com | XXX-XX-4321 |   0.00
```

Como analista (mascaramento mais leve):

```
SET ROLE mike_analyst;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
  2 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
```

Como o usuário ethan_support_intern, que é estagiário e usuário de suporte:

```
SET ROLE ethan_support_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_moderate
-- id : unmasked because no masking policy appliable on ethan_support_intern
--            masks these columns
-- name : masked because of employee_mask_strict policy
-- email, ssn, salary : both employee_mask_strict and employee_mask_moderate mask
 these columns
```

```
--                     but employee_mask_moderate will be use because of higher
 weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | ********** | ##########@#######.com | XXX-XX-4321 |   0.00
```

Como john_analyst_intern, que é, ao mesmo tempo, estagiário e analista:

```
SET ROLE john_analyst_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_light
-- id : unmasked because no masking policy appliable on john_analyst_intern
--            masks these columns
-- name, email : masked because of employee_mask_strict
-- ssn, salary : both employee_mask_strict and employee_mask_light mask these
 columns
--               but employee_mask_light will be use because of higher weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | XXX-XX-6789 | 45000.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | XXX-XX-4321 | 54000.00
```

Noções básicas sobre o comportamento de mascaramento em operações de DML

`pg_columnmask` é aplicado de modo consistente a todas as operações do DML, incluindo INSERT, UPDATE, DELETE e MERGE. Quando você executa essas operações, o Aurora PostgreSQL mascara os dados de acordo com um princípio fundamental: qualquer dado lido do armazenamento é mascarado de acordo com as políticas aplicáveis do usuário atual.

O mascaramento afeta alguns dos seguintes componentes de consulta, como:

- Cláusulas WHERE
- Condições JOIN
- Subconsultas

- Cláusulas RETURNING

Todos esses componentes operam com valores mascarados, não com os dados originais. Enquanto os dados são gravados no armazenamento sem máscara, os usuários só acessam a visualização mascarada ao lê-la novamente.

O Aurora PostgreSQL impõe todas as restrições do banco de dados (NOT NULL, UNIQUE, CHECK, FOREIGN KEY) nos valores reais armazenados, não nos valores mascarados. Ocasionalmente, isso poderá criar inconsistências aparentes se as funções de mascaramento não forem cuidadosamente projetadas.

O mascaramento funciona com permissões em nível de coluna:

- Usuários sem privilégios SELECT não podem ler colunas.
- Usuários com privilégios SELECT veem valores mascarados de acordo com as políticas aplicáveis.

Noções básicas sobre o comportamento de mascaramento nas funções de gatilho

Quando políticas `pg_columnmask` são aplicadas às tabelas, é importante entender como o mascaramento interage com as funções de gatilho. Gatilhos são funções de banco de dados que são executadas automaticamente em resposta a determinados eventos em uma tabela, como as operações INSERT, UPDATE ou DELETE.

Por padrão, o DDM aplica regras de mascaramento diferentes, dependendo do tipo de gatilho:

Gatilhos de tabela

Tabelas de transição não são mascaradas: as funções de gatilho nas tabelas têm acesso aos dados não mascarados em suas tabelas de transição para versões de linha antigas e novas.

Os proprietários de tabelas criam gatilhos e são proprietários dos dados, para que tenham acesso total para gerenciar suas tabelas de forma eficaz

Visualizar gatilhos (gatilhos INSTEAD OF)

As tabelas de transição são mascaradas: as funções de gatilho nas visualizações veem os dados mascarados de acordo com as permissões do usuário atual.

Os proprietários das visualizações podem ser diferentes dos proprietários da tabela base e devem respeitar as políticas de mascaramento nas tabelas subjacentes.

Dois parâmetros de configuração em nível de servidor controlam o comportamento do gatilho com tabelas mascaradas. Eles só podem ser definidos por `rds_superuser`:

- Restringir gatilhos em tabelas mascaradas: impede a execução de gatilhos quando um usuário mascarado executa operações do DML em tabelas com políticas de mascaramento aplicáveis.
- Restringir gatilhos em visualizações com tabelas mascaradas: impede a execução de gatilhos em visualizações quando a definição da visualização inclui tabelas com políticas de mascaramento aplicáveis ao usuário atual.

Example das diferenças entre a aplicação da função à tabela e à visualização

O exemplo a seguir cria uma função de gatilho que imprime valores de linha antigos e novos e, depois, demonstra como a mesma função se comporta de forma diferente quando anexada a uma tabela em comparação a uma visualização.

```
-- Create trigger function
CREATE OR REPLACE FUNCTION print_changes()
    RETURNS TRIGGER AS
    $$
        BEGIN
        RAISE NOTICE 'Old row: name=%, email=%, ssn=%, salary=%',
            OLD.name, OLD.email, OLD.ssn, OLD.salary;

        RAISE NOTICE 'New row: name=%, email=%, ssn=%, salary=%',
            NEW.name, NEW.email, NEW.ssn, NEW.salary;

        RETURN NEW;
        END;
    $$ LANGUAGE plpgsql;

-- Create trigger
CREATE TRIGGER print_changes_trigger
    BEFORE UPDATE ON hr.employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Grant update to analyst role
GRANT UPDATE ON hr.employees TO analyst_role;

-- Unmasked data must be seen inside trigger even for masked user for the OLD and NEW
-- row passed to trigger function
```

```
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)

ROLLBACK;


-- Triggers on views (which are supposed to see masked data for new/old row)
CREATE VIEW hr.view_over_employees AS SELECT * FROM hr.employees;
GRANT UPDATE, SELECT ON hr.view_over_employees TO analyst_role;

-- Create trigger for this view
CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON hr.view_over_employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Masked new and old rows should be passed to trigger if trigger is on view
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.view_over_employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
```

```
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)
ROLLBACK;
```

Recomendamos revisar o comportamento dos gatilhos antes de implementá-los em tabelas mascaradas. Os gatilhos de tabela têm acesso a dados não mascarados em tabelas em transição, enquanto os gatilhos de visualização veem dados mascarados.

Example de renomear uma política de mascaramento

O exemplo a seguir demonstra como renomear políticas existentes usando o procedimento `rename_masking_policy`.

```
-- Rename the strict policy
CALL pgcolumnmask.rename_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    'intern_protection_policy'
);

-- Verify the rename
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;

        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     50
 intern_protection_policy | {intern_role}  |     10
```

Example de alterar o peso da política

O exemplo a seguir demonstra como alterar o peso da política para alterar seu peso.

```
-- Change weight of moderate policy
CALL pgcolumnmask.alter_masking_policy(
    'employee_mask_moderate'::NAME,
    'hr.employees'::REGCLASS,
    NULL,    -- Keep existing masking expressions
    NULL,    -- Keep existing roles
```

```
    75       -- New weight
);

-- Verify the changes
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;
        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     75
 intern_protection_policy | {intern_role}  |     10
```

Example de como limpar

O exemplo a seguir demonstra como eliminar todas as políticas, tabelas e usuários.

```
-- Drop policies
CALL pgcolumnmask.drop_masking_policy(
    'intern_protection_policy',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_moderate',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_light',
    'hr.employees'
);

-- Drop table and functions
DROP VIEW IF EXISTS hr.view_over_employees;
DROP TABLE IF EXISTS hr.employees;
DROP SCHEMA IF EXISTS hr;
DROP FUNCTION IF EXISTS public.mask_ssn(text);
DROP FUNCTION IF EXISTS public.mask_salary(numeric, numeric);

-- Drop users
DROP USER sarah_intern, lisa_support, mike_analyst,
    ethan_support_intern, john_analyst_intern;
```

```
DROP ROLE intern_role, support_role, analyst_role;
```

Configurar o perfil de gerenciamento de políticas de mascaramento

A extensão de mascaramento de colunas do PostgreSQL, `pg_columnmask`, permite delegar o gerenciamento de políticas de mascaramento a um perfil específico, em vez de exigir privilégios de proprietário da tabela ou `rds_superuser`. Isso fornece um controle mais granular sobre quem pode criar, alterar e eliminar políticas de mascaramento.

Para configurar o perfil que terá privilégios de gerenciamento de políticas de mascaramento, siga estas etapas:

1. Crie o perfil de administrador de políticas: como `rds_superuser`, crie um perfil responsável pelo gerenciamento de políticas de mascaramento:

   ```
   CREATE ROLE mask_admin NOLOGIN;
   ```

2. Configure o parâmetro PostgreSQL: em seu grupo de parâmetros de cluster de banco de dados personalizado, defina o parâmetro de configuração do mecanismo `pgcolumnmask.policy_admin_rolname` com o nome do perfil que você criou:

   ```
   pgcolumnmask.policy_admin_rolname = mask_admin
   ```

   Esses parâmetros de configuração do mecanismo podem ser definidos em um grupo de parâmetros do cluster de banco de dados e não exigem a reinicialização da instância. Para acessar detalhes sobre a atualização de parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

3. Conceda o perfil aos usuários como `rds_superuser`, conceda o perfil `mask_admin` aos usuários que devem ser capazes de gerenciar políticas de mascaramento:

   ```
   CREATE USER alice LOGIN;
   CREATE USER bob LOGIN;
   GRANT mask_admin TO alice, bob;
   ```

   Além disso, garanta que os usuários tenham o privilégio USAGE nos esquemas em que gerenciarão as políticas de mascaramento:

   ```
   GRANT USAGE ON SCHEMA hr TO alice, bob;
   ```

Agora, quando os usuários `alice` e `bob` se conectam ao banco de dados, eles podem usar as funções de extensão `pg_columnmask` padrão para criar, alterar e eliminar políticas de mascaramento em todas as tabelas, em todos os esquemas em que eles têm privilégios `USAGE` no esquema.

Práticas recomendadas para a implementação segura de pg_columnmask

A seção a seguir fornece as práticas recomendadas de segurança para implementação de `pg_columnmask` em seu ambiente Aurora PostgreSQL. Siga estas recomendações para:

- Estabelecer uma arquitetura de controle de acesso com base em perfil seguro
- Desenvolver funções de mascaramento que evitem vulnerabilidades de segurança
- Entender e controlar o comportamento do gatilho com dados mascarados

Arquitetura de segurança baseada em perfis

Defina uma hierarquia de perfis para implementar controles de acesso em seu banco de dados. O Aurora PostgreSQL `pg_columnmask` aumenta esses controles fornecendo uma camada adicional para mascaramento de dados refinado dentro desses perfis.

Criar perfis dedicados que se alinhem às funções organizacionais em vez de conceder permissões a usuários individuais. Essa abordagem fornece melhor capacidade de auditoria e simplifica o gerenciamento de permissões à medida que sua estrutura organizacional evolui.

Example de criar uma hierarquia de perfis organizacionais

O exemplo a seguir cria uma hierarquia de perfis organizacionais com perfis dedicados para diferentes funções e, depois, atribui usuários individuais aos perfis apropriados. Neste exemplo, os perfis organizacionais (analyst_role, support_role) são criados primeiro e, depois, os usuários individuais recebem a associação a esses perfis. Essa estrutura permite que você gerencie as permissões em nível de perfil, e não de cada usuário individual.

```
-- Create organizational role hierarchy
CREATE ROLE data_admin_role;
CREATE ROLE security_admin_role;
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE developer_role;

-- Specify security_admin_role as masking policy manager in the DB cluster parameter
```

```
-- group pgcolumnmask.policy_admin_rolname = 'security_admin_role'

-- Create specific users and assign to appropriate roles
CREATE USER security_manager;
CREATE USER data_analyst1, data_analyst2;
CREATE USER support_agent1, support_agent2;

GRANT security_admin_role TO security_manager;
GRANT analyst_role TO data_analyst1, data_analyst2;
GRANT support_role TO support_agent1, support_agent2;
```

Implemente o princípio de privilégio mínimo concedendo apenas o mínimo de permissões necessárias para cada perfil. Evite conceder permissões amplas que poderiam ser exploradas se as credenciais fossem comprometidas.

```
-- Grant specific table permissions rather than schema-wide access
GRANT SELECT ON sensitive_data.customers TO analyst_role;
GRANT SELECT ON sensitive_data.transactions TO analyst_role;
-- Do not grant: GRANT ALL ON SCHEMA sensitive_data TO analyst_role;
```

Os administradores de políticas exigem privilégios USAGE nos esquemas em que gerenciam políticas de mascaramento. Conceda esses privilégios de modo seletivo, seguindo o princípio do privilégio mínimo. Realize análises regulares das permissões de acesso ao esquema para garantir que somente o pessoal autorizado mantenha os recursos de gerenciamento de políticas.

A configuração dos parâmetros do perfil de administrador da política é restrita somente aos administradores do banco de dados. Esse parâmetro não pode ser modificado em nível de banco de dados ou de sessão, impedindo que usuários sem privilégios substituam as atribuições do administrador da política. Essa restrição garante que o controle da política de mascaramento permaneça centralizado e seguro.

Atribua o perfil de administrador de políticas a indivíduos específicos em vez de grupos. Essa abordagem direcionada garante acesso seletivo ao gerenciamento de políticas de mascaramento, pois os administradores de políticas têm a capacidade de mascarar todas as tabelas no banco de dados.

Desenvolvimento seguro da função de mascaramento

Desenvolva funções de mascaramento usando semântica de vinculação antecipada para garantir o monitoramento adequado de dependências e evitar vulnerabilidades de vinculação tardia,

como modificação do caminho de pesquisa durante o runtime. É recomendável usar a sintaxe `BEGIN ATOMIC` para funções SQL a fim de permitir a validação em tempo de compilação (ou seja, vinculação antecipada) e o gerenciamento de dependências.

```
-- Example - Secure masking function with early binding
CREATE OR REPLACE FUNCTION secure_mask_ssn(input_ssn TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN input_ssn IS NULL THEN NULL
            WHEN length(input_ssn) < 4 THEN repeat('X', length(input_ssn))
            ELSE repeat('X', length(input_ssn) - 4) || right(input_ssn, 4)
        END;
    END;
```

Como alternativa, crie funções imunes às alterações do caminho de pesquisa qualificando explicitamente o esquema de todas as referências de objetos, garantindo um comportamento consistente em diferentes sessões de usuário.

```
-- Function immune to search path changes
CREATE OR REPLACE FUNCTION data_masking.secure_phone_mask(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    AS $$
    SELECT CASE
        WHEN phone_number IS NULL THEN NULL
        WHEN public.length(public.regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10
 THEN 'XXX-XXX-XXXX'
        ELSE public.regexp_replace(
            phone_number,
            '([0-9]{3})[0-9]{3}([0-9]{4})',
            public.concat('\1-XXX-\2')
        )
    END;
    $$;
```

Implemente a validação de entrada nas funções de mascaramento para lidar com casos de borda e evitar comportamentos inesperados. Sempre inclua o tratamento de NULL e valide os formatos de entrada para garantir um comportamento consistente de mascaramento.

```
-- Robust masking function with comprehensive input validation
CREATE OR REPLACE FUNCTION secure_mask_phone(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN phone_number IS NULL THEN NULL
            WHEN length(trim(phone_number)) = 0 THEN phone_number
            WHEN length(regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10 THEN
 'XXX-XXX-XXXX'
            ELSE regexp_replace(phone_number, '([0-9]{3})[0-9]{3}([0-9]{4})', '\1-XXX-
\2')
        END;
    END;
```

Comportamento dos gatilhos do DML com pg_columnmask

Para gatilhos de tabela, as tabelas de transição serão totalmente sem máscara. Para gatilhos de visualização (IOT), as tabelas de transição serão mascaradas de acordo com as permissões de visualização do usuário atual.

Gatilhos de tabela com pg_columnmask

Os gatilhos recebem uma tabela de transição que contém a versão antiga e a nova das linhas modificadas pela consulta do DML de acionamento. Dependendo de quando o gatilho é acionado, o Aurora PostgreSQL preenche as linhas antiga e nova. Por exemplo, um gatilho `BEFORE INSERT` só tem novas versões das linhas e versões antigas vazias porque não há uma versão antiga para consultar.

O `pg_columnmask` não mascara as tabelas de transição dentro dos gatilhos das tabelas. Os gatilhos podem usar colunas mascaradas dentro de seu corpo e ver dados não mascarados. O criador do gatilho deve se certificar de como ele é executado por um usuário. O exemplo a seguir funciona corretamente nesse caso.

```
-- Example for table trigger uses masked column in its definition
-- Create a table and insert some rows
CREATE TABLE public.credit_card_table (
    name TEXT,
    credit_card_no VARCHAR(16),
    is_fraud BOOL
```

```
);

INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

-- Create a role which will see masked data and grant it privileges
CREATE ROLE intern_user;
GRANT SELECT, DELETE ON public.credit_card_table TO intern_user;

-- Trigger which will silenty skip delete of non fraudelent credit cards
CREATE OR REPLACE FUNCTION prevent_non_fraud_delete()
    RETURNS TRIGGER AS
    $$
    BEGIN
        IF OLD.is_fraud = false THEN
            RETURN NULL;
        END IF;
        RETURN OLD;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_non_fraud_delete
    BEFORE DELETE ON credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION prevent_non_fraud_delete();

CREATE OR REPLACE FUNCTION public.return_false()
    RETURNS BOOLEAN
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
      SELECT false;
    END;

-- A masking policy that masks both credit card number and is_fraud column.
-- If we apply masking inside trigger then prevent_non_fraud_delete trigger will
-- allow deleting more rows to masked user (even non fraud ones).
CALL pgcolumnmask.create_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    JSON_BUILD_OBJECT('credit_card_no', 'pgcolumnmask.mask_text(credit_card_no)',
```

```
                      'is_fraud', 'public.return_false()')::JSONB,
    ARRAY['intern_user']::NAME[],
    10::INT
);

-- Test trigger behaviour using intern_user
BEGIN;
SET ROLE intern_user;
-- credit card number & is_fraud is completely masked from intern_user
SELECT * FROM public.credit_card_table;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | XXXXXXXXXXXXXXXX | f
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(3 rows)

-- The delete trigger lets the intern user delete rows for Jane and Brad even though
-- intern_user sees their is_fraud = false, but the table trigger works with
 original
-- unmasked value
DELETE FROM public.credit_card_table RETURNING *;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(2 rows)

COMMIT;
```

O criador do gatilho vazará dados não mascarados para o usuário se ele não tiver cuidado com as declarações que usa no corpo do gatilho. Por exemplo, usar `RAISE NOTICE '%', masked_column;` imprime a coluna para o usuário atual.

```
-- Example showing table trigger leaking column value to current user
CREATE OR REPLACE FUNCTION leaky_trigger_func()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old credit card number was: %', OLD.credit_card_no;
        RAISE NOTICE 'New credit card number is %', NEW.credit_card_no;
        RETURN NEW;
```

```
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER leaky_trigger
    AFTER UPDATE ON public.credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION leaky_trigger_func();

-- Grant update on column is_fraud to auditor role
-- auditor will NOT HAVE PERMISSION TO READ DATA
CREATE ROLE auditor;
GRANT UPDATE (is_fraud) ON public.credit_card_table TO auditor;

-- Also add auditor role to existing masking policy on credit card table
CALL pgcolumnmask.alter_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    NULL::JSONB,
    ARRAY['intern_user', 'auditor']::NAME[],
    NULL::INT
);

-- Log in as auditor
-- [auditor]
-- Update will fail if trying to read data from the table
UPDATE public.credit_card_table
    SET is_fraud = true
    WHERE credit_card_no = '4532015112830366';
ERROR:  permission denied for table cc_table

-- [auditor]
-- But leaky update trigger will still print the entire row even though
-- current user does not have permission to select from public.credit_card_table
UPDATE public.credit_card_table SET is_fraud = true;
NOTICE:  Old credit_card_no was: 4532015112830366
NOTICE:  New credit_card_no is 4532015112830366
```

Gatilhos em visualizações com pg_columnmask (em vez de gatilhos)

Os gatilhos só podem ser criados em visualizações no PostgreSQL. Eles são usados para executar instruções do DML em visualizações que não são atualizáveis. As tabelas de trânsito são sempre mascaradas internamente em vez de no gatilho (IOT), porque a visualização e as tabelas base usadas na consulta de visualização podem ter proprietários diferentes. Nesse

caso, as tabelas base podem ter algumas políticas de mascaramento aplicáveis ao proprietário da visualização e este deve sempre ver os dados mascarados das tabelas base dentro dos respectivos gatilhos. Isso é diferente dos gatilhos nas tabelas porque, nesse caso, o criador do gatilho e os dados dentro das tabelas pertencem ao mesmo usuário, o que não é o caso aqui.

```
-- Create a view over credit card table
CREATE OR REPLACE VIEW public.credit_card_view
    AS
    SELECT * FROM public.credit_card_table;

-- Truncate credit card table and insert fresh data
TRUNCATE TABLE public.credit_card_table;
INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

CREATE OR REPLACE FUNCTION public.print_changes()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old row: name=%, credit card number=%, is fraud=%',
            OLD.name, OLD.credit_card_no, OLD.is_fraud;

        RAISE NOTICE 'New row: name=%, credit card number=%, is fraud=%',
            NEW.name, NEW.credit_card_no, NEW.is_fraud;

    RETURN NEW;
   END;
   $$ LANGUAGE plpgsql;

CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON public.credit_card_view
    FOR EACH ROW
    EXECUTE FUNCTION public.print_changes();

GRANT SELECT, UPDATE ON public.credit_card_view TO auditor;

-- [auditor]
-- Login as auditor role
BEGIN;
```

```
-- Any data coming out from the table will be masked in instead of triggers
-- according to masking policies applicable to current user
UPDATE public.credit_card_view
    SET name = CONCAT(name, '_new_name')
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
        name         |  credit_card_no  | is_fraud
---------------------+------------------+----------
 John Doe_new_name   | XXXXXXXXXXXXXXXX | f
 Jane Smith_new_name | XXXXXXXXXXXXXXXX | f
 Brad Smith_new_name | XXXXXXXXXXXXXXXX | f

 -- Any new data going into the table using INSERT or UPDATE command will be
 unmasked
 UPDATE public.credit_card_view
    SET credit_card_no = '9876987698769876'
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith, credit card number=9876987698769876, is fraud=f
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | 9876987698769876 | f
 Jane Smith | 9876987698769876 | f
 Brad Smith | 9876987698769876 | f

 COMMIT;
```

GuCs em nível de banco de dados/usuário para controlar o comportamento dos gatilhos

Dois parâmetros de configuração controlam o comportamento de execução do gatilho para usuários com políticas de mascaramento aplicáveis. Use esses parâmetros para evitar que

os gatilhos sejam executados em tabelas ou visualizações mascaradas quando restrições de segurança adicionais forem necessárias. Ambos os parâmetros são desabilitados por padrão, permitindo que os gatilhos sejam executados normalmente.

Primeiro GUC: restrição de disparo de gatilho em tabelas mascaradas.

Especificações:

- Nome: : `pgcolumnmask.restrict_dml_triggers_for_masked_users`
- Tipo:: `boolean`
- Padrão: `false` (os gatilhos podem ser executados).

Impede a execução do gatilho em tabelas mascaradas para usuários mascarados quando definido como TRUE. `pg_columnmask` é executado ocasionando um erro.

Segundo GUC: restrição de acionamento do gatilho em visualizações com tabelas mascaradas.

Especificações:

- Nome: : `pgcolumnmask.restrict_iot_triggers_for_masked_users`
- Tipo:: `boolean`
- Padrão: `false` (os gatilhos podem ser executados).

Impede a execução de gatilhos em visualizações que incluem tabelas mascaradas em sua definição para usuários mascarados quando definidas como TRUE.

Esses parâmetros funcionam de forma independente e são configuráveis, como os parâmetros padrão de configuração do banco de dados.

Cenários de movimentação de dados do pg_columnmask do Aurora PostgreSQL

O comportamento de `pg_columnmask` varia nas diferentes operações de movimentação de dados, dependendo se a operação ocorre na camada de armazenamento, lógica ou de aplicação. As operações em nível de armazenamento (como clonagem) se comportam de maneira diferente das operações lógicas (como `pg_dump`) e das operações em nível de aplicação (como consultas FDW). Esta seção descreve o comportamento de mascaramento em cenários comuns, incluindo replicação, backups, exportações e migrações, e explica as implicações de segurança de cada um.

Tópicos

- Aurora Global Database e réplicas de leitura
- Clone de banco de dados e restauração de snapshot
- Replicação lógica
- Implantações azuis/verdes
- Fluxos ETL zero e CDC
- AWS Database Migration Service
- Exportações de dados
- Visualizações e visões materializadas
- Despejo e restauração de dados
- Wrapper de dados externos

Aurora Global Database e réplicas de leitura

As políticas `pg_columnmask` do Aurora são armazenadas em tabelas do sistema de banco de dados dentro do volume do cluster. Todas as réplicas acessam as mesmas políticas e exibem resultados consistentemente mascarados. Para implantações do Aurora Global Database, as políticas `pg_columnmask` são replicadas para as Regiões da AWS secundárias junto com outras tabelas do sistema de banco de dados, garantindo proteção de dados consistente em todas as regiões. Durante os cenários de failover, todas as políticas `pg_columnmask` permanecem intactas e funcionais.

Clone de banco de dados e restauração de snapshot

As operações de clone rápido e restauração de snapshot do Aurora preservam todas as políticas `pg_columnmask`, perfis e configurações como parte das tabelas do sistema de banco de dados. O banco de dados clonado ou restaurado herda todas as políticas existentes do cluster de origem. Após a clonagem ou a restauração, cada cluster de banco de dados mantém políticas `pg_columnmask` independentes.

Replicação lógica

Durante a sincronização inicial, a replicação lógica usa operações SQL COPY padrão e as políticas `pg_columnmask` são aplicadas com base nas permissões do usuário de replicação. Durante o CDC (captura de dados de alteração) em andamento, as políticas de mascaramento não são aplicadas e os dados não mascarados são replicados por meio de registros do WAL. Usuários com privilégios

`pg_create_subscription` podem exfiltrar dados não mascarados configurando a replicação em um sistema que eles controlem.

Implantações azuis/verdes

Durante a restauração do snapshot, as políticas `pg_columnmask` são incluídas automaticamente. O ambiente verde começa com uma cópia idêntica de todas as políticas do ambiente azul. Durante a replicação do azul para o verde, os dados não são mascarados. Alterações subsequentes da política de mascaramento (comandos DDL) no cluster azul não são replicadas para o cluster verde e invalidam as implantações azul/verde do RDS.

Fluxos ETL zero e CDC

A replicação de dados não é afetada pelas políticas `pg_columnmask`. O ETL zero comporta a replicação de DDL, mas não replica as políticas de RLS nem `pg_columnmask`. Nenhuma política de mascaramento é aplicada aos dados replicados em ETL zero.

AWS Database Migration Service

A sincronização inicial de dados é mascarada ou não mascarada com base no usuário selecionado para a tarefa do DMS. Os dados do CDC nunca são mascarados. Embora as políticas de RLS internas relacionadas a `pg_columnmask` possam ser migradas, elas não funcionarão em destinos não habilitados para pg_columnmask.

Exportações de dados

A `pg_columnmask` trata as exportações como qualquer outra operação de consulta: o mascaramento é aplicado com base nas permissões do usuário em execução. Isso se aplica a comandos SQL, como COPY, SELECT INTO, CREATE TABLE AS e à funcionalidade de exportação do S3 do Aurora PostgreSQL.

> **Note**
>
> Quando usuários mascarados exportam dados, os arquivos resultantes contêm valores mascarados que podem violar as restrições do banco de dados quando restaurados.

Visualizações e visões materializadas

Lembre-se das seguintes considerações ao utilizar visualizações:

- Visualizações regulares: sempre usam a semântica `INVOKER`. As políticas de mascaramento do usuário atual se aplicam ao consultar a visualização, independentemente de quem a criou.
- Visões materializadas: quando atualizadas, as políticas de mascaramento do proprietário da visão materializada se aplicam, não as políticas do usuário que realiza a atualização. Se o proprietário tiver políticas de mascaramento, a visão materializada sempre conterá dados mascarados.

Despejo e restauração de dados

`pg_dump` funciona como um usuário regular do banco de dados e aplica políticas de mascaramento com base nas permissões do usuário conectado. Se um usuário mascarado realizar um despejo, o arquivo de backup conterá dados mascarados. As politicas `pg_columnmask` são incluídas no despejo como parte do esquema do banco de dados. A restauração bem-sucedida exige que todos os perfis referidos existam no banco de dados de destino e que o destino tenha a extensão `pg_columnmask` instalada.

> **Note**
>
> A partir do PostgreSQL 18, `pg_dump` comporta a opção `–no-policies` que exclui tanto a segurança por linha (RLS) e as políticas de mascaramento `pg_columnmask` dos despejos do banco de dados. Para acessar mais informações, consulte pg_dump.

Wrapper de dados externos

Ao usar wrappers de dados externos, as políticas de mascaramento em tabelas remotas são aplicadas com base nas permissões do usuário associado no servidor de origem, não nas permissões do usuário de consulta local. Embora você possa acessar dados remotos mascarados por meio do FDW, não pode criar políticas de DDM ou RLS diretamente em tabelas externas em seu banco de dados local.

Atualizar aplicações para conexão com clusters de banco de dados PostgreSQL do Aurora usando novos certificados SSL/TLS

Em 13 de janeiro de 2023, o Amazon RDS publicou novos certificados de autoridade de certificação (CA) para conexão com clusters de banco de dados do Aurora via Secure Socket Layer ou Transport Layer Security (SSL/TLS). A seguir, você pode encontrar informações sobre como atualizar seus aplicativos para usar os novos certificados.

Este tópico pode ajudá-lo a determinar se algum aplicativo cliente usa SSL/TLS para conectar-se aos seus clusters de banco de dados. Em caso positivo, examine mais a fundo se esse aplicativo exige verificação de certificado para se conectar.

> **Note**
>
> Alguns aplicativos são configurados para se conectar a clusters de banco de dados PostgreSQL do Aurora apenas quando podem verificar com sucesso o certificado no servidor.
> Para esses aplicativos, você deve atualizar os repositórios confiáveis de aplicativos cliente para incluir os novos certificados de CA.

Depois de atualizar seus certificados de CA nos armazenamentos confiáveis do aplicativo cliente, você pode fazer o rodízio dos certificados nos seus clusters de banco de dados. É altamente recomendável testar esses procedimentos em um ambiente de desenvolvimento ou teste antes de implementá-los em seus ambientes de produção.

Para obter mais informações sobre a mudança de certificados, consulte Alternar o certificado SSL/TLS. Para ter mais informações sobre como fazer download de certificados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados. Para obter informações sobre como usar o SSL/TLS com clusters de banco de dados PostgreSQL, consulte Como proteger dados do Aurora PostgreSQL com SSL/TLS.

Tópicos

- Determinar se as aplicações estão se conectando a clusters de banco de dados PostgreSQL do Aurora usando SSL
- Determinar se um cliente requer verificação de certificado para se conectar
- Atualizar o armazenamento confiável de aplicações
- Usar conexões SSL/TLS para diferentes tipos de aplicações

Determinar se as aplicações estão se conectando a clusters de banco de dados PostgreSQL do Aurora usando SSL

Verifique a configuração do cluster de banco de dados para obter o valor do parâmetro `rds.force_ssl`. Por padrão, o parâmetro `rds.force_ssl` é definido como `0` (desativado). Se o

parâmetro `rds.force_ssl` estiver definido como 1 (ativado), os clientes serão obrigados a usar SSL/TLS para conexões. Para ter mais informações sobre parameter groups, consulte Grupos de parâmetros para Amazon Aurora.

Se `rds.force_ssl` não estiver definido como 1 (ligado), consulte a exibição `pg_stat_ssl` para verificar conexões usando SSL. Por exemplo, a consulta a seguir retorna apenas conexões SSL e informações sobre os clientes que usam SSL.

```
select datname, usename, ssl, client_addr from pg_stat_ssl inner join pg_stat_activity
 on pg_stat_ssl.pid = pg_stat_activity.pid where ssl is true and usename<>'rdsadmin';
```

Somente as linhas que usarem conexões SSL/TLS serão exibidas com informações sobre a conexão. Esta é uma saída de exemplo.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

A consulta anterior exibe apenas as conexões atuais no momento da consulta. A ausência de resultados não indica que nenhum aplicativo esteja usando conexões SSL. Outras conexões SSL podem ser estabelecidas em um momento diferente.

Determinar se um cliente requer verificação de certificado para se conectar

Quando um cliente, como psql ou JDBC, é configurado com suporte para SSL, primeiro ele tenta se conectar ao banco de dados via SSL por padrão. Se esse cliente não puder se conectar via SSL, ele voltará a se conectar sem SSL. O modo padrão `sslmode` utilizado para clientes baseados em libpq (como o psql) e o JDBC é definido como `prefer`. O certificado no servidor é verificado apenas quando `sslrootcert` é fornecido com `sslmode` definido como `verify-ca` ou `verify-full`. Um erro será lançado se o certificado for inválido.

Use `PGSSLROOTCERT` para verificar o certificado com a variável de ambiente `PGSSLMODE`, com `PGSSLMODE` definido como `verify-ca` ou `verify-full`.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U primaryuser -d postgres
```

Use o argumento `sslrootcert` para verificar o certificado com `sslmode` no formato de string de conexão, com `sslmode` definido como `verify-ca` ou `verify-full`.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=primaryuser dbname=postgres"
```

Por exemplo, no caso anterior, se você usar um certificado raiz inválido, verá um erro semelhante ao seguinte no seu cliente.

```
psql: SSL error: certificate verify failed
```

Atualizar o armazenamento confiável de aplicações

Para obter informações sobre como atualizar o armazenamento de confiança para aplicações PostgreSQL, consulte Secure TCP/IP connections with SSL na documentação do PostgreSQL.

> **Note**
>
> Ao atualizar o armazenamento confiável, é possível reter certificados mais antigos, além de adicionar os novos certificados.

Atualizar o armazenamento confiável de aplicações para JDBC

Você pode atualizar o armazenamento confiável para aplicativos que usam conexões JDBC para SSL/TLS.

Para obter informações sobre como baixar o certificado raiz, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

Para obter exemplos de scripts que importam certificados, consulte Script de exemplo para importar certificados para o seu armazenamento confiável.

Usar conexões SSL/TLS para diferentes tipos de aplicações

Veja a seguir informações sobre o uso de conexões SSL/TLS para diferentes tipos de aplicativos:

- psql

O cliente é chamado da linha de comandos especificando opções como uma string de conexão ou como variáveis de ambiente. Para conexões SSL/TLS, as opções relevantes são `sslmode` (variável de ambiente `PGSSLMODE`), `sslrootcert` (variável de ambiente `PGSSLROOTCERT`).

Para conhecer a lista completa de opções, consulte Palavras-chave de parâmetros na documentação do PostgreSQL. Para conhecer a lista completa de variáveis de ambiente, consulte Variáveis de ambiente na documentação do PostgreSQL.

- pgAdmin

  Esse cliente baseado em navegador é uma interface mais amigável para conectar-se a um banco de dados PostgreSQL.

  Para obter informações sobre como configurar conexões, consulte a Documentação de pgAdmin.

- JDBC

  O JDBC permite conexões de banco de dados com aplicativos Java.

  Para obter informações gerais sobre como conectar-se a um banco de dados PostgreSQL com JDBC, consulte Conectar ao banco de dados na documentação do PostgreSQL. Para obter informações sobre como conectar-se com SSL/TLS, consulte Configurar o cliente na documentação do PostgreSQL.

- Python

  Uma biblioteca Python popular para conectar-se a bancos de dados PostgreSQL é `psycopg2`.

  Para obter informações sobre como usar o `psycopg2`, consulte a documentação de psycopg2. Para um breve tutorial sobre como conectar-se a um banco de dados PostgreSQL, consulte Tutorial sobre psycopg2. Você pode encontrar informações sobre as opções aceitas pelo comando de conexão em O conteúdo do módulo psycopg2.

> **Important**
>
> Depois de determinar que suas conexões com o banco de dados usam SSL/TLS e ter atualizado o armazenamento confiável de aplicações, você poderá atualizar o banco de dados para usar os certificados rds-ca-rsa2048-g1. Para obter instruções, consulte a etapa 3 em Atualizar o certificado CA modificando a instância de banco de dados.

Usar a autenticação Kerberos com o Aurora PostgreSQL

É possível usar o Kerberos para autenticar usuários quando eles se conectam ao seu cluster de banco de dados executando o PostgreSQL. Para fazer isso, configure seu cluster de banco de dados para usar o AWS Directory Service for Microsoft Active Directory para autenticação Kerberos. O AWS Directory Service for Microsoft Active Directory também é chamado de AWS Managed Microsoft AD. É um recurso disponível com o Directory Service. Para saber mais, consulte What is Directory Service? (O que é o ?) no Guia de administração do AWS Directory Service.

Para iniciar, crie um diretório AWS Managed Microsoft AD para armazenar credenciais de usuário. Depois, forneça ao cluster de banco de dados PostgreSQL o domínio do Active Directory e outras informações. Quando os usuários são autenticados com o cluster de banco de dados PostgreSQL, as solicitações de autenticação são encaminhadas para o diretório AWS Managed Microsoft AD.

Manter todas as suas credenciais no mesmo diretório pode economizar tempo e esforço. Há um lugar centralizado para armazenar e gerenciar credenciais para vários clusters de banco de dados. O uso de um diretório também pode melhorar o perfil de segurança geral.

Além disso, é possível acessar credenciais de seu próprio Microsoft Active Directory on-premises. Para fazer isso, crie uma relação de domínio confiável para que o diretório AWS Managed Microsoft AD confie no Microsoft Active Directory on-premises. Dessa forma, seus usuários podem acessar as de clusters do PostgreSQL com a mesma experiência de autenticação única (SSO) do Windows, como quando acessam workloads na sua rede on-premises.

Um banco de dados pode usar autenticação Kerberos, do AWS Identity and Access Management (IAM) ou ambas. No entanto, como a autenticação Kerberos e IAM fornecem métodos de autenticação diferentes, um usuário do banco de dados específico somente pode fazer login em um banco de dados usando um ou outro método de autenticação, mas não ambos. Para ter mais informações sobre a autenticação do IAM, consulte Autenticação do banco de dados do IAM.

> **Note**
>
> O RDS para PostgreSQL não aceita a autenticação Kerberos para grupos do Active Directory.

Tópicos

- Disponibilidade de regiões e versões

- Visão geral da autenticação Kerberos para clusters de banco de dados PostgreSQL
- Configurar a autenticação Kerberos para clusters de banco de dados do PostgreSQL
- Gerenciar um cluster de banco de dados do Aurora PostgreSQL em um domínio do Active Directory
- Conectar-se ao PostgreSQL com a autenticação Kerberos
- Usar grupos de segurança do AD para controle de acesso do Aurora PostgreSQL

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e regiões do Aurora PostgreSQL com autenticação do Kerberos, consulte Autenticação do Kerberos com o Aurora PostgreSQL.

Visão geral da autenticação Kerberos para clusters de banco de dados PostgreSQL

Para configurar a autenticação Kerberos para um cluster de banco de dados PostgreSQL, siga as etapas a seguir, descritas em mais detalhes posteriormente:

1. Use AWS Managed Microsoft AD para criar um diretório do AWS Managed Microsoft AD. É possível usar o Console de gerenciamento da AWS, a AWS CLI ou a API do Directory Service para criar o diretório. Certifique-se de abrir as portas de saída relevantes no grupo de segurança do diretório para que o diretório possa se comunicar com o cluster.
2. Crie uma função que forneça ao Amazon Aurora acesso para fazer chamadas para o diretório AWS Managed Microsoft AD. Para fazer isso, crie um perfil do AWS Identity and Access Management (IAM) que use a política gerenciada do IAM `AmazonRDSDirectoryServiceAccess`.

   Para o perfil do IAM permitir acesso, o endpoint do AWS Security Token Service (AWS STS) deve estar ativado na região da AWS correta da conta da AWS. Os endpoints do AWS STS são ativados por padrão em todas as Regiões da AWS e é possível usá-los sem ter que tomar medidas adicionais. Para ter mais informações, consulte Ativar e desativar o AWS STS em uma AWS região da no Manual do usuário do IAM.
3. Crie e configure usuários no diretório AWS Managed Microsoft AD usando as ferramentas do Microsoft Active Directory. Para ter mais informações sobre como criar usuários em seu Active

Directory, consulte Gerenciar usuários e grupos no Microsoft AD gerenciado pela AWS noGuia de administração doDirectory Service.

4. Se você planeja localizar o diretório e a instância de Bancos de Dados em contas da AWS ou nuvens privadas virtuais (VPCs) diferentes, configure o emparelhamento de VPCs. Para ter mais informações, consulte O que é emparelhamento de VPC? no Amazon VPC Peering Guide.
5. Crie ou modifique um cluster de banco de dados PostgreSQL no console, na CLI ou na API do RDS usando um dos seguintes métodos:
   - Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele
   - Modificar um cluster de bancos de dados Amazon Aurora
   - Restauração de um snapshot de um cluster de banco de dados
   - Restaurar um cluster de banco de dados para um horário especificado

   É possível localizar o cluster na mesma Amazon Virtual Private Cloud (VPC) que o diretório ou em uma conta da AWS ou VPC diferente. Ao criar ou modificar o cluster de banco de dados PostgreSQL, faça o seguinte:
   - Forneça o identificador de domínio (identificador `d-*`) que foi gerado quando você criou seu diretório.
   - Forneça o nome do perfil do IAM criado.
   - Certifique-se de que o grupo de segurança da instância de banco de dados possa receber o tráfego de entrada do grupo de segurança do diretório.
6. Use as credenciais de usuário mestre do RDS para conectar-se à de cluster de banco de dados PostgreSQL. Crie o usuário no PostgreSQL para ser identificado externamente. Usuários identificados externamente podem fazer login no cluster de banco de dados PostgreSQL usando a autenticação Kerberos.

Configurar a autenticação Kerberos para clusters de banco de dados do PostgreSQL

Use o AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) para configurar a autenticação Kerberos para um cluster de banco de dados PostgreSQL. Para configurar a autenticação Kerberos, execute as etapas a seguir.

Tópicos

- Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD

- Etapa 2: (Opcional) Criar uma relação de confiança entre o Active Directory on-premises e o Directory Service
- Etapa 3: Criar um perfil do IAM para que o Amazon Aurora acesse o Directory Service
- Etapa 4: Criar e configurar usuários
- Etapa 5: Ativar o tráfego entre VPCs entre o diretório e a instância de banco de dados
- Etapa 6: Criar ou modificar um cluster de banco de dados PostgreSQL
- Etapa 7: Criar usuários do PostgreSQL para suas entidades principais do Kerberos
- Etapa 8: Configurar um cliente PostgreSQL

Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD

O Directory Service cria um Active Directory totalmente gerenciado na Nuvem AWS. Quando você cria um diretório AWS Managed Microsoft AD, o Directory Service cria dois controladores de domínio e servidores DNS para você. Os servidores do diretório são criados em sub-redes diferentes em uma VPC. Essa redundância ajuda a garantir que o diretório permaneça acessível mesmo se ocorrer uma falha.

Ao criar um diretório AWS Managed Microsoft AD, o AWS Directory Service realiza as seguintes tarefas em seu nome:

- Configura um Active Directory dentro da VPC.
- Cria de uma conta de administrador do diretório com o nome de usuário `Admin` e a senha especificada. Use essa conta para gerenciar seu diretório.

  > **Important**
  >
  > Certifique-se de salvar essa senha. O Directory Service não armazena essa senha e não é possível recuperá-la ou redefini-la.

- Cria um grupo de segurança para os controladores do diretório. O grupo de segurança deve permitir a comunicação com o cluster de banco de dados PostgreSQL.

Quando você lança o AWS Directory Service for Microsoft Active Directory, a AWS cria uma Unidade organizacional (OU) que contém todos os objetos do diretório. Essa UO, que tem o nome de NetBIOS que você digitou ao criar o diretório, está localizada na raiz do domínio. A raiz do domínio é controlada e de propriedade da AWS.

A conta `Admin`, que foi criada com o diretório AWS Managed Microsoft AD, tem permissões para as atividades administrativas mais comuns da UO:

- Criar, atualizar ou excluir usuários
- Adicionar recursos ao domínio, como servidores de arquivos ou de impressão, e atribuir permissões para esses recursos aos usuários na UO
- Criar OUs adicionais e contêineres
- Delegar autoridade
- Restaurar objetos excluídos da Lixeira do Active Directory
- Execute os módulos Active Directory e Domain Name Service (DNS) para Windows PowerShell no Active Directory Web Service

A conta `Admin` também possui direitos para executar as seguintes atividades de domínio:

- Gerenciar configurações de DNS (adicionar, remover ou atualizar registros, zonas e encaminhadores)
- Visualizar logs de eventos de DNS
- Visualizar logs de eventos de segurança

Como criar um diretório com AWS Managed Microsoft AD

1. No painel de navegação do console do Directory Service, escolha Directories (Diretórios) e escolha Set up directory (Configurar diretório).
2. Escolha AWS Managed Microsoft AD. O AWS Managed Microsoft AD é a única opção atualmente compatível para uso com o Amazon Aurora.
3. Escolha Next (Próximo).
4. Na página Enter directory information (Inserir informações do diretório), forneça as seguintes informações:

   Edição

   Escolha a edição que atenda às suas necessidades.

   Nome do DNS do diretório

   O nome completo do diretório, como **`corp.example.com`**.

Nome de NetBIOS do diretório

O nome curto opcional do diretório, como `CORP`.

Descrição do diretório

Uma descrição opcional do diretório.

Senha do Admin

A senha do administrador do diretório. O processo de criação do diretório cria uma conta de administrador com o nome de usuário `Admin` e essa senha.

A senha do administrador do diretório não pode incluir a palavra "admin". A senha diferencia letras maiúsculas de minúsculas e deve ter entre 8 e 64 caracteres. Ela também precisa conter pelo menos um caractere de três das quatro categorias a seguir:

- Letras minúsculas (a–z)
- Letras maiúsculas (A–Z)
- Números (0–9)
- Caracteres não alfanuméricos (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirmar senha

Digite a senha do administrador novamente.

> **Important**
>
> Salve essa senha. O Directory Service não armazena essa senha e não é possível recuperá-la ou redefini-la.

5. Escolha Next (Próximo).
6. Na página Choose VPC and subnets (Selecionar VPC e sub-redes), forneça as seguintes informações:

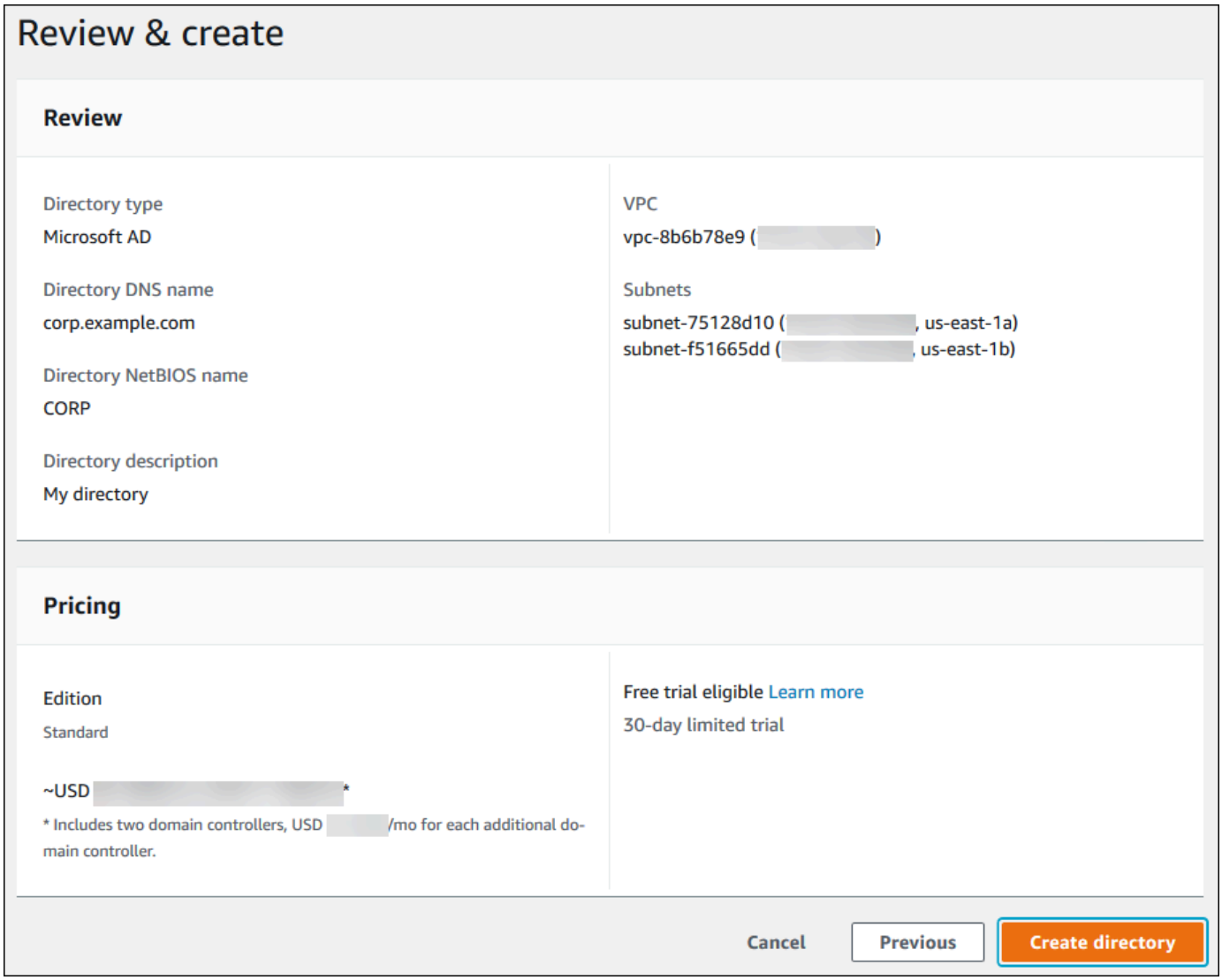
   VPC

   Escolha a VPC do diretório. É possível criar o cluster de banco de dados PostgreSQL nessa mesma VPC ou em outra VPC.

Sub-redes

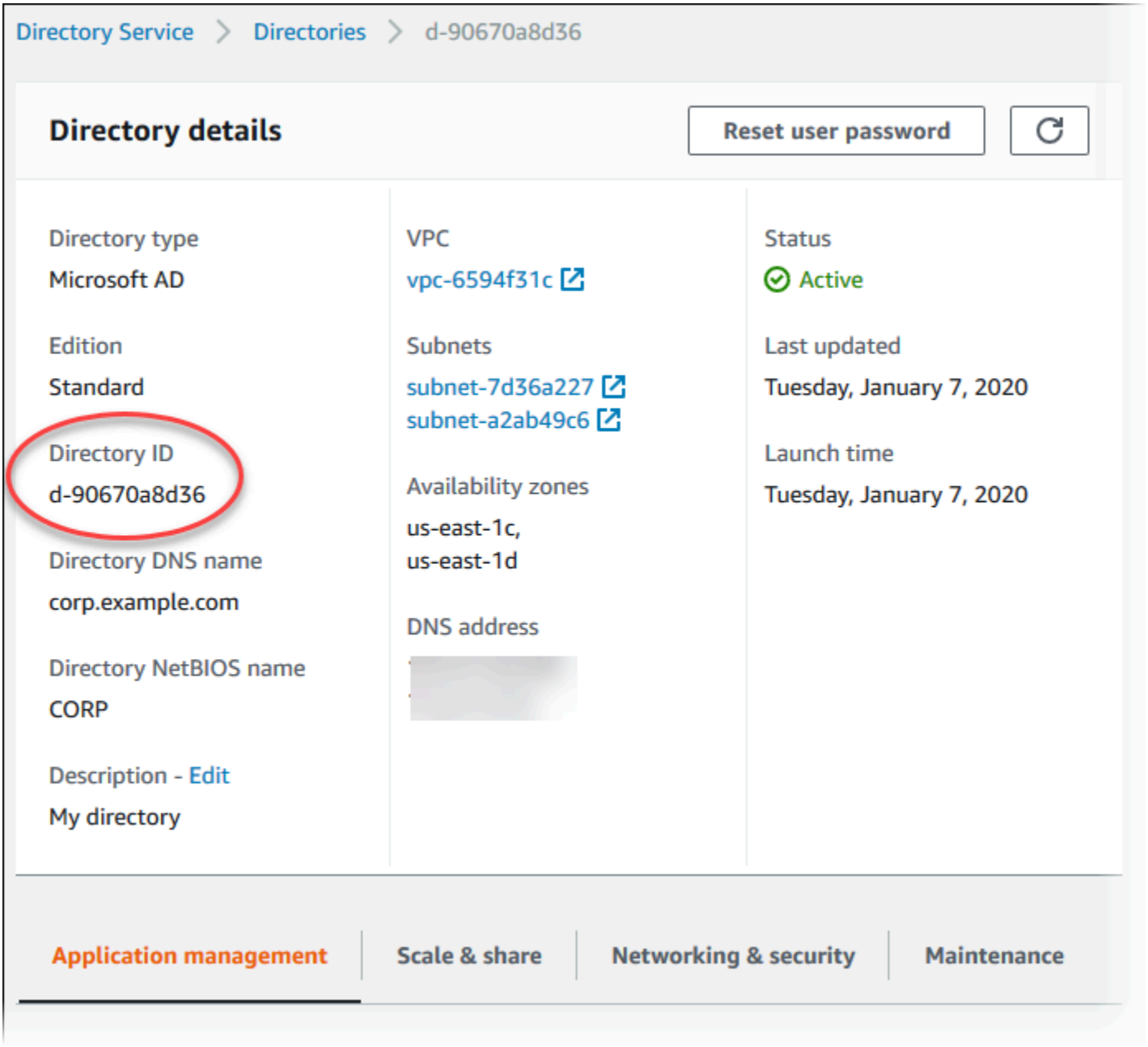
Escolha as sub-redes para os servidores do diretório. As duas sub-redes deve estar em diferentes zonas de disponibilidade.

7. Escolha Next (Próximo).
8. Analise as informações do diretório. Se alterações forem necessárias, escolha Previous (Anterior) e faça as alterações. Quando as informações estiverem corretas, escolha Create directory (Criar diretório).



A criação do diretório leva vários minutos. Depois que o diretório tiver sido criado com sucesso, o valor de Status muda para Active (Ativo).

Para visualizar informações sobre o diretório, escolha o ID do diretório na listagem de diretórios. Anote o valor do Directory ID (ID do diretório). Esse valor será necessário ao criar ou modificar a instância de banco de dados PostgreSQL.



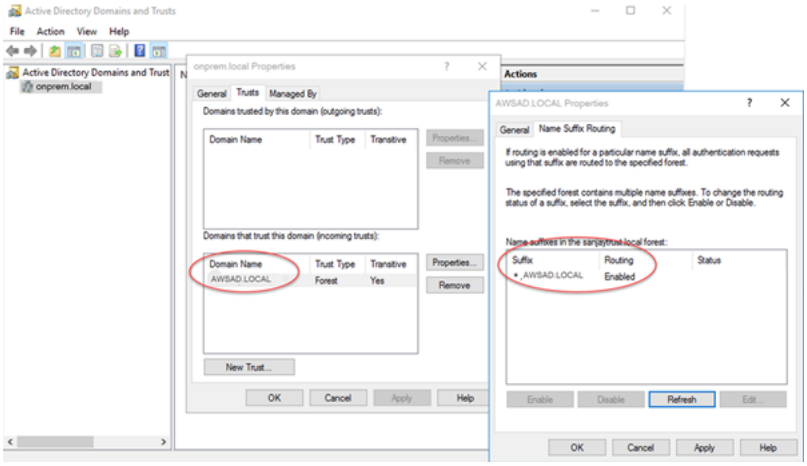
Etapa 2: (Opcional) Criar uma relação de confiança entre o Active Directory on-premises e o Directory Service

Se você não planeja usar seu próprio Microsoft Active Directory on-premises, vá para Etapa 3: Criar um perfil do IAM para que o Amazon Aurora acesse o Directory Service.

Para obter a autenticação Kerberos usando o Active Directory local, é necessário criar uma relação de domínio confiável usando uma confiança de floresta entre o Microsoft Active Directory on-premises e o diretório AWS Managed Microsoft AD (criado em Etapa 1: Criar um diretório usando o AWS Managed Microsoft AD). A relação de confiança pode ser unidirecional, onde o diretório AWS Managed Microsoft AD confia no Microsoft Active Directory on-premises. A confiança também pode ser bidirecional, onde os dois Active Directories confiam um no outro. Para ter mais informações sobre como configurar confianças usando o Directory Service, consulte Quando criar uma relação de confiança no Guia de administração do AWS Directory Service.

> **Note**
>
> Se você usa um Microsoft Active Directory on-premises:
>
> - Os clientes do Windows devem se conectar usando o nome de domínio do Directory Service no endpoint em vez de rds.amazonaws.com. Para obter mais informações, consulte Conectar-se ao PostgreSQL com a autenticação Kerberos.
> - Os clientes do Windows não podem se conectar usando endpoints personalizados do Aurora. Para saber mais, consulte Conexões de endpoints do Amazon Aurora.
> - Para bancos de dados globais:
>   - Os clientes do Windows podem se conectar usando endpoints de instância ou endpoints do cluster somente na Região da AWS primária do banco de dados global.
>   - Os clientes do Windows não podem se conectar usando endpoints do cluster em Regiões da AWS secundárias.

Verifique se o nome de domínio do Microsoft Active Directory on-premises inclui um roteamento de sufixo DNS que corresponde à relação de confiança recém-criada. A captura de tela a seguir mostra um exemplo.



Etapa 3: Criar um perfil do IAM para que o Amazon Aurora acesse o Directory Service

Para que o Amazon Aurora chame o Directory Service para você, sua conta da AWS precisa de um perfil do IAM que use a política gerenciada `AmazonRDSDirectoryServiceAccess` do IAM. Essa função permite que o Amazon Aurora faça chamadas para o Directory Service. (Observe que esse perfil do IAM para acessar o Directory Service é diferente do perfil do IAM usado para Autenticação do banco de dados do IAM).

Quando você cria uma instância de banco de dados usando o Console de gerenciamento da AWS e a conta do usuário do console tiver a permissão `iam:CreateRole`, o console criará o perfil do IAM necessário automaticamente. Nesse caso, o nome da função é `rds-directoryservice-kerberos-access-role`. Caso contrário, é necessário criar a função do IAM manualmente. Ao criar essa função do IAM, escolha `Directory Service` e associe a AWS política gerenciada da `AmazonRDSDirectoryServiceAccess` a ela.

Para ter mais informações sobre como criar funções do IAM para um serviço, consulte o tópico sobre como Criar uma função para delegar permissões a um serviço da AWS, no Guia do usuário do IAM.

> **Note**
>
> O perfil do IAM usado para a autenticação do Windows para RDS para Microsoft SQL Server não pode ser usado para o Amazon Aurora.

Como alternativa ao uso da política gerenciada pelo `AmazonRDSDirectoryServiceAccess`, você pode criar políticas com as permissões exigidas. Nesse caso, o perfil do IAM deve ter a política de confiança do IAM a seguir.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

A função também deve ter a política de perfil do IAM a seguir.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Para as Regiões da AWS pelas quais você optou, use entidades principais de serviço específicas da região nas políticas de confiança de perfil do IAM. Ao criar uma política de confiança para serviços nessas regiões, especifique o código da região na entidade principal do serviço.

O seguinte exemplo mostra uma política de confiança que inclui entidades principais de serviço específicas da região:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Substitua REGION-CODE pelo código da sua região específica. Por exemplo, use as seguintes entidades principais de serviço para a região Ásia-Pacífico (Melbourne):

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

Etapa 4: Criar e configurar usuários

Você pode criar usuários com a ferramenta Usuários e computadores do Active Directory. Essa é uma das ferramentas do Active Directory Domain Services e do Active Directory Lightweight

Directory Services. Para obter mais informações, consulte Add Users and Computers to the Active Directory domain (Adicionar usuários e computadores ao domínio do Active Directory) na documentação da Microsoft. Nesse caso, os usuários são indivíduos ou outras entidades, como seus computadores que fazem parte do domínio e cujas identidades estão sendo mantidas no diretório.

Para criar usuários em um diretório do Directory Service, é necessário estar conectado a uma instância do Amazon EC2 baseada no Windows que seja membro do diretório do Directory Service. Ao mesmo tempo, é necessário estar conectado como um usuário que tenha privilégios para criar usuários. Para obter mais informações, consulte Criar um usuário no Guia de administração do AWS Directory Service.

Etapa 5: Ativar o tráfego entre VPCs entre o diretório e a instância de banco de dados

Se você planeja localizar o diretório e o cluster de banco de dados na mesma VPC, ignore esta etapa e prossiga para Etapa 6: Criar ou modificar um cluster de banco de dados PostgreSQL.

Se você planejar localizar o diretório e a instância de Bancos de Dados em VPCs diferentes, configure o tráfego entre VPCs usando o emparelhamento de VPCs ou o AWS Transit Gateway.

O procedimento a seguir habilita o tráfego entre VPCs usando o emparelhamento de VPCs. Siga as instruções em O que é o emparelhamento de VPCs? no Guia de emparelhamento do Amazon Virtual Private Cloud.

Como habilitar o tráfego entre VPCs usando o emparelhamento de VPCs

1. Configure regras apropriadas de roteamento de VPC para garantir que o tráfego de rede possa fluir em ambos os sentidos.
2. Certifique-se de que o grupo de segurança da instância de banco de dados possa receber o tráfego de entrada do grupo de segurança do diretório.
3. Garanta que não há nenhuma regra na lista de controle de acesso (ACL) de rede para bloquear o tráfego.

Se uma conta diferente da AWS for proprietária do diretório, é necessário compartilhá-lo.

Como compartilhar o diretório entre contas da AWS.

1. Inicie o compartilhamento do diretório com a conta da AWS na qual a instância de Bancos de Dados será criada seguindo as instruções em Tutorial: Compartilhar o diretório da AWS

Managed Microsoft AD para ingresso perfeito no domínio do EC2 no Guia de administração do Directory Service.

2. Faça login no console do Directory Service usando a conta para a instância de banco de dados e garanta que o domínio tenha o status `SHARED` antes de prosseguir.
3. Enquanto estiver conectado ao console do Directory Service usando a conta da instância de banco de dados, observe o valor do Directory ID (ID do diretório). Use esse ID do diretório para associar a instância de banco de dados ao domínio.

Etapa 6: Criar ou modificar um cluster de banco de dados PostgreSQL

Crie ou modifique um cluster de banco de dados PostgreSQL para usar com seu diretório. É possível usar o console, a CLI ou a API do RDS para associar um cluster de banco de dados a um diretório. Você pode fazer isso por meio de uma das seguintes maneiras:

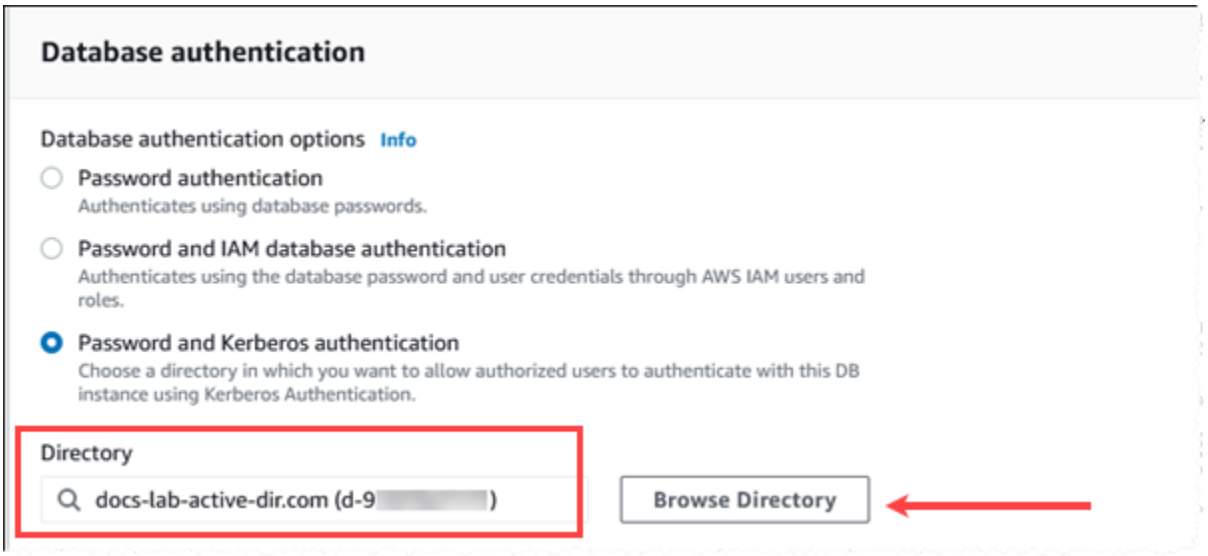
- Crie um cluster de banco de dados PostgreSQL usando o console, o comando da CLI create-db-cluster ou a operação API do RDS CreateDBCluster. Para obter instruções, consulte Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele.
- Modifique um cluster de banco de dados PostgreSQL existente usando o console, o comando da CLI modify-db-cluster ou a operação de API do RDS ModifyDBCluster. Para obter instruções, consulte Modificar um cluster de bancos de dados Amazon Aurora.
- Restaure um cluster de banco de dados PostgreSQL de um snapshot de banco de dados usando o console, o comando da CLI restore-db-cluster-from-db-snapshot ou a operação de API do RDS RestoreDBClusterFromDBSnapshot. Para obter instruções, consulte Restauração de um snapshot de um cluster de banco de dados.
- Restaure um cluster de banco de dados PostgreSQL para um momento específico usando o console, o comando da CLI restore-db-instance-to-point-in-time ou a operação de API do RDS RestoreDBClusterToPointInTime. Para obter instruções, consulte Restaurar um cluster de banco de dados para um horário especificado.

A autenticação Kerberos só é compatível com de clusters de banco de dados PostgreSQL em uma VPC. O cluster de banco de dados pode estar na mesma VPC do diretório ou em uma VPC diferente. O cluster de banco de dados deve usar um grupo de segurança que permita entrada e saída na VPC do diretório para que o cluster de banco de dados possa se comunicar com o diretório.

> **Note**
>
> No momento, não é permitido habilitar a autenticação Kerberos no cluster de banco de dados do Aurora PostgreSQL durante a migração do RDS para o PostgreSQL. É possível habilitar a autenticação Kerberos somente em um cluster de banco de dados do Aurora PostgreSQL autônomo.

Console

Ao usar o console para criar, modificar ou restaurar um cluster de banco de dados, escolha Kerberos authentication (Autenticação Kerberos) na seção Database authentication (Autenticação de banco de dados). Em seguida, escolha Browse Directory (Procurar diretório). Selecione o diretório ou escolha Create a new directory (Criar um novo diretório) para usar o Directory Service.



AWS CLI

Quando você usa a AWS CLI, são necessários os seguintes parâmetros para que o cluster de banco de dados possa usar o diretório criado:

- Para o parâmetro `--domain`, use o identificador de domínio (identificador "d-*") gerado quando o diretório foi criado.
- Para o parâmetro `--domain-iam-role-name`, use a função criada que usa a política gerenciada `AmazonRDSDirectoryServiceAccess` do IAM.

Por exemplo, o comando da CLI a seguir modifica um cluster de banco de dados para usar um diretório.

```
aws rds  modify-db-cluster --db-cluster-identifier mydbinstance --domain d-Directory-ID
 --domain-iam-role-name role-name
```

> **Important**
>
> Se você modificar um cluster de banco de dados para habilitar a autenticação Kerberos, reinicialize o cluster de banco de dados após a alteração.

Etapa 7: Criar usuários do PostgreSQL para suas entidades principais do Kerberos

Nesse ponto, seu cluster de banco de dados do Aurora PostgreSQL é unido ao domínio AWS Managed Microsoft AD. Os usuários que você criou no diretório em Etapa 4: Criar e configurar usuários precisam ser configurados como usuários do banco de dados do PostgreSQL e receber privilégios para fazer login no banco de dados. Você faz isso fazendo login como usuário do banco de dados com privilégios `rds_superuser`. Por exemplo, se você aceitou os padrões ao criar seu cluster de banco de dados do Aurora PostgreSQL use `postgres`, conforme mostrado nas etapas a seguir.

Como criar usuários de banco de dados do PostgreSQL para entidades principais do Kerberos

1. Use o `psql` para conectar-se ao endpoint da instância de banco de dados de seu cluster de banco de dados do Aurora PostgreSQL usando `psql`. O exemplo a seguir usa a conta `postgres` padrão para a função `rds_superuser`.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Crie um nome de usuário do banco de dados para cada entidade principal do Kerberos (nome de usuário do Active Directory) que você deseja que tenha acesso ao banco de dados. Use o nome de usuário canônico (identidade) conforme definido na instância do Active Directory, ou seja, uma letra minúscula `alias` (nome de usuário no Active Directory) e o nome em maiúscula do domínio do Active Directory para esse nome de usuário. O nome de usuário do Active Directory é um usuário autenticado externamente, portanto, use aspas ao redor do nome conforme mostrado a seguir.

```
postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
CREATE ROLE
```

3. Conceda a função `rds_ad` ao usuário do banco de dados.

```
postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
GRANT ROLE
```

Depois de concluir a criação de todos os usuários do PostgreSQL para suas identidades de usuário do Active Directory, os usuários podem acessar o cluster de banco de dados do Aurora PostgreSQL usando suas credenciais do Kerberos.

É necessário que os usuários do banco de dados que se autenticam usando o Kerberos estejam fazendo isso nas máquinas cliente que sejam membros do domínio do Active Directory.

Os usuários do banco de dados aos quais foi concedida a função `rds_ad` não podem ter também a função `rds_iam`. Isso também se aplica a assinaturas aninhadas. Para obter mais informações, consulte Autenticação do banco de dados do IAM.

Configurar seu cluster de banco de dados do Aurora PostgreSQL para nomes de usuário que não fazem distinção de maiúsculas e minúsculas

As versões 14.5, 13.8, 12.12 e 11.17 do Aurora PostgreSQL são compatíveis com o parâmetro `krb_caseins_users` do PostgreSQL. Esse parâmetro é compatível com nomes de usuário do Active Directory que não diferenciam maiúsculas de minúsculas. Por padrão, esse parâmetro é definido como false, então os nomes de usuário são interpretados com distinção entre maiúsculas e minúsculas pelo Aurora PostgreSQL. Esse é o comportamento padrão em todas as versões mais antigas do Aurora PostgreSQL. No entanto, você pode definir esse parâmetro como `true` em seu grupo de parâmetros de cluster de banco de dados personalizado e permitir que seu cluster de banco de dados do Aurora PostgreSQL interprete nomes de usuário, sem diferenciar maiúsculas de minúsculas. Considere fazer isso como uma conveniência para os usuários do seu banco de dados que, às vezes, podem digitar incorretamente a letra maiúscula do nome de usuário ao se autenticar usando o Active Directory.

Para alterar o parâmetro `krb_caseins_users`, seu cluster de banco de dados do Aurora PostgreSQL deve estar usando um grupo de parâmetros de cluster de banco de dados personalizado. Para obter informações sobre como trabalhar com um grupo de parâmetros de cluster de banco de dados personalizado, consulte Grupos de parâmetros para Amazon Aurora.

Você pode usar a AWS CLI ou o Console de gerenciamento da AWS para alterar a configuração. Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Etapa 8: Configurar um cliente PostgreSQL

Para configurar um cliente PostgreSQL, execute as seguintes etapas:

- Crie um arquivo krb5.conf (ou equivalente) para apontar para o domínio.
- Verifique se o tráfego pode fluir entre o host do cliente e o Directory Service. Use um utilitário de rede, como o Netcat, para o seguinte:
  - Verifique o tráfego pelo DNS da porta 53.
  - Verifique o tráfego pelo TCP/UDP da porta 53 e do Kerberos, que inclui as portas 88 e 464 do Directory Service.
- Verifique se o tráfego pode fluir entre o host do cliente e a instância de banco de dados pela porta do banco de dados. Por exemplo, use psql para conectar e acessar o banco de dados.

Veja a seguir um exemplo de conteúdo krb5.conf para o AWS Managed Microsoft AD.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Veja a seguir um exemplo de conteúdo krb5.conf para o Microsoft Active Directory on-premises.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
```

```
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Gerenciar um cluster de banco de dados do Aurora PostgreSQL em um domínio do Active Directory

É possível usar o console, a CLI ou a API do RDS para gerenciar o cluster de banco de dados e suas relações com o Microsoft Active Directory. Por exemplo, é possível associar um Microsoft Active Directory para habilitar a autenticação Kerberos. Também é possível remover a associação de um Microsoft Active Directory para desabilitar a autenticação Kerberos. Também é possível mover um cluster de banco de dados para a autenticação externa por um Microsoft Active Directory para outro.

Por exemplo, usando a CLI, é possível fazer o seguinte:

- Para tentar habilitar a autenticação Kerberos novamente para uma assinatura com falha, use o comando da CLI modify-db-cluster. Especifique o ID do diretório da associação atual para a opção `--domain`.
- Para desabilitar a autenticação Kerberos em uma instância de banco de dados, use o comando da CLI modify-db-cluster. Especifique `none` para a opção `--domain`.
- Para mover uma instância de banco de dados de um domínio para outro, use o comando da CLI modify-db-cluster. Especifique o identificador de domínio do novo domínio para a opção `--domain`.

Compreensão da associação de domínio

Depois de criar ou modificar um cluster de banco de dados, as instâncias de banco de dados se tornam membros do domínio. É possível visualizar o status da associação do domínio no console ou executando o comando da CLI describe-db-instances. O status da instância de banco de dados pode ser um dos seguintes:

- `kerberos-enabled` – a instância de banco de dados que tem a autenticação Kerberos habilitada.
- `enabling-kerberos`: a AWS está no processo de habilitar a autenticação Kerberos nessa instância de bancos de dados.
- `pending-enable-kerberos` – a habilitação da autenticação Kerberos está pendente nessa instância de banco de dados.
- `pending-maintenance-enable-kerberos`: a AWS tentará habilitar a autenticação Kerberos na instância de bancos de dados durante a próxima janela de manutenção programada.
- `pending-disable-kerberos` – a desabilitação da autenticação Kerberos está pendente nessa instância de banco de dados.
- `pending-maintenance-disable-kerberos`: a AWS tentará desabilitar a autenticação Kerberos na instância de banco de dados durante a próxima janela de manutenção programada.
- `enable-kerberos-failed`: um problema de configuração impediu que a AWS habilitasse a autenticação Kerberos na instância de banco de dados. Corrija o problema de configuração antes de emitir o comando novamente para modificar a instância de banco de dados.
- `disabling-kerberos`: a AWS está no processo de desabilitar a autenticação Kerberos nessa instância de bancos de dados.

Uma solicitação para habilitar a autenticação Kerberos pode falhar por conta de um novo problema de conectividade de rede ou de um perfil do IAM incorreto. Em alguns casos, poderá haver falha na tentativa de habilitar a autenticação Kerberos quando você criar ou modificar um cluster de banco de dados. Nesse caso, verifique se você está usando o perfil do IAM correto e modifique o cluster de banco de dados para ingressar no domínio.

Conectar-se ao PostgreSQL com a autenticação Kerberos

Você pode se conectar ao PostgreSQL com autenticação Kerberos com a interface pgAdmin ou com uma interface de linha de comando, como psql. Para obter mais informações sobre a conexão, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL. Para saber como obter o endpoint, o número da porta e outros detalhes necessários para a conexão, consulte Visualizar os endpoints de um cluster do Aurora.

> **Note**
>
> A autenticação e a criptografia GSSAPI no PostgreSQL são implementadas pela biblioteca Kerberos `libkrb5.so`. Recursos como `postgres_fdw` e `dblink` também dependem dessa mesma biblioteca para conexões de saída com autenticação ou criptografia Kerberos.

pgAdmin

Para usar o pgAdmin para conectar-se ao PostgreSQL com autenticação Kerberos, execute as seguintes etapas:

1. Inicie a aplicação pgAdmin no computador cliente.
2. Na guia Dashboard (Painel), escolha Add New Server (Adicionar novo servidor).
3. Na caixa de diálogo Criar - servidor, insira um nome na guia Geral para identificar o servidor no pgAdmin.
4. Na guia Connection (Conexão), insira as seguintes informações do banco de dados do Aurora PostgreSQL .
   - Em Host, insira o endpoint da instância gravadora do seu cluster de banco de dados do Aurora PostgreSQL. Um endpoint é semelhante ao seguinte:

     ```
     AUR-cluster-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Para se conectar a um Microsoft Active Directory on-premises de um cliente Windows, use o nome de domínio do AWS Managed Active Directory em vez de `rds.amazonaws.com` no endpoint do host. Por exemplo, suponha que o nome de domínio do Managed Active Directory da AWS seja `corp.example.com`. Depois, em Host, o endpoint seria especificado da seguinte forma:

     ```
     AUR-cluster-instance.111122223333.aws-region.corp.example.com
     ```

   - Em Porta, insira a porta designada.
   - Em Banco de dados de manutenção, insira o nome do banco de dados inicial ao qual o cliente se conectará.
   - Em Nome de usuário, insira o nome de usuário que você inseriu para a autenticação Kerberos em Etapa 7: Criar usuários do PostgreSQL para suas entidades principais do Kerberos.

5. Escolha Salvar.

Psql

Para usar o psql para conectar-se ao PostgreSQL com autenticação Kerberos, execute as seguintes etapas:

1. Em um prompt de comando, execute o comando a seguir.

   ```
   kinit username
   ```

   Substitua *username* pelo nome de usuário. No prompt, insira a senha armazenada no Microsoft Active Directory para o usuário.

2. Se o cluster de banco de dados PostgreSQL estiver usando uma VPC acessível publicamente, coloque um endereço IP para o endpoint do cluster da de banco de dados em seu arquivo `/etc/hosts` no cliente do EC2. Por exemplo, os comandos a seguir obtêm o endereço IP e o colocam no arquivo `/etc/hosts`.

   ```
   % dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
   hosts
   ```

   Se você estiver usando um Microsoft Active Directory on-premises de um cliente Windows, será necessário se conectar usando um endpoint especializado. Em vez de usar o domínio `rds.amazonaws.com` da Amazon no endpoint do host, use o nome de domínio do Managed Active Directory da AWS.

   Por exemplo, suponha que o nome de domínio do Managed Active Directory da AWS seja `corp.example.com`. Use o formato *PostgreSQL-endpoint*`.`*AWS-Region*`.`*corp.example.com* para o endpoint e coloque-o no arquivo `/etc/hosts`.

   ```
   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
   hosts
   ```

3. Use o comando psql a seguir para fazer login em um cluster de banco de dados PostgreSQL com integração ao Active Directory. Use um endpoint de cluster ou de instância.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

Para fazer login no cluster de banco de dados PostgreSQL de um cliente Windows usando um Active Directory on-premises, use o seguinte comando psql com o nome de domínio da etapa anterior (`corp.example.com`):

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

Usar grupos de segurança do AD para controle de acesso do Aurora PostgreSQL

Nas versões 14.10 e 15.5 do Aurora PostgreSQL, o controle de acesso do Aurora PostgreSQL pode ser gerenciado utilizando grupos de segurança do AWS Directory Service for Microsoft Active Directory (AD). As versões anteriores do Aurora PostgreSQL são compatíveis com a autenticação baseada em Kerberos com AD somente para usuários individuais. Cada usuário do AD precisava ser provisionado explicitamente no cluster de banco de dados para obter acesso.

Em vez de provisionar explicitamente cada usuário do AD no cluster de banco de dados com base nas necessidades de negócios, é possível utilizar os grupos de segurança do AD, conforme explicado abaixo:

- Os usuários do AD são membros de vários grupos de segurança do AD em um Active Directory. Eles não são ditados pelo administrador do cluster de banco de dados, mas são baseados nos requisitos de negócios e são gerenciados por um administrador do AD.
- Os administradores de clusters de banco de dados criam perfis de banco de dados em instâncias de banco de dados com base nos requisitos de negócios. Esses perfis de banco de dados podem ter permissões ou privilégios diferentes.
- Os administradores de clusters de banco de dados configuram um mapeamento dos grupos de segurança do AD para os perfis de banco de dados por cluster.
- Os usuários do banco de dados podem acessar clusters de banco de dados usando as credenciais do AD. O acesso é baseado na associação ao grupo de segurança do AD. Os usuários do AD ganham ou perdem o acesso automaticamente com base nas associações ao grupo do AD.

Pré-requisitos

Antes de configurar a extensão para grupos de segurança do AD, faça o seguinte:

- Configure a autenticação Kerberos para clusters de banco de dados do PostgreSQL. Para ter mais informações, consulte Configurar a autenticação Kerberos para clusters de banco de dados do PostgreSQL.

> **Note**
>
> Para grupos de segurança do AD, ignore a Etapa 7: Criar usuários do PostgreSQL para entidades principais do Kerberos neste procedimento de configuração.

> **Important**
>
> Se você habilitar grupos de segurança do AD em um cluster do Aurora PostgreSQL que já tenha a autenticação Kerberos habilitada, poderá encontrar problemas de autenticação. Isso ocorre quando você adiciona `pg_ad_mapping` ao parâmetro `shared_preload_libraries` e reinicia o banco de dados. Ao usar endpoints de cluster, as tentativas de login usando um usuário do AD que não seja um usuário do banco de dados com o perfil `rds_ad` podem falhar. Isso também pode causar falhas no mecanismo. Para resolver esse problema, desabilite e reabilite a autenticação Kerberos no cluster. Essa solução alternativa é necessária para instâncias existentes, mas não afeta as instâncias criadas após abril de 2025.

- Gerencie um cluster de banco de dados em um domínio. Para ter mais informações, consulte Gerenciar um cluster de banco de dados em um domínio .

Configurar a extensão pg_ad_mapping

O Aurora PostgreSQL agora está fornecendo uma extensão `pg_ad_mapping` para gerenciar o mapeamento entre grupos de segurança do AD e perfis de banco de dados no cluster do Aurora PostgreSQL. Para ter mais informações sobre as funções fornecidas pelo `pg_ad_mapping`, consulte Usar funções da extensão `pg_ad_mapping`.

Para configurar a extensão `pg_ad_mapping` no cluster de banco de dados do Aurora PostgreSQL, primeiro adicione `pg_ad_mapping` às bibliotecas compartilhadas no grupo de parâmetros do cluster de banco de dados personalizado para o cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações sobre como criar um grupo de parâmetros de cluster de banco de dados personalizado, consulte Grupos de parâmetros para Amazon Aurora. Depois, instale a extensão `pg_ad_mapping`. Os procedimentos nesta seção mostram o procedimento. É possível usar o Console de gerenciamento da AWS ou a AWS CLI.

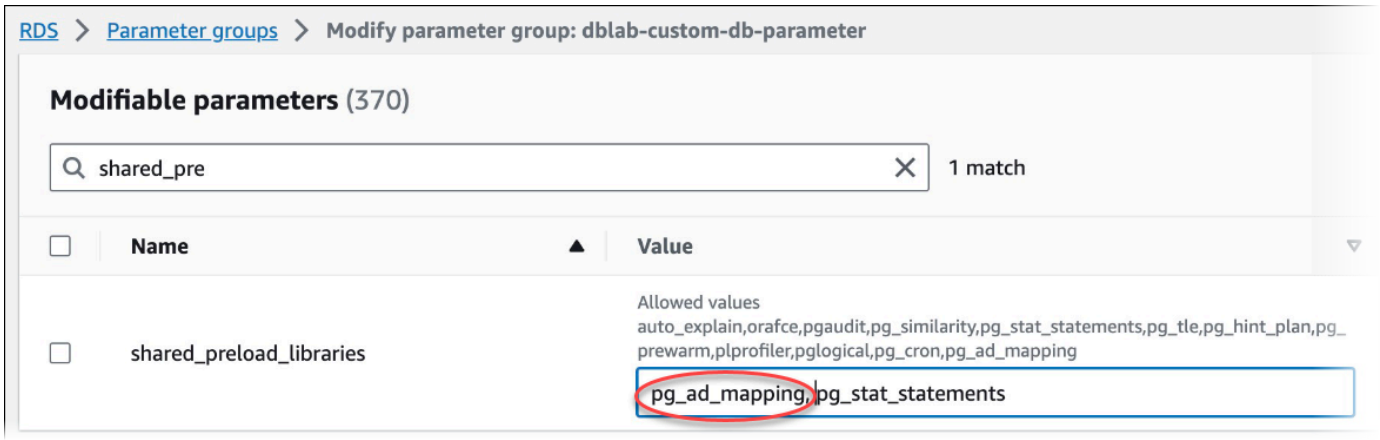
Você deve ter permissões como a função `rds_superuser` para realizar todas essas tarefas.

As etapas a seguir pressupõem que o cluster de banco de dados do Aurora PostgreSQL esteja associado a um grupo de parâmetros de cluster de banco de dados personalizado.

Console

Como configurar a extensão **`pg_ad_mapping`**

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione a instância de gravador do cluster de banco de dados do Aurora PostgreSQL.
3. Abra a guia Configuração para a instância de gravador do cluster de banco de dados do Aurora PostgreSQL. Entre os detalhes da instância, encontre o link Parameter group (Grupo de parâmetros).
4. Clique no link para abrir os parâmetros personalizados associados ao seu cluster de banco de dados do Aurora PostgreSQL.
5. No campo Parameters (Parâmetros), digite `shared_pre` para encontrar o parâmetro `shared_preload_libraries`.
6. Selecione Edit parameters (Editar parâmetros) para acessar os valores das propriedades.
7. Adicione `pg_ad_mapping` à lista no campo Values (Valores). Use uma vírgula para separar itens na lista de valores.



8. Reinicialize a instância de gravador do cluster de banco de dados do Aurora PostgreSQL para que a alteração no parâmetro `shared_preload_libraries` tenha efeito.
9. Quando a instância estiver disponível, verifique se `pg_ad_mapping` foi inicializado. Use `psql` para se conectar à instância de gravador do cluster de banco de dados do Aurora PostgreSQL e depois execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
(1 row)
```

10. Com `pg_ad_mapping` inicializado, agora você pode criar a extensão. É necessário criar a extensão depois de inicializar a biblioteca para começar a usar as funções fornecidas por essa extensão.

```
CREATE EXTENSION pg_ad_mapping;
```

11. Feche a sessão `psql`.

```
labdb=> \q
```

AWS CLI

Como configurar o pg_ad_mapping

Para configurar pg_ad_mapping usando a AWS CLI, chame a operação modify-db-parameter-group para adicionar esse parâmetro ao grupo de parâmetros personalizado, conforme mostrado no procedimento a seguir.

1. Use o comando AWS CLI a seguir para adicionar `pg_ad_mapping` ao parâmetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_ad_mapping,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Use o comando AWS CLI a seguir para reinicializar a instância de gravador do cluster de banco de dados do Aurora PostgreSQL para que o pg_ad_mapping seja inicializado.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
```

```
    --region aws-region
```

3. Quando a instância estiver disponível, verifique se a `pg_ad_mapping` foi inicializada. Use `psql` para se conectar à instância de gravador do cluster de banco de dados do Aurora PostgreSQL e depois execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
-------------------------
rdsutils,pg_ad_mapping
(1 row)
```

   Com pg_ad_mapping inicializada, agora é possível criar a extensão.

```
CREATE EXTENSION pg_ad_mapping;
```

4. Feche a sessão `psql` para que você possa usar a AWS CLI.

```
labdb=> \q
```

Recuperar o SID do grupo do Active Directory no PowerShell

Um identificador de segurança (SID) é usado para identificar de forma exclusiva uma entidade principal ou um grupo de segurança. Sempre que uma conta ou um grupo de segurança é criado no Active Directory, um SID é atribuído a ele. Para obter o SID do grupo de segurança do AD no Active Directory, é possível usar o cmdlet Get-ADGroup da máquina cliente Windows que está associada a esse domínio do Active Directory. O parâmetro Identity especifica o nome do grupo do Active Directory para obter o SID correspondente.

O exemplo a seguir exibe o SID do grupo do AD *adgroup1*.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

             SID
-----------------------------------------------
S-1-5-21-3168537779-1985441202-1799118680-1612
```

Associar o perfil de banco de dados ao grupo de segurança do AD

É necessário provisionar explicitamente os grupos de segurança do AD no banco de dados como um perfil de banco de dados do PostgreSQL. Um usuário do AD, que faz parte de pelo menos um grupo de segurança do AD provisionado, terá acesso ao banco de dados. Não conceda o `rds_ad role` ao perfil de banco de dados baseado em segurança do grupo do AD. A autenticação Kerberos para o grupo de segurança será acionada usando o sufixo do nome de domínio, como *`user1@example.com`*. Esse perfil de banco de dados não pode usar autenticação por senha ou do IAM para obter acesso ao banco de dados.

> **Note**
>
> Os usuários do AD que têm um perfil de banco de dados correspondente no banco de dados com o perfil `rds_ad` concedido a eles não podem fazer login como parte do grupo de segurança do AD. Eles obterão acesso por meio do perfil de banco de dados como usuários individuais.

Por exemplo, accounts-group é um grupo de segurança no AD em que você gostaria de provisionar esse grupo de segurança no Aurora PostgreSQL como accounts-role.

Grupo de segurança do AD	Perfil de banco de dados do PosgreSQL
accounts-group	accounts-role

Ao associar o perfil de banco de dados ao grupo de segurança do AD, é necessário garantir que o perfil de banco de dados tenha o atributo LOGIN definido e tenha o privilégio CONNECT para o banco de dados de login necessário.

```
postgres => alter role accounts-role login;

ALTER ROLE
postgres => grant connect on database accounts-db to accounts-role;
```

Agora, o administrador pode continuar criando o mapeamento entre o grupo de segurança do AD e o perfil de banco de dados do PostgreSQL.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', <SID>, <Weight>);
```

Para ter informações sobre como recuperar o SID do grupo de segurança do AD, consulte Recuperar o SID do grupo do Active Directory no PowerShell.

Pode haver casos em que um usuário do AD pertença a vários grupos; nesse caso, o usuário do AD herdará os privilégios do perfil de banco de dados, que foi provisionado com o maior peso. Se os dois perfis tiverem o mesmo peso, o usuário do AD herdará os privilégios do perfil de banco de dados correspondentes ao mapeamento que foi adicionado recentemente. A recomendação é especificar pesos que reflitam as permissões/privilégios relativos dos perfis individuais do banco de dados. Quanto maiores as permissões ou os privilégios de um perfil de banco de dados, maior o peso que deve ser associado à entrada de mapeamento. Isso evitará a ambiguidade de dois mapeamentos com o mesmo peso.

A tabela a seguir mostra um exemplo de mapeamento dos grupos de segurança do AD para perfis de banco de dados do Aurora PostgreSQL.

Grupo de segurança do AD	Perfil de banco de dados do PosgreSQL	Weight
accounts-group	accounts-role	7
sales-group	sales-role	10
dev-group	dev-role	7

No exemplo a seguir, o `user1` herdará os privilégios de sales-role, pois ele tem o maior peso, enquanto o `user2` herdará os privilégios de `dev-role`, pois o mapeamento desse perfil foi criado depois de `accounts-role`, que compartilha o mesmo peso de `accounts-role`.

Nome de usuário	Associação a grupos de segurança
user1	accounts-group sales-group
user2	accounts-group dev-group

Os comandos psql para estabelecer, listar e limpar os mapeamentos são mostrados abaixo. No momento, não é possível modificar uma única entrada de mapeamento. A entrada existente precisa ser excluída e o mapeamento recriado.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', 'S-1-5-67-890',
 7);
admin=>select pgadmap_set_mapping('sales-group', 'sales-role', 'S-1-2-34-560', 10);
admin=>select pgadmap_set_mapping('dev-group', 'dev-role', 'S-1-8-43-612', 7);

admin=>select * from pgadmap_read_mapping();

ad_sid       | pg_role        | weight | ad_grp
-------------+----------------+--------+---------------
S-1-5-67-890 | accounts-role | 7      | accounts-group
S-1-2-34-560 | sales-role    | 10     | sales-group
S-1-8-43-612 | dev-role      | 7      | dev-group
(3 rows)
```

Registro em log e auditoria de identidade de usuário do AD

Use o comando a seguir para determinar o perfil de banco de dados herdado pelo usuário atual ou da sessão:

```
postgres=>select session_user, current_user;

session_user | current_user
-------------+--------------
dev-role     | dev-role

(1 row)
```

Para determinar a identidade principal de segurança do AD, use o seguinte comando:

```
postgres=>select principal from pg_stat_gssapi where pid = pg_backend_pid();
```

```
 principal
-------------------------
 user1@example.com

(1 row)
```

No momento, a identidade do usuário do AD não está visível nos logs de auditoria. O parâmetro `log_connections` pode ser habilitado para registrar o estabelecimento da sessão de banco de dados. Para ter mais informações, consulte log_connections. A saída para isso inclui a identidade do usuário do AD, conforme mostrado abaixo. O PID de back-end associado a essa saída pode então ajudar a atribuir ações de volta ao usuário real do AD.

```
pgrole1@postgres:[615]:LOG: connection authorized: user=pgrole1
 database=postgres application_name=psql GSS (authenticated=yes, encrypted=yes,
 principal=Admin@EXAMPLE.COM)
```

Limitações

- O Microsoft Entra ID conhecido como Azure Active Directory não é aceito.

Usar funções da extensão **`pg_ad_mapping`**

A extensão `pg_ad_mapping` oferecia compatibilidade com as seguintes funções:

pgadmap_set_mapping

Essa função estabelece o mapeamento entre o grupo de segurança do AD e o perfil de banco de dados com um peso associado.

Sintaxe

```
pgadmap_set_mapping(
ad_group,
db_role,
ad_group_sid,
weight)
```

Argumentos

Parameter	Descrição
ad_group	Nome do grupo do AD. O valor não pode ser uma string nula ou vazia.
db_role	Perfil do banco de dados a ser associado ao grupo do AD especificado. O valor não pode ser uma string nula ou vazia.
ad_group_sid	Identificador de segurança usado para identificar exclusiva mente o grupo do AD. O valor começa com “S-1-” e não pode ser uma string nula ou vazia. Para obter mais informações, consulte Recuperar o SID do grupo do Active Directory no PowerShell.
weight	Peso associado ao perfil de banco de dados. O perfil com o maior peso tem precedência quando o usuário é membro de vários grupos. O valor padrão do peso é 1.

Tipo de retorno

None

Observações de uso

Essa função adiciona um novo mapeamento do grupo de segurança do AD para o perfil de banco de dados. Ela só pode ser executada na instância de banco de dados principal do cluster de banco de dados por um usuário com o privilégio de rds_superuser.

Exemplos

```
postgres=> select pgadmap_set_mapping('accounts-group','accounts-
role','S-1-2-33-12345-67890-12345-678',10);

pgadmap_set_mapping

(1 row)
```

pgadmap_read_mapping

Essa função lista os mapeamentos entre o grupo de segurança do AD e o perfil de banco de dados definidos com a função `pgadmap_set_mapping`.

Sintaxe

```
pgadmap_read_mapping()
```

Argumentos

None

Tipo de retorno

Parameter	Descrição
ad_group_sid	Identificador de segurança usado para identificar exclusiva mente o grupo do AD. O valor começa com “S-1-” e não pode ser uma string nula ou vazia. Para ter mais informações, consulte Recuperar o SID do grupo do Active Directory no PowerShell.accounts-role@example.com
db_role	Perfil do banco de dados a ser associado ao grupo do AD especificado. O valor não pode ser uma string nula ou vazia.
weight	Peso associado ao perfil de banco de dados. O perfil com o maior peso tem precedência quando o usuário é membro de vários grupos. O valor padrão do peso é 1.
ad_group	Nome do grupo do AD. O valor não pode ser uma string nula ou vazia.

Observações de uso

Chame essa função para listar todos os mapeamentos disponíveis entre o grupo de segurança do AD e o perfil de banco de dados.

Exemplos

```
postgres=> select * from pgadmap_read_mapping();

ad_sid                              | pg_role       | weight | ad_grp
------------------------------------+---------------+--------+------------------
S-1-2-33-12345-67890-12345-678      | accounts-role | 10     | accounts-group
(1 row)

(1 row)
```

pgadmap_reset_mapping

Essa função redefine um ou todos os mapeamentos definidos com a função `pgadmap_set_mapping`.

Sintaxe

```
pgadmap_reset_mapping(
ad_group_sid,
db_role,
weight)
```

Argumentos

Parameter	Descrição
ad_group_sid	Identificador de segurança usado para identificar exclusiva mente o grupo do AD.
db_role	Perfil do banco de dados a ser associado ao grupo do AD especificado.
weight	Peso associado ao perfil de banco de dados.

Se nenhum argumento for fornecido, todos os mapeamentos entre grupos do AD e perfis de banco de dados serão redefinidos. Todos os argumentos precisam ser fornecidos ou nenhum.

Tipo de retorno

None

Observações de uso

Chame essa função para excluir um mapeamento entre um grupo específico do AD e um perfil de banco de dados ou para redefinir todos os mapeamentos. Essa função só pode ser executada na instância de banco de dados principal do cluster de banco de dados por um usuário com o privilégio de `rds_superuser`.

Exemplos

```
postgres=> select * from pgadmap_read_mapping();

   ad_sid                       | pg_role      | weight      | ad_grp
--------------------------------+--------------+-------------+-------------------
 S-1-2-33-12345-67890-12345-678 | accounts-role| 10          | accounts-group
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(2 rows)
postgres=> select pgadmap_reset_mapping('S-1-2-33-12345-67890-12345-678', 'accounts-
role', 10);

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                       | pg_role      | weight      | ad_grp
--------------------------------+--------------+-------------+---------------
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(1 row)
postgres=> select pgadmap_reset_mapping();

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                       | pg_role      | weight      | ad_grp
--------------------------------+--------------+-------------+--------------
```

```
(0 rows)
```

Migrar dados para o Amazon Aurora com compatibilidade com o PostgreSQL

Você tem várias opções para migrar dados do seu banco de dados existente para um cluster de bancos de dados Amazon Aurora Edição compatível com PostgreSQL. Suas opções de migração também dependem do banco de dados do qual você está migrando e do tamanho dos dados que você está migrando. Veja a seguir suas opções:

Migrar uma instância de banco de dados do RDS for PostgreSQL usando um snapshot

É possível migrar dados diretamente de um snapshot de banco de dados do RDS for PostgreSQL para um cluster de bancos de dados Aurora PostgreSQL.

Como migrar uma instância de banco de dados do RDS for PostgreSQL usando uma réplica de leitura do Aurora

Também é possível migrar de uma instância de banco de dados do RDS for PostgreSQL criando uma réplica de leitura do Aurora PostgreSQL de uma instância de banco de dados do RDS for PostgreSQL. Quando o atraso da réplica entre a instância de banco de dados do RDS for PostgreSQL e a réplica de leitura do Aurora PostgreSQL for zero, você poderá interromper a replicação. Nesse ponto, é possível transformar a réplica de leitura do Aurora em um cluster de banco de dados independente do Aurora PostgreSQL para leitura e gravação.

Como importar dados do Amazon S3 para o Aurora PostgreSQL

É possível migrar dados importando-os de Amazon S3 para uma tabela pertencente a um cluster de bancos de dados Aurora PostgreSQL.

Como migrar de um banco de dados incompatível com o PostgreSQL

Você pode usar o AWS Database Migration Service (AWS DMS) para migrar dados de um banco de dados incompatível com PostgreSQL. Para ter mais informações sobre o AWS DMS, consulte O que é o AWS Database Migration Service? no Guia do usuário do AWS Database Migration Service.

> **Note**
>
> No momento, não é permitido habilitar a autenticação Kerberos no cluster de banco de dados do Aurora PostgreSQL durante a migração do RDS para o PostgreSQL. É possível habilitar a autenticação Kerberos somente em um cluster de banco de dados do Aurora PostgreSQL autônomo.

Para obter uma lista de Regiões da AWS onde o Aurora está disponível, consulte Amazon Aurora na Referência geral da AWS.

> **Important**
>
> Se você planeja migrar uma instância de banco de dados do RDS for PostgreSQL para um cluster de Aurora PostgreSQL banco de dados no futuro próximo, recomendamos que você desative os upgrades de versão secundária automática para a instância de banco de dados no início da fase de planejamento de migração. A migração para Aurora PostgreSQL pode ser adiada se a versão do RDS para PostgreSQL ainda não for compatível com o Aurora PostgreSQL.
>
> Para obter informações sobre Aurora PostgreSQL versões, consulte Versões de mecanismo para Amazon Aurora PostgreSQL.

Migrar um snapshot de uma instância de banco de dados do RDS for PostgreSQL para um cluster de bancos de dados Aurora PostgreSQL

Para criar um cluster de bancos de dados Aurora PostgreSQL, você pode migrar um snapshot de banco de dados de uma instância de banco de dados do RDS for PostgreSQL. O novo cluster de bancos de dados Aurora PostgreSQL é preenchido com os dados da instância de banco de dados do RDS for PostgreSQL original . Para obter informações sobre a criação de um snapshot de banco de dados, consulte Criar um snapshot de banco de dados.

Em alguns casos, o snapshot do banco de dados pode não estar na Região da AWS em que deseja localizar seus dados. Se esse for o caso, use o console do Amazon RDS para copiar o snapshot do banco de dados para essa Região da AWS. Para obter informações sobre a cópia de um snapshot de banco de dados, consulte Cópia de um snapshot de banco de dados.

Você pode migrar snapshots do RDS for PostgreSQL compatíveis com as versões do Aurora PostgreSQL disponíveis na determinada Região da AWS. Por exemplo, você pode migrar um snapshot de uma instância de banco de dados do RDS for PostgreSQL 11.1 para o Aurora PostgreSQL versão 11.4, 11.7, 11.8 ou 11.9 na região Oeste dos EUA (Norte da Califórnia). Você pode migrar um snapshot do RDS for PostgreSQL 10.11 para o Aurora PostgreSQL 10.11, 10.12, 10.13 e 10.14. Em outras palavras, o snapshot do RDS for PostgreSQL deve usar a mesma ou uma versão menor que o Aurora PostgreSQL.

Também é possível determinar que o seu novo cluster de bancos de dados Aurora PostgreSQL seja criptografado em repouso usando uma AWS KMS key. Essa opção está disponível somente para snapshot de banco de dadoss não criptografados.

Para migrar um snapshot do banco de dados do RDS for PostgreSQL para um cluster de bancos de dados Aurora PostgreSQL, você pode usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Ao usar o Console de gerenciamento da AWS, o console realiza as ações necessárias para criar o cluster de banco de dados e a instância primária.

Console

Para migrar um snapshot de banco de dados PostgreSQL usando o console do RDS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Snapshots.
3. Na página Snapshots, escolha o snapshot do RDS for PostgreSQL que você deseja migrar para um cluster de bancos de dados Aurora PostgreSQL.
4. Escolha Actions (Ações) e escolha Migrate snapshot (Migrar snapshot).
5. Defina os seguintes valores na página Migrate Database (Migrar banco de dados):

   - Versão do mecanismo de banco de dados: escolha uma versão do mecanismo de banco de dados que você deseja usar para a nova instância migrada.
   - Identificador da instância do banco de dados: insira um nome para o cluster de banco de dados exclusivo à sua conta na Região da AWS escolhida. Esse identificador é usado em endereços de endpoint para as instâncias no cluster de banco de dados. É possível optar por adicionar inteligência ao nome, como incluir a Região da AWS e o mecanismo de banco de dados escolhidos, por exemplo, **`aurora-cluster1`**.

     O DB instance identifier tem as seguintes restrições:

  - Deve conter de 1 a 63 caracteres alfanuméricos ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hífens consecutivos.
  - Deve ser exclusivo para todas as instâncias de banco de dados por conta da AWS, por Região da AWS.
- DB Instance Class (Classe de instância de banco de dados): escolha uma categoria de instância de banco de dados que tenha o armazenamento e a capacidade necessários para seu banco de dados, por exemplo `db.r6g.large`. Os volumes de cluster do Aurora crescem automaticamente à medida em que aumenta a quantidade de dados em seu banco de dados. Assim, só será necessário escolher uma classe de instância de banco de dados que atenda aos requisitos atuais de armazenamento. Para ter mais informações, consulte Visão geral do armazenamento do Amazon Aurora.
- Virtual private cloud (VPC) (Nuvem privada virtual):: se você tiver uma VPC existente, poderá usá-la com o seu cluster de bancos de dados Aurora PostgreSQL escolhendo o identificador de VPC, por exemplo `vpc-a464d1c1`. Para obter informações sobre a criação de uma VPC, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

  Do contrário, é possível optar por fazer com que Amazon RDS crie uma VPC para você, escolhendo Create new VPC (Criar nova VPC).
- DB subnet group (Grupo de sub-redes do banco de dados): se você tiver um grupo de sub-redes existente, poderá usá-lo com o cluster de bancos de dados do Aurora PostgreSQL escolhendo o identificador de grupo de sub-redes, por exemplo, `gs-subnet-group1`.
- Public access (Acesso público): escolha No (Não) para especificar que as instâncias no seu cluster de banco de dados só podem ser acessadas por recursos dentro da sua VPC. Escolha Yes (Sim) para especificar que as instâncias no seu cluster de banco de dados podem ser acessadas por recursos na rede pública.

> **Note**
>
> O cluster de banco de dados de produção talvez não precise estar em uma sub-rede pública, porque apenas os seus servidores de aplicativo precisarão acessá-lo. Se o cluster de banco de dados não precisa estar em uma sub-rede pública, defina Public access (Acesso público) para No (Não).

- VPC security group (Grupo de segurança da VPC): escolha um grupo de segurança da VPC para permitir o acesso ao seu banco de dados.
- Availability Zone (Zona de disponibilidade): escolha a Zona de disponibilidade para hospedar a instância primária do seu cluster de bancos de dados Aurora PostgreSQL. Para que o Amazon RDS escolha uma zona de disponibilidade para você, escolha No preference (Sem preferência).
- Database port (Porta de banco de dados): insira a porta padrão a ser usada ao conectar-se a instâncias no cluster de bancos de dados Aurora PostgreSQL. O padrão é `5432`.

  > **Note**
  >
  > Pode ser que haja um firewall corporativo que impeça o acesso a portas padrão, como a porta padrão do PostgreSQL, 5432. Nesse caso, forneça um valor de porta permitido pelo seu firewall corporativo. Lembre-se desse valor de porta mais tarde ao se conectar ao cluster de banco de dados Aurora PostgreSQL.

- Encryption (Criptografia): selecione Enable Encryption (Habilitar criptografia) para que o novo cluster de bancos de dados Aurora PostgreSQL seja criptografado em repouso. Escolha também uma chave do KMS como um valor AWS KMS key.
- Auto minor version upgrade (Atualização da versão secundária automática): escolha Enable auto minor version upgrade (Ativar atualização da versão secundária automática) para permitir que o cluster de bancos de dados Aurora PostgreSQL receba atualizações secundárias de versão de mecanismos de banco de dados PostgreSQL automaticamente quando forem disponibilizadas.

  A opção Auto minor version upgrade (Atualização da versão secundária automática) aplica-se apenas a atualizações de versões secundárias de mecanismos PostgreSQL do seu cluster de bancos de dados Aurora PostgreSQL. Ela não se aplica a patches regulares aplicados para manter a estabilidade do sistema.

6. Selecione Migrate para migrar o snapshot de banco de dados.
7. Escolha Database (Bancos de dados) para ver o novo cluster de banco de dados. Escolha o novo cluster de banco de dados para monitorar o andamento da migração. Quando a migração for concluída, o status do cluster será Available (Disponível). Na guia Connectivity & security (Conectividade e segurança) , você pode encontrar o endpoint do cluster a ser usado para se conectar à instância de gravador primário do cluster de banco de dados. Para ter mais

informações sobre a conexão com um cluster de bancos de dados Aurora PostgreSQL, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

AWS CLI

Usar a AWS CLI para migrar um snapshot do banco de dados do RDS for PostgreSQL para um Aurora PostgreSQL envolve dois comandos da AWS CLI separados. Primeiro, você usa o comando da AWS CLI `restore-db-cluster-from-snapshot` de criar um novo cluster de bancos de dados Aurora PostgreSQL. Em seguida, você usa o comando `create-db-instance` para criar a instância de banco de dados principal no novo cluster para concluir a migração. O procedimento a seguir cria um cluster de banco de dados Aurora PostgreSQL com instância de banco de dados primária que tem a mesma configuração que a instância de banco de dados usada para criar o snapshot.

Para migrar um snapshot do banco de dados do RDS for PostgreSQL para um cluster de bancos de dados Aurora PostgreSQL

1. Use o comando describe-db-snapshots para obter informações sobre o snapshot do banco de dados que você deseja migrar. Você pode especificar tanto o parâmetro `--db-instance-identifier` quanto `--db-snapshot-identifier` no comando. Se você não especificar um desses parâmetros, obterá todos os snapshots.

   ```
   aws rds describe-db-snapshots --db-instance-identifier <your-db-instance-name>
   ```

2. O comando retorna todos os detalhes de configuração para quaisquer snapshots criados a partir da instância de banco de dados especificada. Na saída, encontre o snapshot que você deseja migrar e localize o nome do recurso da Amazon (ARN). Para saber mais sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS). O ARN será semelhante à saída a seguir.

   ```
   “DBSnapshotArn": "arn:aws:rds:aws-region:111122223333:snapshot:<snapshot_name>"
   ```

   Também na saída, você pode encontrar detalhes de configuração para a instância de banco de dados do RDS for PostgreSQL, como a versão do mecanismo, o armazenamento alocado, se a instância de banco de dados está ou não criptografada e assim por diante.

3. Use o comando restore-db-cluster-from-snapshot para iniciar a migração. Especifique os seguintes parâmetros:

- `--db-cluster-identifier` – O nome que você deseja dar ao cluster de bancos de dados Aurora PostgreSQL. Este cluster de bancos de dados Aurora é o destino para a migração de snapshot do banco de dados.
- `--snapshot-identifier` – O nome do recurso da Amazon (ARN) do snapshot do banco de dados a ser migrado.
- `--engine` – Especifique `aurora-postgresql` para o mecanismo de cluster de bancos de dados Aurora.
- `--kms-key-id` – Esse parâmetro opcional permite criar um cluster de banco de dados Aurora PostgreSQL criptografado de um snapshot do banco de dados não criptografado. Ele também permite que você escolha uma chave de criptografia para o cluster de banco de dados diferente da chave usada para o snapshot do banco de dados.

> **Note**
>
> Você não pode criar um cluster de banco de dados não criptografado do Aurora PostgreSQL de um snapshot do banco de dados criptografado.

Sem o parâmetro `--kms-key-id` especificado como mostrado a seguir, o comando da AWS CLI restore-db-cluster-from-snapshot cria um cluster de bancos de dados Aurora PostgreSQL vazio que é criptografado usando a mesma chave que o snapshot do banco de dados ou não criptografado se a fonte do snapshot do banco de dados não for criptografada.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier cluster-name \
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name \
    --engine aurora-postgresql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier new_cluster ^
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name ^
```

```
    --engine aurora-postgresql
```

4. O comando retorna detalhes sobre o cluster de bancos de dados Aurora PostgreSQL que está sendo criado para a migração. É possível verificar o status do cluster de bancos de dados Aurora PostgreSQL usando o comando da AWS CLI describe-db-clusters.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-name
```

5. Quando o cluster de banco de dados ficar “disponível”, você usa o comando create-db-instance para preencher o cluster de bancos de dados Aurora PostgreSQL com a instância de banco de dados baseada em seu snapshot do banco de dados do Amazon RDS. Especifique os seguintes parâmetros:

   - `--db-cluster-identifier` – O nome do novo cluster de bancos de dados Aurora PostgreSQL que você criou na etapa anterior.
   - `--db-instance-identifier` – O nome que você deseja dar à instância de banco de dados. Essa instância torna-se o nó primário no cluster de bancos de dados Aurora PostgreSQL.
   - `----db-instance-class` – Especifique a classe de instância de banco de dados a ser usada. Escolha entre as classes de instância de banco de dados aceitas pela versão do Aurora PostgreSQL para a qual você está migrando. Para obter mais informações, consulte Tipos de classe de instância de banco de dados e Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.
   - `--engine` – Especifique `aurora-postgresql` para a instância de banco de dados.

   Você também pode criar a instância de banco de dados com uma configuração diferente da fonte do snapshot do banco de dados passando as opções apropriadas no comando da AWS CLI `create-db-instance`. Para ter mais informações, consulte o comando create-db-instance.

   Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier cluster-name \
    --db-instance-identifier --db-instance-class db.instance.class \
    --engine aurora-postgresql
```

   Para Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier cluster-name ^
    --db-instance-identifier --db-instance-class db.instance.class ^
    --engine aurora-postgresql
```

Quando o processo de migração for concluído, o cluster do Aurora PostgreSQL terá uma instância de banco de dados primária preenchida.

Migrar dados de uma instância de banco de dados do RDS for PostgreSQL para um cluster de bancos de dados Aurora PostgreSQL usando uma réplica de leitura do Aurora

Você pode usar uma instância de banco de dados do RDS for PostgreSQL como base para um novo cluster de banco de dados Aurora PostgreSQL usando uma réplica de leitura do Aurora. A opção de réplica de leitura do Aurora está disponível apenas para migração dentro da mesma Região da AWS e conta, e estará disponível somente se a região oferecer uma versão compatível do Aurora PostgreSQL para sua instância de banco de dados do RDS for PostgreSQL. Compatível significa que a versão do Aurora PostgreSQL é a mesma versão do RDS for PostgreSQL, ou que é uma versão secundária posterior na mesma família de versões principais.

Por exemplo, para usar essa técnica ao migrar uma instância de banco de dados do RDS for PostgreSQL 11.14, a região deve oferecer o Aurora PostgreSQL versão 11.14 ou uma versão secundária posterior na família PostgreSQL versão 11.

Tópicos

- Visão geral da migração de dados usando uma réplica de leitura do Aurora
- Preparação para migrar dados usando uma réplica de leitura do Aurora
- Criar uma réplica de leitura do Aurora
- Promover uma réplica de leitura do Aurora

Visão geral da migração de dados usando uma réplica de leitura do Aurora

Migrar de uma instância de banco de dados do RDS for PostgreSQL para um cluster de banco de dados Aurora PostgreSQL é um procedimento com várias etapas. Primeiro, você cria uma réplica de leitura do Aurora da instância de banco de dados do RDS for PostgreSQL de origem. Isso inicia um processo de replicação da instância de banco de dados do RDS for PostgreSQL para um cluster de banco de dados de finalidade especial conhecido como cluster de réplica. O cluster de réplica consiste exclusivamente em uma réplica de leitura do Aurora (uma instância do leitor).

> **Note**
>
> Pode levar algumas horas por tebibyte de dados para que a migração seja concluída.

Promoção de uma réplica do Aurora PostgreSQL

Depois de criar um cluster de banco de dados do Aurora PostgreSQL, siga estas etapas para promover a réplica do Aurora:

1. Pare todo a workload de gravação de banco de dados na instância de banco de dados do RDS for PostgreSQL de origem.
2. Obtenha o `WAL LSN` atual da instância de banco de dados do RDS para PostgreSQL de origem:

```
SELECT pg_current_wal_lsn();
pg_current_wal_lsn
--------------------
0/F0000318
(1 row)
```

3. No cluster de réplica do Aurora PostgreSQL, verifique se o LSN reproduzido é maior que o LSN da etapa 2:

```
SELECT pg_last_wal_replay_lsn();
pg_last_wal_replay_lsn
------------------------
0/F0000400
(1 row)
```

   Como alternativa, você pode usar a seguinte consulta na instância de banco de dados do RDS para PostgreSQL de origem:

```
SELECT restart_lsn FROM pg_replication_slots;
```

4. Promova o cluster de réplica do Aurora PostgreSQL.

   Quando a replicação é interrompida, o cluster de réplica é promovido a um cluster de banco de dados do Aurora PostgreSQL autônomo e o leitor é promovido a instância do gravador do cluster. Nesse ponto, você pode adicionar instâncias ao cluster de banco de dados do Aurora PostgreSQL para dimensioná-lo de acordo com seu caso de uso. Se você não precisar mais da instância de banco de dados do RDS para PostgreSQL original, poderá excluí-la.

Não será possível criar uma réplica de leitura do Aurora se a instância de banco de dados do RDS for PostgreSQL já tiver uma réplica de leitura do Aurora ou uma réplica de leitura entre regiões.

Preparação para migrar dados usando uma réplica de leitura do Aurora

> **Note**
>
> Ao se preparar para migrar dados para o Aurora PostgreSQL, é importante identificar e lidar adequadamente com as tabelas não registradas. Para obter mais informações, consulte Manipular tabelas não registradas em log durante a migração.

Durante o processo de migração usando a réplica de leitura do Aurora, as atualizações feitas na instância de banco de dados do RDS for PostgreSQL de origem são replicadas de forma assíncrona na réplica de leitura do Aurora do cluster de réplica. O processo usa a funcionalidade de replicação de transmissão nativa do PostgreSQL, que armazena segmentos Write-Ahead Logs (WAL – Logs de gravação antecipada) na instância de origem. Antes de iniciar esse processo de migração, certifique-se de que sua instância tenha capacidade de armazenamento suficiente verificando os valores das métricas listadas na tabela.

Métrica	Descrição
`FreeStorageSpace`	O espaço de armazenamento disponível. Unidade: bytes
`OldestReplicationSlotLag`	O tamanho do atraso dos dados de WAL na réplica que está demorando mais. Unidades: megabytes
`RDSToAuroraPostgreSQLReplicaLag`	A quantidade de tempo em segundos que um cluster de bancos de dados Aurora PostgreSQL atrasa em comparação com a instância de banco de dados do RDS de origem.
`TransactionLogsDiskUsage`	O espaço em disco usado pelos logs de transação.

Métrica	Descrição
	Unidades: megabytes

Para ter mais informações sobre o monitoramento da instância do RDS, consulte Monitoramento no Manual do usuário da Amazon RDS.

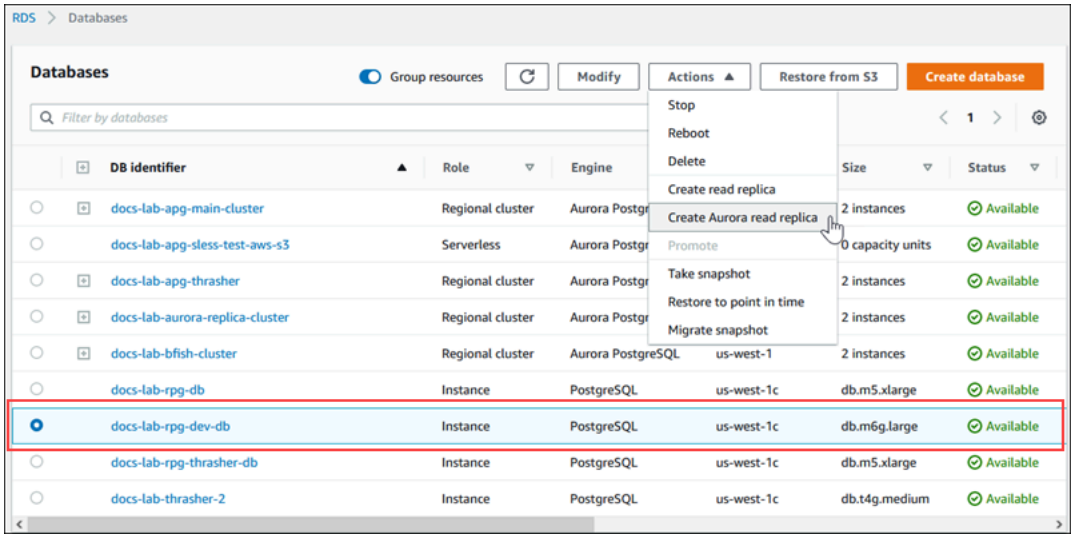
Criar uma réplica de leitura do Aurora

Você pode criar uma réplica de leitura do Aurora para uma instância de banco de dados do RDS for PostgreSQL usando o Console de gerenciamento da AWS ou a AWS CLI. A opção de criar uma réplica de leitura do Aurora usando o Console de gerenciamento da AWS estará disponível somente se a Região da AWS oferecer uma versão compatível do Aurora PostgreSQL. Ou seja, estará disponível apenas se houver uma versão do Aurora PostgreSQL idêntica à versão do RDS for PostgreSQL ou uma versão secundária posterior na mesma família de versões principais.

Console

Como criar uma réplica de leitura do Aurora a partir de uma instância de banco de dados PostgreSQL de origem

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha a instância de banco de dados do RDS for PostgreSQL que você deseja usar como a fonte da réplica de leitura do Aurora. Em Actions (Ações), selecione Create Aurora read replica (Criar réplica de leitura do Aurora). Se essa opção não for exibida, significa que uma versão compatível do Aurora PostgreSQL não está disponível na região.



4. Na página “Create Aurora read replica settings” (Criar configurações de réplica de leitura do Aurora), você configura as propriedades do cluster de banco de dados Aurora PostgreSQL conforme mostrado na tabela a seguir. O cluster de banco de dados da réplica é criado de um snapshot da instância do banco de dados de origem usando o mesmo nome de usuário e senha “primários” como a origem, portanto, não é possível alterá-los no momento.

Opção	Descrição
Classe da instância de banco de dados	Escolha uma classe de instância de banco de dados que atenda aos requisitos de processamento e memória da instância principal no cluster de banco de dados. Para ter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.
implantação Multi-AZ	Não disponível durante a migração
Identificador de instância de banco de dados	Insira o nome que você deseja dar à instância de banco de dados. Esse identificador é usado no endereço do endpoint da instância principal do novo cluster de banco de dados. O DB instance identifier tem as seguintes restrições: • Deve conter de 1 a 63 caracteres alfanuméricos ou hifens. • O primeiro caractere deve ser uma letra.

Opção	Descrição
	• Não pode terminar com um hífen ou conter dois hífens consecutivos. • Deve ser exclusivo para todas as instâncias de banco de dados para cada conta da AWS por Região da AWS.
Virtual Private Cloud (VPC)	Escolha a VPC que hospeda o cluster de banco de dados. Escolha Create new VPC (Criar nova VPC) para fazer o Amazon RDS criar uma VPC para você. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.
DB subnet group (Grupo de subredes do banco de dados	Selecione o grupo de sub-redes de banco de dados a ser usado para o cluster de banco de dados. Escolha Create new DB Subnet Group (Criar novo grupo de sub-redes de banco de dados) para fazer o Amazon RDS criar um grupo de sub-redes de banco de dados para você. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.
Public accessibility	Selecione Yes (Sim) para dar um endereço IP público ao cluster de banco de dados; senão, selecione No (Não). As instâncias em seu cluster de banco de dados podem ser uma combinação de instâncias de banco de dados públicas e privadas. Para ter mais informaçõ es sobre como ocultar instâncias do acesso público, consulte Ocultar um cluster de banco de dados em uma VPC da Internet.
Availability zone	Determine se você deseja especificar uma Zona de disponibilidade específica. Para ter mais informações sobre zonas de disponibilidade, consulte Regiões e zonas de disponibilidade.

Opção	Descrição
Grupos de segurança da VPC	Selecione um ou mais grupos de segurança da VPC para proteger o acesso à rede para o cluster de banco de dados. Escolha Create new VPC security group (Criar novo grupo de segurança da VPC) para fazer o Amazon RDS criar um grupo de segurança da VPC para você. Para ter mais informações, consulte Pré-requisitos do cluster de banco de dados.
Database port	Especifique a porta que os aplicativos e os utilitários usam para acessar o banco de dados. Os clusters de banco de dados PostgreSQL do Aurora assumem como padrão a porta PostgreSQL 5432. Em algumas empresas, os firewalls bloqueiam conexões a esta porta. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o novo cluster de banco de dados.
Grupo de parâmetros de banco de dados	Escolha um grupo de parâmetros de banco de dados para o cluster de bancos de dados Aurora PostgreSQL. O Aurora conta com um grupo de parâmetros de banco de dados padrão que você pode usar, ou é possível criar seu próprio grupo de parâmetros de banco de dados. Para ter mais informações sobre os parameter groups de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.
Grupo de parâmetros do cluster de banco de dados	Escolha um grupo de parâmetros de cluster de banco de dados para o cluster de bancos de dados Aurora PostgreSQL. O Aurora conta com um grupo de parâmetros de cluster de banco de dados que você pode usar, ou é possível criar seu próprio grupo de parâmetros. Para ter mais informações sobre os parameter groups do cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Opção	Descrição
Criptografia	Escolha Enable encryption para que o novo cluster de banco de dados Aurora seja criptografado em repouso. Se você selecionar Enable encryption (Habilitar criptografia), escolha também uma chave do KMS como o valor de AWS KMS key.
Priority	Escolha uma prioridade de failover para o cluster de banco de dados. Se você não selecionar um valor, o padrão será tier-1. Essa prioridade determina a ordem em que as réplicas do Aurora são promovidas durante a recuperação de uma falha de instância primária. Para ter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.
Backup retention period (Período de retenção de backup	Selecione o tempo, de 1 a 35 dias, para o Aurora reter cópias de backup do banco de dados. As cópias de backup podem ser usadas para restaurações point-in-time (PITR) do banco de dados, contabilizando até os segundos.
Enhanced monitoring	Escolha Enable enhanced monitoring (Habilitar monitoramento avançado) para habilitar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado. Para ter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado.
Monitoring Role (Monitoramento de perfis	Disponível apenas se você tiver escolhido Enable enhanced monitoring (Habilitar monitoramento avançado). A função do AWS Identity and Access Management (IAM) a ser usada para monitoram ento avançado. Para ter mais informações, consulte Configurar e habilitar o monitoramento avançado.

Opção	Descrição
Granularity	Disponível apenas se você tiver escolhido Enable enhanced monitoring (Habilitar monitoramento avançado). Defina o intervalo, em segundos, em que as métricas são coletadas para o seu cluster de banco de dados.
Atualização da versão secundária automática	Selecione Yes (Sim) para permitir que o cluster de bancos de dados Aurora PostgreSQL receba atualizaç ões de versões secundárias do mecanismo de banco de dados PostgreSQL automaticamente quando forem disponibilizadas. A opção Auto minor version upgrade (Atualização da versão secundária automática) aplica-se apenas a atualizações de versões secundárias de mecanismos PostgreSQL do seu cluster de bancos de dados Aurora PostgreSQL. Ela não se aplica a patches regulares aplicados para manter a estabilidade do sistema.
Janela de manutenção	Escolha o intervalo de tempo semanal durante o qual pode ocorrer a manutenção do sistema.

5. Escolha Create read replica (Criar réplica de leitura).

AWS CLI

Para criar uma réplica de leitura do Aurora a partir de uma instância de banco de dados do RDS for PostgreSQL de origem usando a AWS CLI, use primeiro o comando da CLI create-db-cluster para criar um cluster de banco de dados do Aurora PostgreSQL. Assim que houver um cluster de banco de dados, você usará o comando create-db-instance para criar a instância principal para seu cluster de banco de dados. A instância principal é a primeira instância criada em um cluster de banco de dados do Aurora. Nesse caso, ele é criado inicialmente como uma réplica de leitura do Aurora da instância de banco de dados do RDS for PostgreSQL. Quando o processo for concluído, sua instância de banco de dados do RDS for PostgreSQL terá sido efetivamente migrada para um cluster de banco de dados do Aurora PostgreSQL.

Não é necessário especificar a conta de usuário principal (normalmente, `postgres`), sua senha nem o nome do banco de dados. A réplica de leitura do Aurora obtém essas informações automaticamente da instância de banco de dados do RDS for PostgreSQL de origem identificada quando você chama os comandos da AWS CLI.

Você precisa especificar a versão do mecanismo a ser usada para o cluster de banco de dados do Aurora PostgreSQL e para a instância de banco de dados. A versão especificada deve corresponder à instância de banco de dados do RDS for PostgreSQL de origem. Se a instância de banco de dados do RDS for PostgreSQL de origem estiver criptografada, você também precisará especificar a criptografia da instância principal do cluster de banco de dados do Aurora PostgreSQL. A migração de uma instância criptografada para um cluster de banco de dados não criptografado do Aurora não é compatível.

Os exemplos a seguir criam um cluster de banco de dados do Aurora PostgreSQL chamado `my-new-aurora-cluster` que usará uma instância de origem de banco de dados do RDS não criptografada. Primeiro, você cria o cluster de banco de dados do Aurora PostgreSQL chamando o comando `create-db-cluster` da CLI. O exemplo mostra como usar o parâmetro `--storage-encrypted` opcional para especificar se o cluster de banco de dados deve ser criptografado. Como o banco de dados de origem não está criptografado, o `--kms-key-id` é usado para especificar a chave a ser usada. Para ter mais informações sobre parâmetros obrigatórios e opcionais, consulte a lista após o exemplo.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster  \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-subnet-group-name my-db-subnet
    --vpc-security-group-ids sg-11111111
    --engine aurora-postgresql \
    --engine-version same-as-your-rds-instance-version \
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-db \
    --storage-encrypted \
    --kms-key-id arn:aws:kms:aws-region:111122223333:key/11111111-2222-3333-444444444444
```

Para Windows:

```
aws rds create-db-cluster  ^
    --db-cluster-identifier my-new-aurora-cluster ^
```

```
    --db-subnet-group-name my-db-subnet ^
    --vpc-security-group-ids sg-11111111 ^
    --engine aurora-postgresql ^
    --engine-version same-as-your-rds-instance-version ^
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db ^
    --storage-encrypted ^
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Na lista a seguir, você pode encontrar mais informações sobre algumas das opções mostradas no exemplo. A menos que especificado de outra forma, esses parâmetros são obrigatórios.

- `--db-cluster-identifier`: você precisa fornecer um nome ao novo cluster de banco de dados do Aurora PostgreSQL.
- `--db-subnet-group-name`: crie seu cluster de banco de dados do Aurora PostgreSQL na mesma sub-rede de banco de dados que a instância de banco de dados de origem.
- `--vpc-security-group-ids`: especifique o grupo de segurança do cluster de banco de dados do Aurora PostgreSQL.
- `--engine-version`: especifique a versão a ser usada para o cluster de banco de dados do Aurora PostgreSQL. Deve ser o mesmo da versão usada ou uma versão inferior maior que a versão usada pela instância de banco de dados do RDS for PostgreSQL de origem.
- `--replication-source-identifier`: identifique sua instância de banco de dados do RDS for PostgreSQL usando seu nome de recurso da Amazon (ARN). Para ter mais informações sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS) na Referência geral da AWS de seu cluster de banco de dados.
- `--storage-encrypted`: opcional. Use somente quando necessário para especificar a criptografia da seguinte forma:
  - Use esse parâmetro quando a instância de banco de dados de origem tiver armazenamento criptografado. A chamada para `create-db-cluster` falhará se você não usar esse parâmetro com uma instância de banco de dados de origem que tenha armazenamento criptografado. Se você quiser usar uma chave para o cluster de banco de dados do Aurora PostgreSQL diferente da chave usada pela instância de banco de dados de origem, especifique também o `--kms-key-id`.
  - Use se o armazenamento da instância de banco de dados de origem não estiver criptografado, mas você quiser que o cluster de banco de dados do Aurora PostgreSQL use criptografia. Em

caso afirmativo, você também precisará identificar a chave de criptografia a ser usada com o parâmetro `--kms-key-id`.

- `--kms-key-id`: opcional. Quando usado, você pode especificar a chave a ser usada para criptografia de armazenamento (`--storage-encrypted`) usando o ARN da chave, o ID, o ARN do alias ou seu nome de alias. Esse parâmetro é necessário somente nas seguintes situações:
  - Para escolher uma chave para o cluster de banco de dados do Aurora PostgreSQL diferente da usada pela instância de banco de dados de origem.
  - Para criar um cluster criptografado em uma fonte não criptografada. Nesse caso, você precisa especificar a chave que o Aurora PostgreSQL deve usar para criptografia.

Depois de criar o cluster de banco de dados do Aurora PostgreSQL, crie a instância principal usando o comando `create-db-instance` da CLI, conforme mostrado a seguir:

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-instance-class db.x2g.16xlarge \
    --db-instance-identifier rpg-for-migration \
    --engine aurora-postgresql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-instance-class db.x2g.16xlarge ^
    --db-instance-identifier rpg-for-migration ^
    --engine aurora-postgresql
```

Na lista a seguir, você pode encontrar mais informações sobre algumas das opções mostradas no exemplo.

- `--db-cluster-identifier`: especifique o nome do cluster de banco de dados do Aurora PostgreSQL que você criou com o comando `create-db-instance` na etapa anterior.
- `--db-instance-class`: o nome da classe da instância de banco de dados a ser usada para sua instância principal, como `db.r4.xlarge`, `db.t4g.medium`, `db.x2g.16xlarge` etc. Para obter uma lista das classes de instância de banco de dados, consulte Tipos de classe de instância de banco de dados.

- `--db-instance-identifier`: especifique o nome para fornecer à instância principal.
- `--engine aurora-postgresql`: especifique `aurora-postgresql` para o mecanismo.

API do RDS

Para criar uma réplica de leitura do Aurora a partir de uma instância de banco de dados do RDS for PostgreSQL de origem, primeiro use a operação da API do RDS `CreateDBCluster` para criar um cluster de banco de dados do Aurora para a réplica de leitura do Aurora criada a partir da instância de banco de dados do RDS for PostgreSQL de origem. Quando o cluster de banco de dados do Aurora PostgreSQL estiver disponível, você use `CreateDBInstance` para criar a instância principal para o cluster de banco de dados do Aurora.

Não é necessário especificar a conta de usuário principal (normalmente, `postgres`), sua senha nem o nome do banco de dados. A réplica de leitura do Aurora obtém essas informações automaticamente da instância de banco de dados do RDS for PostgreSQL especificada com o `ReplicationSourceIdentifier`.

Você precisa especificar a versão do mecanismo a ser usada para o cluster de banco de dados do Aurora PostgreSQL e para a instância de banco de dados. A versão especificada deve corresponder à instância de banco de dados do RDS for PostgreSQL de origem. Se a instância de banco de dados do RDS for PostgreSQL de origem estiver criptografada, você também precisará especificar a criptografia da instância principal do cluster de banco de dados do Aurora PostgreSQL. A migração de uma instância criptografada para um cluster de banco de dados não criptografado do Aurora não é compatível.

Para criar o cluster de banco de dados do Aurora para a réplica de leitura do Aurora, use a operação da API do RDS `CreateDBCluster` com os seguintes parâmetros:

- `DBClusterIdentifier`: o nome do cluster de banco de dados a ser criado.
- `DBSubnetGroupName`: o nome do grupo de sub-redes de banco de dados a ser associado a esse cluster de banco de dados.
- `Engine=aurora-postgresql`: o nome do mecanismo a ser usado.
- `ReplicationSourceIdentifier`: o nome de recurso da Amazon (ARN) para a instância de banco de dados do PostgreSQL de origem. Para ter mais informações sobre ARNs do Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS) no Referência geral da Amazon Web Services. Se o `ReplicationSourceIdentifier` identificar uma fonte

criptografada, o Amazon RDS usará sua chave do KMS padrão, a menos que você especifique uma chave diferente com o uso da opção `KmsKeyId`.

- `VpcSecurityGroupIds`: a lista de grupos de segurança da VPC do Amazon EC2 a ser associada a esse cluster de banco de dados.
- `StorageEncrypted`: indica se o cluster de banco de dados é criptografado. Quando você usa esse parâmetro sem especificar também o `ReplicationSourceIdentifier`, o Amazon RDS usa sua chave padrão do KMS.
- `KmsKeyId`: a chave para um cluster criptografado. Quando usado, você pode especificar a chave a ser usada para criptografia de armazenamento usando o ARN da chave, o ID, o ARN do alias ou seu nome de alias.

Para ter mais informações, consulte `CreateDBCluster` na Referência de API do Amazon RDS.

Depois que o cluster de banco de dados do Aurora estiver disponível, você poderá criar uma instância principal para ele usando a operação `CreateDBInstance` da API do RDS com os seguintes parâmetros:

- `DBClusterIdentifier`: o nome de seu cluster de banco de dados.
- `DBInstanceClass`: o nome da classe de instância de banco de dados a ser usada para sua instância principal.
- `DBInstanceIdentifier`: o nome da sua instância principal.
- `Engine=aurora-postgresql`: o nome do mecanismo a ser usado.

Para ter mais informações, consulte `CreateDBInstance` na Referência de API do Amazon RDS.

Promover uma réplica de leitura do Aurora

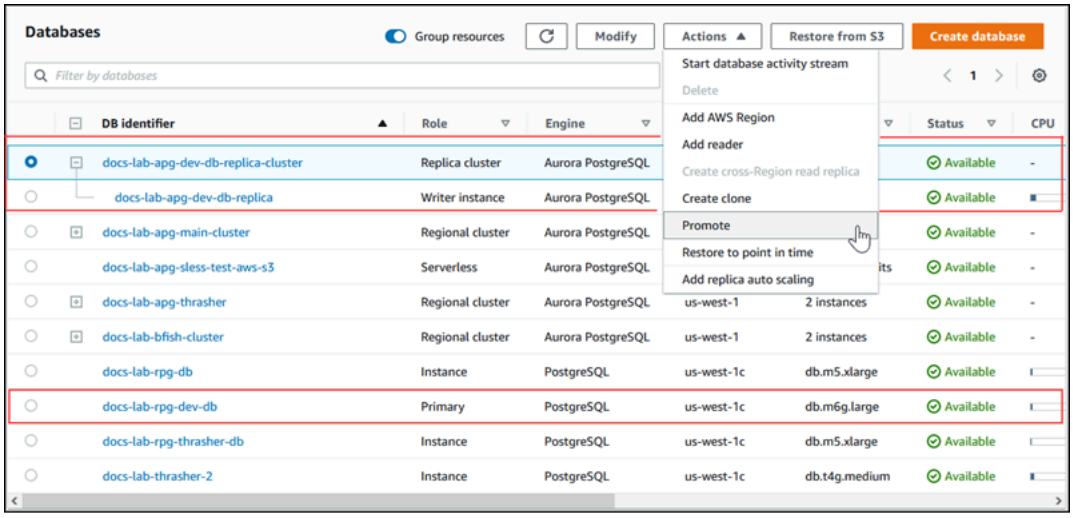
A migração para o Aurora PostgreSQL não estará concluída até que você promova o cluster de réplica, portanto, não exclua a instância de banco de dados de origem do RDS for PostgreSQL ainda.

Antes de promover o cluster de réplica, certifique-se de que a instância de banco de dados do RDS for PostgreSQL não tenha nenhuma transação em processamento ou outra atividade de gravação no banco de dados. Quando o atraso da réplica na réplica de leitura do Aurora alcança 0 (zero), você pode promover o cluster de réplica. Para ter mais informações sobre o monitoramento do atraso da réplica, consulte Monitorar a replicação do Aurora PostgreSQL e Métricas no nível da instância do Amazon Aurora.

Console

Como promover uma réplica de leitura do Aurora a um cluster de bancos de dados Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o cluster de réplica.



4. Em Actions (Ações), selecione Promote (Promover). Isso pode levar alguns minutos e pode levar a um tempo de inatividade.

Quando o processo é concluído, o cluster de réplica do Aurora é um cluster de banco de dados Aurora PostgreSQL regional, com uma instância do gravador contendo os dados da instância de banco de dados do RDS for PostgreSQL.

AWS CLI

Para promover uma réplica de leitura do Aurora a um cluster de banco de dados autônomo, use o comando promote-read-replica-db-cluster da AWS CLI.

Example

Para Linux, macOS ou Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Para Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

API do RDS

Para promover uma réplica de leitura Aurora para um cluster de banco de dados autônomo, use a operação de API do RDS PromoteReadReplicaDBCluster.

Depois de promover o cluster de réplica, você pode confirmar se a promoção foi concluída verificando o log de eventos da seguinte forma.

Para confirmar que o cluster de réplica do Aurora foi promovido

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Events.
3. Na página Events (Eventos), encontre o nome do seu cluster na lista Source (Fonte). Cada evento tem uma fonte, um tipo, um horário e uma mensagem. Você pode ver todos os eventos que ocorreram em sua Região da AWS relacionados à sua conta. Uma promoção bem-sucedida gera a mensagem a seguir.

   ```
   Promoted Read Replica cluster to a stand-alone database cluster.
   ```

Depois que promoção estiver concluída, a instância de banco de dados do RDS for PostgreSQL de origem e o cluster de banco de dados Aurora PostgreSQL serão desvinculadas. Você pode direcionar suas aplicações cliente para o endpoint da réplica de leitura do Aurora. Para ter mais informações sobre os endpoints do Aurora, consulte Conexões de endpoints do Amazon Aurora. Neste ponto, você poderá excluir com segurança a instância de banco de dados.

Otimizar a performance de consultas no Aurora PostgreSQL

Otimizar a performance das consultas é crucial porque torna a execução dos bancos de dados mais rápida e eficiente, usando menos recursos, o que melhora a experiência do usuário e reduz os custos operacionais. O Amazon Aurora PostgreSQL oferece vários recursos que ajudam a otimizar a performance das consultas para workloads do PostgreSQL.

Tópicos

- Melhorar a performance das consultas do Aurora PostgreSQL com o Aurora Optimized Reads
- Otimizar subconsultas correlacionadas no Aurora PostgreSQL
- Melhorar o desempenho de consultas usando junção adaptável
- Usar o cache de plano compartilhado

Melhorar a performance das consultas do Aurora PostgreSQL com o Aurora Optimized Reads

É possível acelerar o processamento de consultas do Aurora PostgreSQL com o Aurora Optimized Reads. Uma instância do banco de dados Aurora PostgreSQL que utiliza Leituras Otimizadas do Aurora proporciona até 8 vezes menos latência nas consultas e até 30% de economia para aplicações com grandes volumes de dados que excedem a capacidade de memória da instância do banco de dados.

Tópicos

- Visão geral do Aurora Optimized Reads no PostgreSQL
- Utilizar o Aurora Optimized Reads
- Casos de uso do Aurora Optimized Reads
- Monitorar instâncias de banco de dados que utilizam o Aurora Optimized Reads
- Práticas recomendadas para o Aurora Optimized Reads

Visão geral do Aurora Optimized Reads no PostgreSQL

O Aurora Optimized Reads está disponível por padrão quando você cria um cluster de banco de dados que use instâncias R6gd e R8gd baseadas em Graviton e R6id baseadas em Intel com armazenamento de memória expressa não volátil (NVMe). Ele está disponível nas seguintes versões do PostgreSQL:

- 14.12 e versões posteriores, 15.7 e versões posteriores, 16.3 e versões posteriores, 17.4 e versões posteriores das instâncias R8gd.
- 14.9 e versões posteriores, 15.4 e versões posteriores, 16.1 e todas as versões posteriores das instâncias R6gd e R6id.

O Aurora Optimized Reads oferece suporte a dois recursos: cache em camadas e objetos temporários.

Cache em camadas habilitado para o Aurora Optimized Reads: usando o cache em camadas, você pode estender a capacidade de cache da instância de banco de dados em até cinco vezes a memória da instância. Isso mantém automaticamente o cache para conter os dados mais recentes e transacionalmente consistentes, liberando as aplicações da sobrecarga de gerenciar a moeda de dados de soluções externas de armazenamento em cache baseadas em conjuntos de resultados. Ele oferece latência até 8 vezes melhor para consultas que anteriormente buscavam dados do armazenamento Aurora.

No Aurora, o valor para `shared_buffers` no grupo de parâmetros padrão geralmente é definido como cerca de 75% da memória disponível. No entanto, para os tipos de instância r8gd, r6gd e r6id, o Aurora reduzirá o espaço de `shared_buffers` em 4,5% para hospedar os metadados do cache do Optimized Reads.

Objetos temporários habilitados para o Optimized Reads: usando objetos temporários, você pode acelerar o processamento de consultas colocando os arquivos temporários gerados pelo PostgreSQL no armazenamento NVMe local. Isso reduz o tráfego para o Elastic Block Storage (EBS) pela rede. Ele oferece latência e taxa de throughput até duas vezes melhores para consultas avançadas que classificam, unem ou mesclam grandes volumes de dados que não se ajustam à capacidade de memória disponível em uma instância de banco de dados.

Em um cluster otimizado para E/S do Aurora, o Optimized Reads usa tanto o cache em camadas quanto os objetos temporários no armazenamento NVMe. Com o recurso de cache hierárquico habilitado para o Optimized Reads, o Aurora aloca o dobro da memória da instância para objetos temporários, aproximadamente 10% do armazenamento para operações internas e o armazenamento restante como cache em camadas. Em um cluster Aurora Standard, o Optimized Reads usa somente objetos temporários.

Os clusters do Aurora otimizados para E/S permitem redimensionar o espaço alocado a objetos temporários habilitados para leituras otimizadas usando o parâmetro dinâmico `aurora_temp_space_size` no nível da instância. O recurso de redimensionamento está disponível nas seguintes versões do PostgreSQL:

- 16.8 e todas as versões posteriores
- 15.12 e versões 15 posteriores
- 14.17 e versões 14 posteriores

Com esse parâmetro, você pode redimensionar a capacidade de memória da instância de duas até seis vezes sem precisar reinicializar o mecanismo de banco de dados. Ao expandir o espaço de objetos temporários, a alteração entra em vigor de imediato, independentemente das workloads simultâneas. Entretanto, ao reduzir o espaço, o ajuste é concluído somente depois que há espaço não utilizado suficiente em objetos temporários para atender à nova solicitação de tamanho. Depois de redimensionar objetos temporários habilitados para leituras otimizadas, o cache em camadas se ajusta automaticamente para usar qualquer espaço disponível.

Mecanismo	Configuração de armazenamento do cluster	Objetos temporários habilitados para o Optimized Reads	Cache em camadas habilitado para o Optimized Reads	Versões compatíveis
Aurora Edição Compatível com PostgreSQL	Standard	Sim	Não	• 14.9 e versões posteriores, 15.4 e versões posteriores, 16.1 e todas as versões posteriores das instâncias r6gd e r6id. • 14.12 e versões posteriores, 15.7 e versões posteriores, 16.3 e versões posteriores, 17.4 e versões posteriores das instâncias r8gd.
	Otimizado para E/S	Sim	Sim	

> **Note**
>
> Uma alternância entre clusters otimizados para E/S e Standard em uma classe de instância de banco de dados baseada em NVMe causa uma reinicialização imediata do mecanismo de banco de dados.

No Aurora PostgreSQL, use o parâmetro `temp_tablespaces` para configurar o espaço da tabela onde os objetos temporários são armazenados.

Para verificar se os objetos temporários estão configurados, use o seguinte comando:

```
postgres=> show temp_tablespaces;
temp_tablespaces
---------------------
aurora_temp_tablespace
(1 row)
```

O `aurora_temp_tablespace` é um espaço de tabela configurado pelo Aurora que aponta para o armazenamento NVMe local. Não é possível modificar esse parâmetro nem voltar para o armazenamento no Amazon EBS.

Para verificar se o cache de leitura otimizado está ativado, use o seguinte comando:

```
postgres=> show shared_preload_libraries;
                shared_preload_libraries
-----------------------------------------------------------
rdsutils,pg_stat_statements,aurora_optimized_reads_cache
```

Utilizar o Aurora Optimized Reads

Quando você provisiona uma instância de banco de dados do Aurora PostgreSQL com uma instância de banco de dados baseada em NVMe, a instância de banco de dados utiliza automaticamente o Aurora Optimized Reads.

Para ativar o Aurora Optimized Reads, execute um destes procedimentos:

- Crie um cluster de banco de dados do Aurora PostgreSQL utilizando uma das classes de instância de banco de dados NVMe. Para obter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.
- Modifique um cluster de banco de dados do Aurora PostgreSQL para utilizar uma das classes de instância de banco de dados baseada em NVMe. Para obter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.

O Aurora Optimized Reads está disponível em todas as Regiões da AWS onde há suporte para uma ou mais dessas classes de instância de banco de dados com SSD NVMe local. Para obter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Para voltar para uma instância do Aurora de leituras não otimizadas, modifique a classe de instância de banco de dados da instância do Aurora para a classe de instância semelhante sem o armazenamento efêmero do NVMe para as workloads de banco de dados. Por exemplo, se a classe de instância de banco de dados atual for db.r6gd.4xlarge, selecione db.r6g.4xlarge para voltar. Para ter mais informações, consulte Modificar uma instância de banco de dados do Aurora.

Casos de uso do Aurora Optimized Reads

Cache em camadas habilitado para o Optimized Reads

Veja a seguir alguns casos de uso que podem se beneficiar do Optimized Reads com cache em camadas:

- Aplicações de escala na Internet, como processamento de pagamentos, cobrança e comércio eletrônico com SLAs altamente rigorosos.
- Painéis de relatórios em tempo real que executam centenas de consultas pontuais para coleta de métricas/dados.
- Aplicações de IA generativa com a extensão pgvector para pesquisar vizinhos exatos ou mais próximos em milhões de incorporações vetoriais.

Objetos temporários habilitados para o Optimized Reads

Veja a seguir alguns casos de uso que podem se beneficiar do Optimized Reads com objetos temporários:

- Consultas analíticas que incluem expressões de tabela comuns (CTEs), tabelas derivadas e operações de agrupamento.

- Réplicas de leitura que lidam com as consultas não otimizadas de uma aplicação.
- Consultas de relatórios dinâmicos ou sob demanda com operações complexas, como GROUP BY e ORDER BY, que nem sempre podem usar índices apropriados.
- Operações `CREATE INDEX` ou `REINDEX` de classificação.
- Outras workloads que usam tabelas temporárias internas.

Monitorar instâncias de banco de dados que utilizam o Aurora Optimized Reads

Você pode monitorar as consultas que usam o cache em camadas habilitado para o Optimized Reads com o comando EXPLAIN, conforme mostrado no seguinte exemplo:

```
Postgres=> EXPLAIN (ANALYZE, BUFFERS) SELECT c FROM sbtest15 WHERE id=100000000


QUERY PLAN
------------------------------------------------------------------------------------------
 Index Scan using sbtest15_pkey on sbtest15  (cost=0.57..8.59 rows=1 width=121) (actual
 time=0.287..0.288 rows=1 loops=1)
   Index Cond: (id = 100000000)
   Buffers: shared hit=3 read=2 aurora_orcache_hit=2
   I/O Timings: shared/local read=0.264
 Planning:
   Buffers: shared hit=33 read=6 aurora_orcache_hit=6
   I/O Timings: shared/local read=0.607
 Planning Time: 0.929 ms
 Execution Time: 0.303 ms
(9 rows)
Time: 2.028 ms
```

> **Note**
>
> Os campos `aurora_orcache_hit` e `aurora_storage_read` na seção `Buffers` do plano de explicação são mostrados somente quando o Optimized Reads está ativado e seus valores são maiores que zero. O campo de leitura é o total dos campos `aurora_orcache_hit` e `aurora_storage_read`.

Você pode monitorar instâncias de banco de dados que usam o Aurora Optimized Reads com as seguintes métricas do CloudWatch:

- `AuroraOptimizedReadsCacheHitRatio`
- `FreeEphemeralStorage`
- `ReadIOPSEphemeralStorage`
- `ReadLatencyEphemeralStorage`
- `ReadThroughputEphemeralStorage`
- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Essas métricas fornecem dados sobre armazenamento de instâncias, IOPS e throughput. Para ter mais informações sobre essas métricas, consulte Métricas no nível da instância do Amazon Aurora.

Também é possível usar a extensão `pg_proctab` para monitorar o armazenamento NVMe.

```
postgres=>select * from pg_diskusage();

major | minor |       devname       | reads_completed | reads_merged | sectors_read |
 readtime | writes_completed | writes_merged | sectors_written | writetime | current_io
 | iotime  | totaliotime
------+-------+---------------------+-----------------+--------------+--------------
+----------+------------------+---------------+-----------------+-----------
+------------+---------+-------------
      |       | rdstemp             |           23264 |            0 |       191450 |
  11670 |          1750892 |             0 |        24540576 |    819350 |          0 |
 3847580 |      831020
      |       | rdsephemeralstorage |           23271 |            0 |       193098 |
   2620 |           114961 |             0 |        13845120 |    130770 |          0 |
  215010 |      133410
(2 rows)
```

Práticas recomendadas para o Aurora Optimized Reads

Use as práticas recomendadas a seguir para o Aurora Optimized Reads:

- Monitore o espaço de armazenamento disponível no armazenamento de instâncias com a métrica do CloudWatch `FreeEphemeralStorage`. Se o armazenamento de instância estiver atingindo

seu limite devido à workload na instância de banco de dados, ajuste a simultaneidade e as consultas que fazem uso intenso de objetos temporários ou modifique para usar uma classe de instância de banco de dados maior.

- Monitore a métrica do CloudWatch para a taxa de acertos de cache do Optimized Reads. Operações como VACUUM modificam um grande número de blocos muito rapidamente. Isso pode causar uma queda temporária na taxa de acerto. A extensão `pg_prewarm` pode ser usada para carregar dados no cache de buffer, permitindo que o Aurora grave proativamente alguns desses blocos no cache do Optimized Reads.
- Você pode ativar o gerenciamento de cache de cluster (CCM) para aquecer o cache de buffer e o cache em camadas em um leitor de nível 0, que será usado como destino de failover. Quando o CCM está ativado, o cache do buffer é escaneado periodicamente para gravar páginas elegíveis para remoção em cache em camadas. Para obter mais informações sobre CCM, consulte Recuperação rápida após failover com o gerenciamento de cache do cluster para o Aurora PostgreSQL.

Otimizar subconsultas correlacionadas no Aurora PostgreSQL

Uma subconsulta correlacionada faz referência às colunas da tabela da consulta externa. Ela é avaliada uma vez para cada linha exibida pela consulta externa. No exemplo a seguir, a subconsulta faz referência a uma coluna da tabela ot. Essa tabela não está incluída na cláusula FROM da subconsulta, mas é referenciada na cláusula FROM da consulta externa. Se a tabela ot tiver 1 milhão de linhas, a subconsulta precisará ser avaliada 1 milhão de vezes.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Note

- A transformação e o cache de subconsultas estão disponíveis no Aurora PostgreSQL a partir da versão 16.8, enquanto o Babelfish para Aurora PostgreSQL é compatível com esses recursos a partir da versão 4.2.0.
- A partir das versões 4.6.0 e 5.2.0 do Babelfish para Aurora PostgreSQL, os seguintes parâmetros controlam esses recursos:
  - babelfishpg_tsql.apg_enable_correlated_scalar_transform
  - babelfishpg_tsql.apg_enable_subquery_cache

Por padrão, ambos os parâmetros estão ativados.

Melhorar a performance de consultas do Aurora PostgreSQL usando a transformação de subconsultas

O Aurora PostgreSQL pode acelerar subconsultas correlacionadas transformando-as em junções externas equivalentes. Essa otimização se aplica a estes dois tipos de subconsultas correlacionadas:

- Subconsultas que exibem um único valor agregado e aparecem na lista SELECT.

```
SELECT ot.a, ot.b, (SELECT AVG(it.b) FROM it WHERE it.a = ot.a) FROM ot;
```

- Subconsultas que exibem um único valor agregado e aparecem em uma cláusula WHERE.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Ativar a transformação na subconsulta

Para permitir a transformação de subconsultas correlacionadas em junções externas equivalentes, defina o parâmetro `apg_enable_correlated_scalar_transform` como `ON`. O valor padrão desse parâmetro é `OFF`.

É possível modificar o grupo de parâmetros do cluster ou da instância para definir os parâmetros. Para saber mais, consulte Grupos de parâmetros para Amazon Aurora.

Como alternativa, é possível definir a configuração apenas para a sessão atual chamando o seguinte comando:

```
SET apg_enable_correlated_scalar_transform TO ON;
```

Verificar a transformação

Use o comando EXPLAIN para verificar se a subconsulta correlacionada foi transformada em uma junção externa no plano de consulta.

Quando a transformação estiver habilitada, a parte da subconsulta correlacionada aplicável será transformada em junção externa. Por exemplo:

```
postgres=> CREATE TABLE ot (a INT, b INT);
CREATE TABLE
postgres=> CREATE TABLE it (a INT, b INT);
CREATE TABLE

postgres=> SET apg_enable_correlated_scalar_transform TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);

                        QUERY PLAN
------------------------------------------------------------
 Hash Join
   Hash Cond: (ot.a = apg_scalar_subquery.scalar_output)
   Join Filter: ((ot.b)::numeric < apg_scalar_subquery.avg)
   ->  Seq Scan on ot
   ->  Hash
         ->  Subquery Scan on apg_scalar_subquery
               ->  HashAggregate
                     Group Key: it.a
                     ->  Seq Scan on it
```

A mesma consulta não é transformada quando o parâmetro GUC é `OFF`. O plano não terá junção externa, mas, sim, um subplano.

```
postgres=> SET apg_enable_correlated_scalar_transform TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);
                QUERY PLAN
---------------------------------------
 Seq Scan on ot
   Filter: ((b)::numeric < (SubPlan 1))
   SubPlan 1
     ->  Aggregate
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Limitações

- A subconsulta deve estar na lista SELECT ou em uma das condições na cláusula where. Caso contrário, ela não será transformada.

- A subconsulta deve exibir uma função agregada. Não é possível usar funções agregadas definidas pelo usuário para transformação.
- Uma subconsulta cuja expressão de retorno não seja uma função agregada simples não será transformada.
- A condição correlacionada nas cláusulas WHERE da subconsulta deve ser uma referência de coluna simples. Caso contrário, ela não será transformada.
- A condição correlacionada na subconsulta em que as cláusulas devem ser um predicado de igualdade simples.
- A subconsulta não pode conter uma cláusula HAVING nem GROUP BY.
- A cláusula WHERE na subconsulta pode conter um ou mais predicados combinados com AND.

> **Note**
>
> O impacto da transformação no desempenho varia de acordo com seu esquema, dados e workload. A execução de subconsulta correlacionada com transformação pode melhorar significativamente o desempenho à medida que o número de linhas produzidas pela consulta externa aumenta. Recomendamos que você teste esse recurso em um ambiente de não produção com seu esquema, dados e workload reais antes de habilitá-lo em um ambiente de produção.

Usar o cache de subconsultas para melhorar a performance das consultas do Aurora PostgreSQL

O Aurora PostgreSQL permite o uso de cache de subconsultas para armazenar os resultados de subconsultas correlacionadas. Esse recurso ignora execuções repetidas de subconsultas correlacionadas quando os resultados da subconsulta já estão no cache.

Noções básicas sobre cache de subconsultas

O nó Memoize do PostgreSQL é a parte principal do cache de subconsultas. Ele mantém uma tabela de hash no cache local para associar os valores dos parâmetros de entrada às linhas de resultados da consulta. O limite de memória para a tabela de hash é o produto de work_mem e hash_mem_multiplier. Para saber mais, consulte Resource Consumption.

Durante a execução da consulta, o cache de subconsultas usa a taxa de acertos de cache (CHR) para estimar se o cache está melhorando a performance das consultas e para decidir, no tempo de

execução da consulta, se deve continuar usando o cache. CHR é a razão entre o número de acertos de cache e o número total de solicitações. Por exemplo, se uma subconsulta correlacionada precisar ser executada 100 vezes e 70 desses resultados de execução puderem ser recuperados do cache, a CHR será 0,7.

Para cada número de falhas de cache de apg_subquery_cache_check_interval, o benefício do cache de subconsultas é avaliado conferindo se a CHR é maior que apg_subquery_cache_hit_rate_threshold. Caso contrário, o cache será excluído da memória e a execução da consulta retornará à reexecução da subconsulta original sem cache.

Parâmetros que controlam o comportamento do cache de subconsultas

A tabela a seguir lista os parâmetros que controlam o comportamento do cache de subconsultas.

Parameter	Descrição	Padrão	Permitido
apg_enable_subquery_cache	Permite o uso de cache para subconsultas escalares correlacionadas.	DESL.	ON, OFF
apg_subquery_cache_check_interval	Define a frequência, em número de falhas de cache, para avaliar a taxa de acertos de cache da subconsulta.	500	0–2147483647
apg_subquery_cache_hit_rate_threshold	Define o limite para a taxa de acertos de cache da subconsulta.	0.3	0, 0 a 1,0

Note

- Valores maiores de `apg_subquery_cache_check_interval` podem melhorar a precisão da estimativa dos benefícios do cache baseada em CHR, mas aumentarão a sobrecarga no cache, pois a CHR só será avaliada quando a tabela de cache tiver linhas `apg_subquery_cache_check_interval`.

- Valores maiores de `apg_subquery_cache_hit_rate_threshold` têm a tendência de abandonar o cache de subconsultas e retornar à reexecução da subconsulta original sem cache.

É possível modificar o grupo de parâmetros do cluster ou da instância para definir os parâmetros. Para saber mais, consulte Grupos de parâmetros para Amazon Aurora.

Como alternativa, é possível definir a configuração apenas para a sessão atual chamando o seguinte comando:

```
SET apg_enable_subquery_cache TO ON;
```

Ativar o cache de subconsultas no Aurora PostgreSQL

Quando o cache de subconsultas está habilitado, o Aurora PostgreSQL o aplica para salvar os resultados das subconsultas. O plano de consulta terá então um nó Memoize em SubPlan.

Por exemplo, a sequência de comandos a seguir mostra o plano de execução de consulta estimado de uma subconsulta correlacionada simples sem cache de subconsultas.

```
postgres=> SET apg_enable_subquery_cache TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Seq Scan on it
           Filter: (a = ot.a)
```

Após a ativação de `apg_enable_subquery_cache`, o plano de consulta conterá um nó Memoize sob o nó SubPlan, indicando que a subconsulta planeja usar o cache.

```
postgres=> SET apg_enable_subquery_cache TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);
```

```
             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Memoize
           Cache Key: ot.a
           Cache Mode: binary
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

O plano real de execução da consulta contém mais detalhes do cache de subconsultas, incluindo acertos de cache e ausências no cache. A saída a seguir mostra o plano real de execução da consulta do exemplo acima após a inserção de alguns valores nas tabelas.

```
postgres=> EXPLAIN (COSTS FALSE, TIMING FALSE, ANALYZE TRUE) SELECT ot.a, ot.b FROM ot
 WHERE ot.b < (SELECT it.b FROM it WHERE it.a = ot.a);
            QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on ot (actual rows=2 loops=1)
   Filter: (b < (SubPlan 1))
   Rows Removed by Filter: 8
   SubPlan 1
     ->  Memoize (actual rows=0 loops=10)
           Cache Key: ot.a
           Cache Mode: binary
           Hits: 4  Misses: 6  Evictions: 0  Overflows: 0  Memory Usage: 1kB
           ->  Seq Scan on it (actual rows=0 loops=6)
                 Filter: (a = ot.a)
                 Rows Removed by Filter: 4
```

O número total de acertos de cache é quatro e o número total de ausências no cache é seis. Se o número total de acertos e ausências for menor que o número de loops no nó Memoize, isso significa que a avaliação da CHR não foi aprovada e o cache foi limpo e abandonado em algum momento. A execução da subconsulta então retornou à reexecução original sem cache.

Limitações

O cache de subconsultas não comporta determinados padrões de subconsultas correlacionadas. Esses tipos de consulta serão executados sem cache, mesmo que o cache de subconsultas esteja ativado:

- Consultas relacionadas IN/EXISTS/ANY/ALL
- Subconsultas correlacionadas contendo funções não determinísticas.
- Subconsultas correlacionadas que fazem referência a colunas de tabelas externas com tipos de dados que não comportam operações de hash ou igualdade.

Melhorar o desempenho de consultas usando junção adaptável

Visão geral

A junção adaptável é um recurso de versão prévia no Aurora PostgreSQL 17.4 que ajuda a melhorar o desempenho das consultas. Esse recurso está desabilitado por padrão, mas você pode habilitá-lo usando os parâmetros de configuração global do usuário (GUC). Como esse é um recurso de versão prévia, os valores padrão dos parâmetros podem mudar. Quando habilitada, a junção adaptável ajuda a otimizar o desempenho da consulta ao alternar dinamicamente de uma junção de loop aninhado para uma junção de hash em tempo de execução. Essa troca ocorre quando o otimizador do PostgreSQL escolhe incorretamente uma junção de loop aninhado devido a estimativas de cardinalidade imprecisas.

Configurar junção adaptável

Você pode controlar a junção adaptável usando estes três parâmetros de GUC:

Parâmetros de configuração de junção adaptável

Parâmetro de GUC	Descrição	Opções padrão e de configuração
apg_adaptive_join_ crossover_multiplier	Esse multiplicador funciona com o ponto de cruzamento de linhas para determinar quando mudar de um loop aninhado para uma junção de hash. O ponto de cruzamento de linhas é onde o otimizador SQL estima que as operações de loop aninhado e junção de hash tenham o mesmo custo. Um valor multiplicador mais alto reduz a probabilidade de	Controla se a junção adaptável está habilitada • Valor padrão: -1 (desabilitado) • Intervalo válido: -1 a DBL_MAX • Para habilitar: defina como >= 1

Parâmetro de GUC	Descrição	Opções padrão e de configuração
	a junção adaptável mudar para uma junção de hash.	
apg_adaptive_join_cost_threshold	Esse parâmetro define um limite mínimo de custo de consulta. A junção adaptável se desabilita automaticamente para consultas abaixo desse limite. Isso evita a sobrecarga de desempenho em consultas simples, nas quais o custo de planejar uma junção adaptável pode superar os benefícios de mudar de loop aninhado para junção de hash.	Define o limite mínimo de custo para a consulta • Valor padrão: 100 • Intervalo válido: 0 a DBL_MAX

Parâmetro de GUC	Descrição	Opções padrão e de configuração

apg_enable_parameterized_adaptive_join	Quando habilitado, esse parâmetro estende a funcionalidade de junção adaptável às junções de loop aninhado parametrizadas. Por padrão, a junção adaptável funciona somente com junções de loop aninhado não parametrizadas, pois é mais provável que elas se beneficiem da mudança para a junção de hash. As junções de loop aninhado parametrizadas geralmente têm melhor desempenho, tornando a mudança para a junção de hash menos essencial.	Controla o comportamento de junção adaptável para junções de loop aninhado
		• Valor padrão: falso
		• Valores válidos: verdadeiro/falso
		  • Quando falso: funciona somente com junções de loop aninhado não parametrizadas.
		  • Quando verdadeiro: funciona com junções de loop aninhado parametrizadas e não parametrizadas.
		Note
		Requer que `apg_adaptive_join_crossover_multiplier` seja habilitado primeiro.

Usar o cache de plano compartilhado

Visão geral

O Aurora PostgreSQL usa um modelo de processo por usuário em que cada conexão de cliente cria um processo de backend dedicado. Cada processo de backend mantém seu próprio cache de plano local para instruções preparadas. Como esses caches não podem ser compartilhados entre processos, as aplicações que usam muitas instruções preparadas podem criar caches duplicados em diferentes processos de backend, o que aumenta o uso da memória.

As versões 17.6 e posteriores e 16.10 e posteriores do Aurora PostgreSQL introduzem a funcionalidade de cache de plano compartilhado. Quando você habilita esse recurso, os processos de backend podem compartilhar planos genéricos, o que reduz o uso de memória e melhora o desempenho ao eliminar a geração de planos duplicados.

O cache de plano compartilhado usa os seguintes componentes como chave de cache:

- string de consulta (incluindo comentários);
- parâmetros de GUC relacionados ao planejador (inclusive `search_path`);
- ID de usuário
- ID do banco de dados.

As reinicializações da instância redefinem o cache compartilhado.

Parâmetros

A seguinte tabela descreve os parâmetros de um que controlam o recurso de cache de plano compartilhado:

Parameter	Descrição	Padrão	Permitido
`apg_shared_plan_cache.enable`	Habilita ou desabilita o cache de plano compartilhado.	0 (DESABILITADO)	0, 1
`apg_shared_plan_cache.max`	O número máximo de entradas de cache.	200-1.000 (depende do	100-50.000

Parameter	Descrição	Padrão	Permitido
		tamanho da instância)	
`apg_shared_plan_cache.min_size_per_entry`	O tamanho mínimo do plano para armazenar no cache compartilhado. Planos menores usam cache local para otimizar o desempenho de OLTP.	16 KB	0-32.768 (KB)
`apg_shared_plan_cache.max_size_per_entry`	O tamanho máximo do plano para cache compartilhado. Planos maiores armazenam somente informações de custo.	256 KB-4 MB (depende do tamanho da instância)	0-32.768 (KB)
`apg_shared_plan_cache.idle_generic_plan_release_timeout`	O tempo após o qual as sessões ociosas lançam planos genéricos locais. Valores mais baixos economizam memória; valores mais altos podem melhorar o desempenho.	10 segundos	0-2.147.483.647 (ms)

> **Note**
>
> É possível modificar todos os parâmetros sem reiniciar.

Monitorar visualizações e funções

- `apg_shared_plan_cache()`: mostra informações detalhadas de entradas do cache (acertos, validade e carimbos de data/hora).
- `apg_shared_plan_cache_stat()`: exibe estatísticas em nível de instância (remoções e invalidações).
- `apg_shared_plan_cache_reset()`: remove todas as entradas em `apg_shared_plan_cache()` e `apg_shared_plan_cache_stat()`.

- `apg_shared_plan_cache_remove(cache_key)`: remove uma entrada de `apg_shared_plan_cache()` quando a entrada corresponde à `cache_key`.

Limitações

- Funciona somente com instruções preparadas e não armazena em cache instruções PL/pgSQL.
- Não armazena em cache consultas que contenham tabelas temporárias ou tabelas de catálogo.
- Não armazena em cache consultas que dependem de segurança em nível de linha (RLS).
- Cada réplica mantém seu próprio cache (sem compartilhamento entre réplicas).

Trabalhar com tabelas não registradas em log no Aurora PostgreSQL

O Amazon Aurora PostgreSQL comporta tabelas não registradas em log que são à prova de falhas e mantêm a integridade dos dados mesmo após falhas ou failovers de instâncias de gravador. No PostgreSQL padrão, as tabelas não registradas em log ignoram o log de gravação antecipada (WAL) durante as operações de gravação, possibilitando velocidades de gravação mais altas. Entretanto, isso ocorre à custa de uma durabilidade reduzida, pois tabelas não registradas em log não são à prova de falhas e podem perder dados após uma falha do sistema ou um desligamento inadequado. Essas tabelas não registradas em log são automaticamente truncadas após uma falha ou um desligamento inadequado. Os respectivos conteúdos e índices também não são replicados para servidores em espera.

Em contrapartida, o Aurora PostgreSQL lida com tabelas não registradas em log de forma diferente devido à sua arquitetura de armazenamento distribuído. Isso ocorre porque o sistema de armazenamento do Aurora não depende do WAL tradicional do PostgreSQL para durabilidade. Contudo, os benefícios de performance normalmente associados às tabelas não registradas em log no PostgreSQL padrão podem não ser tão significativos no Aurora. Isso se deve à arquitetura de armazenamento distribuído do Aurora, que pode introduzir despesas operacionais adicionais em comparação ao armazenamento local usado no PostgreSQL padrão.

Ao usar tabelas não registradas em log no Aurora PostgreSQL, pense no seguinte:

- É possível acessar tabelas não registradas em log somente por meio do nó de gravador no cluster de banco de dados do Aurora.

- Os nós de leitor podem acessar tabelas não registradas em log somente quando promovidos ao status de gravador.
- Quando você tentar acessar tabelas não registradas em log por meio de um nó de leitor, o seguinte erro será gerado:

```
cannot access temporary or unlogged relations during recovery.
```

Criar tabelas não registradas em log

Para criar uma tabela não registrada em log no Aurora PostgreSQL, adicione a palavra-chave UNLOGGED na declaração CREATE TABLE:

```
CREATE UNLOGGED TABLE staging_sales_data (
    transaction_id bigint,
    customer_id bigint,
    product_id bigint,
    transaction_date date,
    amount NUMERIC
);
```

Manipular tabelas não registradas em log durante a migração

Ao se preparar para migrar dados para o Aurora PostgreSQL, é importante identificar e lidar adequadamente com as tabelas não registradas. As tabelas não registradas em log não são registradas no WAL e são excluídas do fluxo de replicação, o que significa que elas não serão copiadas em uma réplica de leitura do Aurora.

Converta tabelas não registradas em log em tabelas registradas ou descarte-as se não estiverem sendo usadas antes de criar uma réplica de leitura do Aurora.

Para verificar a presença de tabelas não registradas em log em cada banco de dados na instância, use o seguinte comando:

```
SELECT oid, relfilenode, relname, relpersistence, relkind
FROM pg_class
WHERE relpersistence ='u';
```

Para converter uma tabela não registrada em log em uma tabela registrada em log, use o seguinte comando:

```
ALTER TABLE table_name SET LOGGED;
```

Essa operação regrava a tabela inteira e coloca um bloqueio exclusivo nela até que a conclusão. Para tabelas grandes, isso pode provocar um tempo de inatividade significativo.

Converter tabelas não registradas em log em tabelas registradas em log

Quando precisar reconverter uma tabela não registrada em log em uma tabela registrada em log, você pode usar o seguinte comando:

```
ALTER TABLE table_name SET LOGGED;
```

Essa operação regrava a tabela inteira e coloca um bloqueio exclusivo nela até que a operação seja concluída. Para tabelas grandes, isso pode provocar um tempo de inatividade significativo.

Tabelas não registradas em log e replicação lógica

As tabelas não registradas em log geralmente não são incluídas na replicação lógica porque a replicação lógica depende do WAL para capturar e transferir alterações. Por padrão, as alterações nas tabelas não registradas em log não são registradas no WAL e são excluídas do fluxo de replicação, o que as torna inadequadas para casos de uso em que a replicação lógica é necessária. No entanto, o Aurora PostgreSQL fornece um parâmetro, chamado `rds.logically_replicate_unlogged_tables`, que permite controlar esse comportamento:

- Quando `rds.logically_replicate_unlogged_tables` é definido como 0 (desativado), as tabelas não registradas em log são excluídas da replicação lógica.
- Quando `rds.logically_replicate_unlogged_tables` é definido como 1 (ativado), as tabelas não registradas em log são incluídas na replicação lógica.

> **Note**
>
> No Aurora PostgreSQL, o parâmetro `rds.logically_replicate_unlogged_tables` é definido por padrão como 1 (ativado) nas versões 14 e anteriores e como 0 (desativado) nas versões 15 e posteriores.

Trabalhar com o autovacuum do PostgreSQL no Amazon Aurora PostgreSQL

É altamente recomendável que você use o recurso de autovacuum para bancos de dados PostgreSQL a fim de manter a integridade de sua instância de banco de dados PostgreSQL. O autovacuum automatiza a execução dos comandos VACUUM e ANALYZE. Ele verifica as tabelas com um grande número de tuplas inseridas, atualizadas ou excluídas. Após essa verificação, ele recupera o armazenamento removendo dados obsoletos ou tuplas do banco de dados PostgreSQL.

Por padrão, o autovacuum é ativado nas instâncias de banco de dados do Aurora PostgreSQL que você cria usando qualquer um dos grupos de parâmetros de banco de dados padrão do PostgreSQL. Outros parâmetros de configuração associados ao recurso autovacuum também são definidos por padrão. Como esses padrões são genéricos, você pode se beneficiar do ajuste de alguns dos parâmetros associados ao recurso autovacuum para seu workload específico.

A seguir, você pode encontrar mais informações sobre o autovacuum e como ajustar alguns dos respectivos parâmetros em sua instância de banco de dados do Aurora PostgreSQL.

Tópicos

- Alocar memória para autovacuum
- Reduzir a probabilidade de conclusão de IDs de transação
- Determinar se as tabelas no seu banco de dados precisam de vacuum
- Determinar quais tabelas são atualmente elegíveis para autovacuum
- Determinar se o autovacuum está em execução e por quanto tempo
- Realização de um congelamento manual de vacuum
- Reindexação de uma tabela quando o autovacuum está em execução
- Gerenciar o autovacuum com grandes índices
- Outros parâmetros que afetam o autovacuum

- Definir parâmetros de autovacuum em nível de tabela
- Registrar atividades do autovacuum e do vacuum em log
- Noções básicas sobre o comportamento do autovacuum com bancos de dados inválidos
- Identificar e resolver bloqueadores de limpeza agressivos no Aurora PostgreSQL

Alocar memória para autovacuum

Um dos parâmetros mais importantes que influenciam a performance do autovacuum é o `autovacuum_work_mem`. No Aurora PostgreSQL versões 14 e anteriores, o parâmetro `autovacuum_work_mem` é definido como -1, indicando que a configuração de `maintenance_work_mem` é usada em vez disso. Para todas as outras versões, `autovacuum_work_mem` é determinado por GREATEST({DBInstanceClassMemory/32768}, 65536).

As operações vacuum manuais sempre usam a configuração `maintenance_work_mem`, com uma configuração padrão de GREATEST({DBInstanceClassMemory/63963136*1024}, 65536), e também é possível fazer o ajuste no nível da sessão usando o comando `SET` para operações `VACUUM` manuais mais específicas.

O `autovacuum_work_mem` determina a memória do autovacuum para armazenar identificadores de tuplas mortas (`pg_stat_all_tables.n_dead_tup`) para índices de vacuum.

Ao executar cálculos para determinar o valor do parâmetro `autovacuum_work_mem`, esteja ciente do seguinte:

- Se você definir o parâmetro com um valor muito baixo, o processo de vacuum talvez precise verificar a tabela várias vezes para concluir o trabalho. Essas várias verificações podem ter um impacto negativo sobre a performance. Para instâncias maiores, a configuração de `maintenance_work_mem` ou `autovacuum_work_mem` para pelo menos 1 GB pode melhorar o desempenho de vaccum nas tabelas com um grande número de tuplas mortas. No entanto, no PostgreSQL versões 16 e anteriores, o uso de memória do vacuum é limitado a 1 GB, o que é suficiente para processar aproximadamente 179 milhões de tuplas mortas em uma única passagem. Se uma tabela tiver mais tuplas mortas do que isso, o vacuum precisará fazer várias passagens pelos índices da tabela, aumentando significativamente o tempo necessário. A partir da versão 17 do PostgreSQL, não há um limite de 1 GB, e o autovacuum pode processar mais de 179 milhões de tuplas usando árvores radix.

  Um identificador de tupla tem 6 bytes de tamanho. Para estimar a memória necessária para limpar um índice de uma tabela, consulte `pg_stat_all_tables.n_dead_tup` para encontrar

o número de tuplas mortas e multiplique esse número por seis para determinar a memória necessária para realizar vacuum do índice em uma única passagem. Você pode usar a seguinte consulta:

```
SELECT
    relname AS table_name,
    n_dead_tup,
    pg_size_pretty(n_dead_tup * 6) AS estimated_memory
FROM
    pg_stat_all_tables
WHERE
    relname = 'name_of_the_table';
```

- O parâmetro `autovacuum_work_mem` funciona em conjunto com o parâmetro `autovacuum_max_workers`. Cada operador entre `autovacuum_max_workers` pode usar a memória que você alocar. Se você tiver muitas tabelas pequenas, aloque mais `autovacuum_max_workers` e menos `autovacuum_work_mem`. Se você tiver tabelas grandes (com mais de 100 GB), aloque mais memória e menos processos de operadores. Você precisa ter memória suficiente alocada para ter sucesso na sua maior tabela. Portanto, garanta que a combinação de processos de operadores e memória seja igual à memória total que deseja alocar.

Reduzir a probabilidade de conclusão de IDs de transação

Em alguns casos, as configurações de grupos de parâmetros relacionadas ao autovacuum podem não ser agressivas o suficiente para evitar a conclusão de IDs de transação. Para resolver isso, o Aurora PostgreSQL oferece um mecanismo que adapta automaticamente os valores dos parâmetros de autovacuum. O autovacuum adaptável é um recurso do Aurora PostgreSQL. Uma explicação detalhada da conclusão de TransactionID encontra-se na documentação do PostgreSQL.

O autovacuum adaptável está ativado por padrão para instâncias do Aurora PostgreSQL com o parâmetro dinâmico `rds.adaptive_autovacuum` definido como ATIVADO. É altamente recomendável manter esse recurso ativado. No entanto, para desabilitar esse recurso, defina o parâmetro `rds.adaptive_autovacuum` como 0 ou OFF (desativado).

O wraparound de ID de transação ainda é possível mesmo quando o Aurora ajusta os parâmetros de autovacuum. Nós encorajamos que você implemente um alarme do Amazon CloudWatch para a conclusão de IDs de transação. Para ter mais informações, consulte a postagem Implement an early warning system for transaction ID wraparound in RDS for PostgreSQL (Implementar um sistema de

alertas antecipados para conclusão de IDs de transação no RDS for PostgreSQL) no Blog de banco de dados da AWS.

Com o ajuste de parâmetros de autovacuum adaptável ativado, o Amazon RDS começa a ajustar parâmetros de autovacuum quando a métrica do CloudWatch `MaximumUsedTransactionIDs` atingir o valor do parâmetro `autovacuum_freeze_max_age` ou 500.000.000, o que for maior.

O Amazon RDS continuará a ajustar os parâmetros para o autovacuum se uma tabela continuar a tendência para a conclusão de IDs de transação. Cada um desses ajustes dedica mais recursos ao autovacuum para evitar o envolvimento. O Amazon RDS atualiza os seguintes parâmetros relacionados ao autovacuum:

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`
- autovacuum_naptime

O RDS modificará esses parâmetros somente se o novo valor tornar o autovacuum mais agressivo. Os parâmetros são modificados na memória na instância de banco de dados. Os valores no grupo de parâmetros não são alterados. Para visualizar as configurações atuais na memória, use o comando SQL SHOW PostgreSQL.

Quando o Amazon RDS modifica qualquer um desses parâmetros de autovacuum, ele gera um evento para a instância de banco de dados afetada. Esse evento é visível no Console de gerenciamento da AWS e por meio da API do Amazon RDS. Depois que a métrica `MaximumUsedTransactionIDs` do CloudWatch volta a ficar abaixo do limite, o Amazon RDS redefine os parâmetros relacionados a autovacuum na memória de volta aos valores especificados no grupo de parâmetros. Em seguida, ele gera outro evento correspondente a essa alteração.

Determinar se as tabelas no seu banco de dados precisam de vacuum

Você pode usar a consulta a seguir para mostrar o número de transações sem restrições em um banco de dados. A coluna `datfrozenxid` de uma linha `pg_database` do banco de dados é um limite inferior nos IDs de transação normais que aparecem nesse banco de dados. Essa coluna é o mínimo dos valores de `relfrozenxid` por tabela no banco de dados.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

Por exemplo, os resultados da execução da consulta anterior podem ser os seguintes.

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

Quando a idade de um banco de dados atingir 2 bilhões de IDs de transação, ocorrerá a conclusão de IDs de transação (XID), e o banco de dados se tornará somente leitura. Use essa consulta para produzir uma métrica e execute-a algumas vezes por dia. Por padrão, o autovacuum é definido para manter a idade das transações como um máximo de 200,000,000 (`autovacuum_freeze_max_age`).

Uma estratégia de monitoramento de exemplo pode ser assim:

- Defina o valor `autovacuum_freeze_max_age` como 200 milhões de transações.
- Se uma tabela atingir 500 milhões de transações sem restrições, um alarme de baixa gravidade será disparado. Este não é um valor não razoável, mas pode indicar que o autovacuum não está dando conta.
- Se uma tabela chegar a 1 bilhão, ela deverá ser tratada como um alarme que requer medidas. Em geral, convém manter as idades mais próximas de `autovacuum_freeze_max_age` por motivos de performance. Recomendamos que você investigue usando as recomendações a seguir.
- Se uma tabela atingir 1,5 bilhões de transações sem vacuum, um alarme de alta gravidade será disparado. Dependendo da rapidez com que seu banco de dados usa IDs de transação, esse alarme pode indicar que o sistema está ficando sem tempo para executar o autovacuum. Nesse caso, recomendamos que você resolva isso imediatamente.

Se uma tabela estiver ultrapassando constantemente esses limiares, modifique ainda mais seus parâmetros de autovacuum. Por padrão, usar VACUUM manualmente (que tem atrasos baseados em custos desabilitados) é mais agressivo do que usar o autovacuum padrão, mas também é mais intrusivo para o sistema como um todo.

Recomendamos o seguinte:

- Esteja ciente e ative um mecanismo de monitoramento para que você tenha conhecimento da idade das transações mais antigas.

  Para obter informações sobre como criar um processo que avisa sobre a conclusão de IDs de transação, consulte a publicação do blog de banco de dados da AWS Implement an early warning system for transaction ID wraparound no Amazon RDS for PostgreSQL (Implementar um sistema de alerta antecipado para conclusão de IDs de transação no Amazon RDS for PostgreSQL).
- Para tabelas mais ocupadas, execute um congelamento de vacuum manual regularmente durante uma janela de manutenção além de depender do autovacuum. Para obter informações sobre a realização de um congelamento manual de vacuum, consulte Realização de um congelamento manual de vacuum.

Determinar quais tabelas são atualmente elegíveis para autovacuum

Muitas vezes, uma ou duas tabelas precisam de vacuum. Tabelas cujo valor de `relfrozenxid` é maior que o número de transações em `autovacuum_freeze_max_age` são sempre visadas pelo autovacuum. Caso contrário, se o número de tuplas obsoletas desde o último VACUUM exceder o limite de vacuum, um vacuum será realizado na tabela.

O limite de autovacuum é definido como:

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

em que o `vacuum base threshold` é `autovacuum_vacuum_threshold`, o `vacuum scale factor` é `autovacuum_vacuum_scale_factor` e o `number of tuples` é `pg_class.reltuples`.

Enquanto você estiver conectado ao seu banco de dados, execute a consulta a seguir para ver uma lista de tabelas que o autovacuum considera elegíveis para o vacuum.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
```

```
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determinar se o autovacuum está em execução e por quanto tempo

Se você precisa realizar vacuum em uma tabela manualmente, determine se o autovacuum está sendo executado. Se estiver, talvez seja necessário ajustar parâmetros para fazê-lo funcionar mais eficientemente ou desativar o autovacuum temporariamente para poder executar VACUUM manualmente.

Use a seguinte consulta para determinar se o autovacuum está sendo executado e por quanto tempo, e se ele está esperando por outra sessão.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

Depois de executar a consulta, você deverá ver uma saída semelhante à seguinte:

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
         |          |       |        |            |                         | FROM
 pg_stat_activity
           +
         |          |       |        |            |                         | WHERE
 query like '%VACUUM%'
          +
         |          |       |        |            |                         | ORDER BY
 xact_start;
       +
```

Vários problemas podem causar uma sessão de autovacuum de longa execução (ou seja, vários dias de duração). O problema mais comum é que o valor do seu parâmetro `maintenance_work_mem` está muito baixo para o tamanho da tabela ou a taxa de atualizações.

Recomendamos o uso da fórmula a seguir para definir o valor do parâmetro `maintenance_work_mem`.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Sessões de autovacuum em execução por pouco tempo também podem indicar problemas:

- Isso pode indicar que não há `autovacuum_max_workers` suficientes para sua workload. Neste caso, você precisa indicar o número de operadores.
- Pode indicar que existe uma corrupção de índice (o autovacuum trava e reinicia na mesma relação, mas não faz nenhum progresso). Nesse caso, execute um `vacuum freeze verbose` *`table`* manual para ver a causa exata.

Realização de um congelamento manual de vacuum

Você pode querer realizar um vacuum manual em uma tabela que tenha um processo de vacuum já em execução. Isso é útil se você identificou uma tabela com uma idade XID próxima de 2 bilhões de transações (ou acima de qualquer limite que você esteja monitorando).

As etapas a seguir são uma orientação com diversas variações no processo. Por exemplo, durante testes, suponha que você perceba que o valor do parâmetro `maintenance_work_mem` está muito baixo e que precisa tomar medidas imediatas em uma tabela. No entanto, talvez você não queira devolver a instância no momento. Usando as consultas nas seções anteriores, você determina qual tabela é o problema e observa uma sessão de autovacuum de longa execução. Você sabe que precisa alterar a configuração do parâmetro `maintenance_work_mem`, mas também precisa tomar medidas imediatas e evacuar a tabela em questão. O procedimento a seguir mostra o que fazer nessa situação.

Para executar manualmente um congelamento de vacuum

1. Abra duas sessões no banco de dados que contém a tabela em que você deseja realizar vacuum. Para a segunda sessão, use "screen" ou outro utilitário que mantenha a sessão se a conexão for encerrada.
2. Na sessão um, obtenha o “Process ID” (PID – ID de processo) da sessão de autovacuum em execução na tabela.

   Execute a seguinte consulta para obter o PID da sessão de autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```

3. Na sessão dois, calcule a quantidade de memória necessária para essa operação. Neste exemplo, determinamos que podemos usar até 2 GB de memória para essa operação e, portanto, definimos `maintenance_work_mem` da sessão atual como 2 GB.

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```

4. Na sessão dois, emita um comando `vacuum freeze verbose` para a tabela. A configuração detalhada é útil porque, embora não haja um relatório de andamento para isso no PostgreSQL, você pode ver as atividades.

```
\timing on
Timing is on.
vacuum freeze verbose pgbench_branches;
```

```
INFO:  vacuuming "public.pgbench_branches"
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. Na sessão um, se o autovacuum estava bloqueando a sessão de vacuum, `pg_stat_activity` mostrará que a espera é T para a sessão de vacuum. Nesse caso, encerre o processo de autovacuum da maneira a seguir.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Algumas versões anteriores do Amazon Aurora não podem encerrar um processo de autovacuum usando o comando anterior e falham com o seguinte erro: `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227`. Para encontrar as versões do PostgreSQL que foram corrigidas, pesquise o seguinte tópico em Atualizações do Amazon Aurora PostgreSQL:

```
Allow rds_superuser to terminate backends which are not explicitly
 associated with a role
```

Nesse ponto, sua sessão começa. O autovacuum é reiniciado imediatamente, pois essa tabela é provavelmente a mais alta em sua lista de trabalho.

6. Inicie seu comando `vacuum freeze verbose` na sessão dois e depois termine o processo de autovacuum na sessão um.

Reindexação de uma tabela quando o autovacuum está em execução

Se um índice se tornar corrompido, o autovacuum continuará processando a tabela e falhará. Se você tentar um vacuum manual nessa situação, receberá uma mensagem de erro semelhante à mostrada a seguir.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

Quando o índice está corrompido e o autovacuum está tentando ser executado na tabela, você lutará com uma sessão de autovacuum já em execução. Ao emitir um comando REINDEX, você remove um bloqueio exclusivo na tabela. As operações de gravação são bloqueadas, bem como as operações de leitura que usam esse índice específico.

Para reindexar uma tabela quando o autovacuum está em execução nela

1. Abra duas sessões no banco de dados que contém a tabela em que você deseja realizar vacuum. Para a segunda sessão, use "screen" ou outro utilitário que mantenha a sessão se a conexão for encerrada.
2. Na sessão um, obtenha o PID da sessão de autovacuum em execução na tabela.

   Execute a seguinte consulta para obter o PID da sessão de autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   ```

```
xact_start;
```

3. Na segunda sessão, emita o comando reindex.

```
\timing on
Timing is on.
reindex index pgbench_branches_test_index;
REINDEX
  Time: 9.966 ms
```

4. Na sessão um, se o autovacuum estava bloqueando o processo, você verá em `pg_stat_activity` que a espera é "T" para a sua sessão de vacuum. Nesse caso, você encerra o processo de autovacuum.

```
SELECT pg_terminate_backend('the_pid');
```

   Nesse ponto, sua sessão começa. É importante observar que o autovacuum é reiniciado imediatamente, pois essa tabela é provavelmente a mais alta em sua lista de trabalho.

5. Inicie seu comando na sessão dois e depois termine o processo de autovacuum na sessão um.

Gerenciar o autovacuum com grandes índices

Como parte de sua operação, o autovacuum executa várias fases de aspiração ao trabalhar em uma tabela. Antes de a tabela ser limpa, todos os seus índices são aspirados primeiro. Ao remover vários índices grandes, essa fase consome uma quantidade significativa de tempo e recursos. Portanto, como prática recomendada, controle o número de índices em uma tabela e elimine os índices não usados.

Para esse processo, primeiro confira o tamanho geral do índice. Depois, determine se há índices possivelmente não usados que podem ser removidos conforme mostrado nos exemplos a seguir.

Como conferir o tamanho da tabela e os respectivos índices

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
```

```
pg_size_pretty
11 GB
(1 row)
```

Neste exemplo, o tamanho dos índices é maior do que a tabela. Essa diferença pode causar problemas de performance, pois os índices estão sobrecarregados ou não são usados, o que afeta as operações de autovacuum e de inserção.

Como conferir índices não usados

Usando a visualização pg_stat_user_indexes, você pode conferir com que frequência um índice é usado com a coluna `idx_scan`. No exemplo a seguir, os índices não usados têm o valor `idx_scan` de `0`.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+--------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
```

```
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

> **Note**
>
> Essas estatísticas são incrementais a partir do momento em que as estatísticas são redefinidas. Suponha que você tenha um índice usado apenas no final de um trimestre comercial ou apenas para um relatório específico. É possível que esse índice não tenha sido usado desde que as estatísticas foram redefinidas. Para ter mais informações, consulte Funções de estatística. Os índices usados para impor a exclusividade não terão verificações realizadas e não devem ser identificados como índices não usados. Para identificar os índices não usados, você deve ter um conhecimento profundo da aplicação e das respectivas consultas.

Para conferir quando as estatísticas foram redefinidas pela última vez em um banco de dados, use `pg_stat_database`

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

Aspirar uma tabela o mais rápido possível

RDS para PostgreSQL 12 e posterior

Se você tiver muitos índices em uma tabela grande, a instância de banco de dados poderá estar se aproximando do encapsulamento de ID (XID), que é quando o contador XID chega a zero. Se não for conferida, essa situação poderá ocasionar perda de dados. No entanto, você pode aspirar rapidamente a tabela sem limpar os índices. No RDS para PostgreSQL 12 e posterior, você pode usar VACUUM com a cláusula `INDEX_CLEANUP`.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

Se uma sessão de autovacuum já estiver em execução, você deverá encerrá-la para iniciar a aspiração manual. Para ter informações sobre a realização de um congelamento manual de aspiração, consulte Realização de um congelamento manual de vacuum.

> **Note**
>
> Ignorar a limpeza do índice regularmente causa sobrecarga no índice, o que diminui a performance geral da verificação. O índice retém as linhas inativas e a tabela retém os ponteiros das linhas inativas. Consequentemente, `pg_stat_all_tables.n_dead_tup` aumenta até que o autovacuum ou um VACUUM manual com limpeza de índice seja executado. Como prática recomendada, use esse procedimento somente para evitar o encapsulamento de ID.

RDS para PostgreSQL 11 e versões mais antigas

No entanto, no RDS para PostgreSQL 11 e versões anteriores, a única maneira de permitir que a aspiração seja concluída mais rapidamente é reduzir o número de índices em uma tabela. A eliminação de um índice pode afetar os planos de consulta. Recomendamos que você elimine primeiro os índices não usados e, depois, descarte os índices quando o encapsulamento XID estiver muito próximo. Depois que o processo de aspiração for concluído, você poderá recriar esses índices.

Outros parâmetros que afetam o autovacuum

A consulta a seguir mostra os valores de alguns dos parâmetros que afetam diretamente o autovacuum e seu comportamento. Os parâmetros de autovacuum são descritos inteiramente na documentação do PostgreSQL.

```
SELECT name, setting, unit, short_desc
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

Enquanto todos estes afetem o autovacuum, alguns dos mais importantes são:

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

Definir parâmetros de autovacuum em nível de tabela

Você pode definir parâmetros de armazenamento relacionados ao autovacuum em nível de tabela, o que pode ser melhor do que alterar o comportamento do banco de dados inteiro. Para tabelas grandes, talvez você precise definir configurações agressivas e talvez você não queira que o autovacuum se comporte dessa maneira para todas as tabelas.

A consulta a seguir mostra quais tabelas atualmente têm opções em nível de tabela.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

Um exemplo em que isso pode ser útil é em tabelas que são muito maiores do que o resto das suas tabelas. Suponha que você tenha uma tabela de 300 GB e outras 30 tabelas com menos de 1 GB. Nesse caso, você pode definir alguns parâmetros específicos para a sua tabela grande, para não alterar o comportamento de todo o sistema.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

Fazer isso desativará o atraso de autovacuum baseado em custos para essa tabela, mas aumentará o uso de recursos no seu sistema. Normalmente, o autovacuum faz uma pausa durante `autovacuum_vacuum_cost_delay` sempre que `autovacuum_cost_limit` é alcançado. Para obtert mais detalhes, consulte a documentação do PostgreSQL sobre vacuum baseado no custo.

Registrar atividades do autovacuum e do vacuum em log

Informações sobre atividades de autovacuum são enviadas para `postgresql.log` com base no nível especificado no parâmetro `rds.force_autovacuum_logging_level`. A seguir estão os valores permitidos para esse parâmetro e as versões do PostgreSQL para as quais esse valor é a configuração padrão:

- `disabled` (PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info` (PostgreSQL 12, PostgreSQL 11)
- `notice`
- `warning` (PostgreSQL 13 e superior)
- `error`, log, `fatal`, `panic`

A configuração `rds.force_autovacuum_logging_level` funciona com o parâmetro `log_autovacuum_min_duration`. O valor do parâmetro `log_autovacuum_min_duration` é o limite (em milissegundos) acima do qual as ações autovacuum são registradas em log. Uma configuração `-1` não registra nada em log, enquanto uma configuração 0 registra todas as ações

em log. Como ocorre com `rds.force_autovacuum_logging_level`, os valores padrão de `log_autovacuum_min_duration` são dependentes da versão, da seguinte maneira:

- `10000 ms`: PostgreSQL 14, PostgreSQL 13, PostgreSQL 12 e PostgreSQL 11
- `(empty)`: nenhum valor padrão para PostgreSQL 10 e PostgreSQL 9.6

Recomendamos que você defina `rds.force_autovacuum_logging_level` como `WARNING`. Também recomendamos definir `log_autovacuum_min_duration` para um valor entre 1000 e 5000. Uma configuração 5000 registra em log a atividade que leva mais de 5.000 milissegundos. Qualquer configuração diferente de -1 também registrará mensagens se a ação autovacuum for ignorada devido a um bloqueio conflitante ou relações descartadas simultaneamente. Para ter mais informações, consulte Automatic Vacuuming (Aplicação automática do vacuum) na documentação do PostgreSQL.

Para solucionar problemas, você pode alterar o parâmetro `rds.force_autovacuum_logging_level` para um dos níveis de depuração, de `debug1` a `debug5`, para obter informações mais detalhadas. Recomendamos que você use as configurações de depuração por curtos períodos e apenas para fins de solução de problemas. Para saber mais, consulte When to log (Quando registrar em log) na documentação do PostgreSQL.

> **Note**
>
> O PostgreSQL permite que a conta `rds_superuser` visualize sessões de autovacuum em `pg_stat_activity`. Por exemplo, você pode identificar e encerrar uma sessão de autovacuum que esteja bloqueando a execução de um comando, ou executando de forma mais lenta do que um comando de vacuum emitido manualmente.

Noções básicas sobre o comportamento do autovacuum com bancos de dados inválidos

Um novo valor `-2` é introduzido na coluna `datconnlimit` do catálogo `pg_database` para indicar bancos de dados que foram interrompidos no meio da operação DROP DATABASE como inválidos.

Esse novo valor está disponível nas seguintes versões do Aurora PostgreSQL:

- 15.4 e todas as versões posteriores

- 14.9 e versões posteriores
- 13.12 e versões posteriores
- 12.16 e versões posteriores
- 11.21 e versões posteriores

Bancos de dados inválidos não afetam a capacidade do autovacuum de congelar a funcionalidade de bancos de dados válidos. O Autovacuum ignora bancos de dados inválidos. Consequentemente, as operações regulares de limpeza continuarão funcionando de forma adequada e eficiente em todos os bancos de dados válidos no ambiente do PostgreSQL.

Tópicos

- ID da transação de monitoramento
- Ajustar a consulta de monitoramento
- Resolver o problema de banco de dados inválido

ID da transação de monitoramento

A função `age(datfrozenxid)` geralmente é usada para monitorar a idade do ID de transação (XID) dos bancos de dados para evitar o encapsulamento de IDs de transação.

Como bancos de dados inválidos são excluídos do autovacuum, seu contador de ID de transação (XID) pode atingir o valor máximo de `2 billion`, ser encapsulado em `- 2 billion` e continuar esse ciclo indefinidamente. Uma consulta típica para monitorar o encapsulamento de IDs de transação pode ser:

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

No entanto, com a introdução do valor -2 para `datconnlimit`, bancos de dados inválidos podem distorcer os resultados dessa consulta. Como esses bancos de dados não são válidos e não devem fazer parte das verificações de manutenção regulares, eles podem causar falsos positivos, levando você a acreditar que o `age(datfrozenxid)` é maior do que realmente é.

Ajustar a consulta de monitoramento

Para garantir um monitoramento preciso, você deve ajustar sua consulta de monitoramento para excluir bancos de dados inválidos. Siga esta consulta recomendada:

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

Essa consulta garante que somente bancos de dados válidos sejam considerados no cálculo `age(datfrozenxid)`, fornecendo um reflexo real da idade do ID da transação no ambiente do PostgreSQL.

Resolver o problema de banco de dados inválido

Ao tentar se conectar a um banco de dados inválido, talvez seja exibida uma mensagem de erro semelhante à seguinte:

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

Além disso, se o parâmetro `log_min_messages` estiver definido como `DEBUG2` ou posterior, você poderá observar as seguintes entradas de log indicando que o processo de autovacuum está ignorando o banco de dados inválido:

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

Para resolver o problema, siga a `HINT` fornecida durante a tentativa de conexão. Conecte-se a qualquer banco de dados válido usando sua conta principal do RDS ou uma conta de banco de dados com o perfil `rds_superuser` e elimine bancos de dados inválidos.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

Identificar e resolver bloqueadores de limpeza agressivos no Aurora PostgreSQL

No PostgreSQL, a limpeza é vital para garantir a integridade do banco de dados, pois recupera o armazenamento e evita problemas de conclusão do ID da transação. No entanto, há momentos em que a limpeza pode ser impedida de operar conforme desejado, o que pode resultar em degradação do desempenho e sobrecarga do armazenamento e até mesmo afetar a disponibilidade da instância de banco de dados pelo wraparound de ID de transação. Portanto, identificar e resolver esses problemas é essencial para o desempenho e a disponibilidade ideais do banco de dados. Leia Understanding autovacuum in Amazon RDS for PostgreSQL environments para saber mais sobre o autovacuum.

A função `postgres_get_av_diag()` ajuda a identificar problemas que impedem ou atrasam o progresso da limpeza agressiva. São fornecidas sugestões, que podem incluir comandos para resolver o problema onde ele é identificável ou orientações para diagnósticos adicionais quando o problema não é identificável. Bloqueadores de limpeza agressiva são relatados quando a idade excede o limite de autovacuum adaptativo do RDS de 500 milhões de IDs de transação.

Qual é a idade do ID da transação?

A função `age()` para IDs de transação calcula o número de transações que ocorreram desde o ID de transação não congelado mais antigo de um banco de dados (`pg_database.datfrozenxid`) ou tabela (`pg_class.relfrozenxid`). Esse valor indica a atividade do banco de dados desde a última operação de limpeza agressiva e destaca a provável workload para os próximos processos de VACUUM.

O que é uma limpeza agressiva?

Uma operação VACUUM agressiva conduz uma varredura abrangente de todas as páginas em uma tabela, incluindo aquelas normalmente ignoradas durante operações VACUUM regulares. Essa

verificação completa visa "congelar" os IDs de transação que se aproximam de sua idade máxima, evitando efetivamente uma situação conhecida como conclusão de ID de transação.

Para que `postgres_get_av_diag()` reporte bloqueadores, o bloqueador deve ter pelo menos 500 milhões de transações de idade.

Tópicos

- Instalar ferramentas de monitoramento e diagnóstico de autovacuum no Aurora PostgreSQL
- Funções de postgres_get_av_diag() no Aurora PostgreSQL
- Solucionar bloqueadores de limpeza identificáveis no Aurora PostgreSQL
- Solucionar bloqueadores de limpeza não identificáveis no Aurora PostgreSQL
- Resolver problemas de desempenho de limpeza no Aurora PostgreSQL
- Explicação das mensagens de AVISO no Aurora PostgreSQL

Instalar ferramentas de monitoramento e diagnóstico de autovacuum no Aurora PostgreSQL

No momento, a função `postgres_get_av_diag()` está disponível nas seguintes versões do Aurora PostgreSQL:

- 17.4 e versões 17 posteriores
- 16.7 e versões 16 posteriores
- 15.11 e versões 15 posteriores
- 14.16 e versões 14 posteriores
- 13.19 e versões 13 posteriores

Para usar `postgres_get_av_diag()`, crie a extensão `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

Para verificar se a extensão está instalada.

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
```

```
 ----------+---------+-----------
+-------------------------------------------------------------
 rds_tools |   1.9   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

Verifique se a função foi criada.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

Funções de postgres_get_av_diag() no Aurora PostgreSQL

A função `postgres_get_av_diag()` recupera informações de diagnóstico sobre processos de autovacuum que estão bloqueando ou atrasando em um banco de dados do Aurora PostgreSQL. A consulta precisa ser executada no banco de dados com o ID de transação mais antigo para obter resultados precisos. Para obter mais informações sobre como usar o banco de dados com o ID de transação mais antigo, consulte Não conectado ao banco de dados com o ID de transação mais antigo

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
```

```
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

A função `postgres_get_av_diag()` retorna uma tabela com as informações a seguir:

blocker

Especifica a categoria da atividade do banco de dados que está bloqueando o vacuum.

- Instrução ativa
- Ocioso na transação
- Transação preparada
- Slot de replicação lógica
- Instâncias de leitor
- Tabelas temporárias

banco de dados

Especifica o nome do banco de dados, quando aplicável e suportado. Este é o banco de dados no qual a atividade está em andamento e bloqueia ou bloqueará o autovacuum. Este é o banco de dados ao qual você precisa se conectar e tomar ação.

blocker_identifier

Especifica o identificador da atividade que está bloqueando ou bloqueará o autovacuum. O identificador pode ser um ID de processo junto com uma instrução SQL, uma transação preparada, um endereço IP de uma réplica de leitura e o nome do slot de replicação, seja lógico ou físico.

wait_event

Especifica o evento de espera da sessão de bloqueio e é aplicável para os seguintes bloqueadores:

- Instrução ativa
- Ocioso na transação

autovacuum_lagging_by

Especifica o número de transações que o autovacuum está atrasado no trabalho pendente por categoria.

suggestion

Especifica sugestões para resolver o bloqueador. Essas instruções incluem o nome do banco de dados no qual a atividade existe, quando aplicável, o ID do processo (PID) da sessão, quando aplicável, e a ação a ser tomada.

suggested_action

Sugere a ação que precisa ser tomada para resolver o bloqueador.

Solucionar bloqueadores de limpeza identificáveis no Aurora PostgreSQL

O Autovacuum realiza limpezas agressivas e reduz a idade dos IDs de transação para abaixo do limite especificado pelo parâmetro `autovacuum_freeze_max_age` de sua instância do RDS. É possível rastrear essa idade usando a métrica `MaximumUsedTransactionIDs` do Amazon CloudWatch.

Para encontrar a configuração de `autovacuum_freeze_max_age` (que tem um padrão de 200 milhões de IDs de transação) da sua instância do Amazon RDS, use a seguinte consulta:

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

Observe que `postgres_get_av_diag()` só verifica se há bloqueadores de limpeza agressiva quando a idade excede o limite de 500 milhões de IDs de transação do autovacuum adaptativo do Amazon RDS. Para que `postgres_get_av_diag()` detecte bloqueadores, o bloqueador deve ter pelo menos 500 milhões de transações de idade.

A função `postgres_get_av_diag()` identifica os seguintes tipos de bloqueadores:

Tópicos

- Instrução ativa
- Ocioso na transação
- Transação preparada
- Slot de replicação lógica

- Instâncias de leitor
- Tabelas temporárias

Instrução ativa

No PostgreSQL, uma instrução ativa é uma instrução SQL que está sendo executada atualmente pelo banco de dados. Isso inclui consultas, transações ou quaisquer operações em andamento. Ao monitorar por meio do `pg_stat_activity`, a coluna de estado indica que o processo com o PID correspondente está ativo.

A função `postgres_get_av_diag()` exibe uma saída semelhante à saída a seguir quando identifica uma instrução que é uma instrução ativa.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

Ação sugerida

Seguindo as orientações na coluna `suggestion`, o usuário pode se conectar ao banco de dados em que a instrução ativa está presente e, conforme especificado na coluna `suggested_action`, é recomendável analisar cuidadosamente a opção de encerrar a sessão. Se o encerramento for seguro, use a função `pg_terminate_backend()` para encerrar a sessão. Essa ação pode ser executada por um administrador (como a conta mestra do RDS) ou por um usuário com o privilégio `pg_terminate_backend()` necessário.

> **Warning**
>
> Uma sessão encerrada reverterá (`ROLLBACK`) as alterações feitas. Dependendo de seus requisitos, pode ser necessário executar a instrução novamente. No entanto, é recomendável fazer isso somente após o processo do autovacuum terminar a operação de limpeza agressiva.

Ocioso na transação

Uma instrução ociosa na transação se refere a qualquer sessão que abriu uma transação explícita (como emitir uma instrução `BEGIN`), realizou algum trabalho e agora está esperando que o cliente passe mais trabalho ou sinalize o final da transação emitindo um `COMMIT`, `ROLLBACK` ou `END` (o que resultaria em um `COMMIT` implícito).

A função `postgres_get_av_diag()` exibe uma saída semelhante à saída a seguir quando identifica uma instrução `idle in transaction` como um bloqueador.

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

Ação sugerida

Conforme indicado na coluna `suggestion`, conecte-se ao banco de dados em que a sessão ociosa na transação está presente e encerre a sessão usando a função `pg_terminate_backend()`. O usuário pode ser seu usuário administrador (conta mestra do RDS) ou um usuário com o privilégio `pg_terminate_backend()`.

> **Warning**
>
> Uma sessão encerrada reverterá (`ROLLBACK`) as alterações feitas. Dependendo de seus requisitos, pode ser necessário executar a instrução novamente. No entanto, é recomendável fazer isso somente após o processo do autovacuum terminar a operação de limpeza agressiva.

Transação preparada

O PostgreSQL permite transações que fazem parte de uma estratégia de confirmação bifásica chamada transações preparadas. Elas são habilitadas ao definir o parâmetro `max_prepared_transactions` como um valor diferente de zero. As transações preparadas são

projetadas para garantir que uma transação seja durável e permaneça disponível mesmo após falhas no banco de dados, reinicializações ou desconexões do cliente. Como as transações regulares, elas recebem um ID de transação e podem afetar o autovacuum. Se deixado em um estado preparado, o autovacuum não realiza o congelamento e isso pode levar à conclusão do ID da transação.

Quando as transações são deixadas no estado de preparação indefinidamente, sem serem resolvidas por um gerenciador de transações, elas se tornam transações preparadas órfãs. A única maneira de corrigir isso é confirmar ou reverter a transação usando os comandos `COMMIT PREPARED` ou `ROLLBACK PREPARED`, respectivamente.

> **Note**
>
> Esteja ciente de que um backup feito durante uma transação preparada ainda conterá essa transação após a restauração. Consulte as informações a seguir sobre como localizar e fechar essas transações.

A função `postgres_get_av_diag()` exibe a saída a seguir quando identifica um bloqueador que é uma transação preparada.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

Ação sugerida

Conforme mencionado na coluna de sugestões, conecte-se ao banco de dados em que a transação preparada está localizada. Com base na coluna `suggested_action`, analise cuidadosamente se deseja executar `COMMIT` ou `ROLLBACK`, e então execute a ação apropriada.

Para monitorar transações preparadas em geral, o PostgreSQL oferece uma visualização de catálogo chamada `pg_prepared_xacts`. Use a consulta a seguir para encontrar transações preparadas.

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

Slot de replicação lógica

O objetivo de um slot de replicação é manter as alterações não consumidas até que elas sejam replicadas em um servidor de destino. Consulte a página sobre replicação lógica do PostgreSQL para obter mais informações.

Existem dois tipos de slots de replicação lógica.

Slots de replicação lógica inativa

Quando a replicação é encerrada, os logs de transações não consumidos não podem ser removidos e o slot de replicação fica inativo. Embora um slot de replicação lógica inativo não seja usado atualmente por um assinante, ele permanece no servidor, levando à retenção de arquivos WAL e impedindo a remoção de logs de transações antigos. Isso pode aumentar o uso do disco e, especificamente, impedir que o autovacuum limpe as tabelas internas do catálogo, pois o sistema deve evitar que as informações do LSN sejam sobrescritas. Se não for resolvido, esse problema pode resultar em inchaço do catálogo, degradação do desempenho e aumento do risco de conclusão da limpeza, potencialmente causando tempo de inatividade das transações.

Slots de replicação lógica ativos, mas lentos

Às vezes, a remoção de tuplas inativas do catálogo é adiada devido à degradação do desempenho da replicação lógica. Esse atraso na replicação retarda a atualização do `catalog_xmin` e pode causar inchaço do catálogo e conclusão da limpeza.

A função `postgres_get_av_diag()` exibe uma saída semelhante à saída a seguir quando encontra um slot de replicação lógica como um bloqueador.

```
blocker               | Logical replication slot
```

```
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

Ação sugerida

Para resolver esse problema, verifique se há problemas com o esquema ou os dados de destino na configuração de replicação que possam estar encerrando o processo de aplicação. Os motivos mais comuns são:

- Colunas ausentes
- Tipos de dados incompatíveis
- Incompatibilidade de dados
- Tabela ausente

Se o problema estiver relacionado a questões de infraestrutura:

- Problemas de rede: Como resolver problemas com um banco de dados do Amazon RDS que está em um estado de rede incompatível?.
- O banco de dados ou a instância de banco de dados não está disponível devido aos seguintes motivos:
  - A instância da réplica está sem armazenamento: consulte Como resolver problemas que ocorrem quando as instâncias de banco de dados do Amazon RDS ficam sem armazenamento? para obter informações sobre como adicionar armazenamento.
  - Parâmetros incompatíveis: consulte Como corrijo uma instância de banco de dados do Amazon RDS que está presa no status de parâmetros incompatíveis? para obter mais informações sobre como resolver o problema.

Se sua instância estiver fora da rede da AWS ou no AWS EC2, consulte seu administrador sobre como resolver os problemas de disponibilidade ou problemas relacionados à infraestrutura.

Descartar o slot inativo

> **Warning**
>
> Cuidado: antes de descartar um slot de replicação, verifique cuidadosamente se ele não tem nenhuma replicação em andamento, está inativo e está em um estado irrecuperável. Descartar um slot prematuramente pode interromper a replicação ou causar perda de dados.

Depois de confirmar que o slot de replicação não é mais necessário, descarte-o para permitir que o autovacuum continue. A condição `active = 'f'` garante que somente um slot inativo seja descartado.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

Instâncias de leitor

Quando a configuração hot_standby_feedback está habilitada, ela impede que o autovacuum na instância de gravador remova linhas inativas que ainda podem ser necessárias para consultas executadas na instância de leitor. Esse comportamento é necessário porque as consultas executadas na instância de leitor (também aplicável às instâncias de leitor no Aurora Global Database) exigem que essas linhas permaneçam disponíveis na instância de gravador, evitando conflitos de consulta e cancelamentos.

> **Note**
>
> `hot_standby_feedback` está habilitado por padrão e não pode ser modificado no Aurora PostgreSQL.

A função `postgres_get_av_diag()` exibe uma saída semelhante à saída a seguir quando encontra uma réplica de leitura com slot de replicação física como um bloqueador.

```
blocker              | Oldest query running on aurora reader
database             | Not applicable
blocker_identifier   | my-aurora-reader-2
wait_event           | Not applicable
autovacuum_lagging_by | 540,122,859
```

```
suggestion            | Run the following query on the reader "my-aurora-reader-2" to
 find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 523476310;

                      | Review carefully and you may consider terminating the query on
 reader using suggested_action.
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 523476310;","

                      | [OR]


                      | ","Delete the reader if not needed"}
```

Conforme recomendado na coluna `suggested_action`, analise cuidadosamente essas opções para desbloquear o autovacuum.

- Encerrar a consulta: seguindo as orientações na coluna de sugestões, é possível se conectar à réplica de leitura, conforme especificado na coluna suggested_action. É recomendável analisar cuidadosamente a opção de encerrar a sessão. Se o encerramento for considerado seguro, use a função `pg_terminate_backend()` para encerrar a sessão. Essa ação pode ser executada por um administrador (como a conta mestra do RDS) ou por um usuário com o privilégio pg_terminate_backend() necessário.

  Execute o comando SQL a seguir na réplica de leitura para encerrar a consulta que está impedindo a limpeza das linhas antigas no primário. O valor de `backend_xmin` é reportado na saída da função:

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- Excluir a réplica de leitura se não for necessária: se a réplica de leitura não for mais necessária, você poderá excluí-la. Isso removerá os custos indiretos da replicação associada e permitirá que o primário recicle os logs de transação sem ser retido pela instância.

Tabelas temporárias

Tabelas temporárias, criadas usando a palavra-chave `TEMPORARY`, residem no esquema temporário, por exemplo pg_temp_xxx, e só podem ser acessadas pela sessão que as criou. As tabelas temporárias são removidas quando a sessão termina. No entanto, essas tabelas são invisíveis para o processo de autovacuum do PostgreSQL e devem ser limpas manualmente pela sessão que as criou. Tentar limpar a tabela temporária de outra sessão não tem efeito.

Em circunstâncias incomuns, uma tabela temporária pode existir sem uma sessão ativa que a possua. Se a sessão proprietária terminar inesperadamente devido a uma falha fatal, problemas de rede ou eventos do tipo, a tabela temporária pode não ser limpa, deixando-a como uma tabela "órfã". Quando o processo de autovacuum do PostgreSQL detecta uma tabela temporária órfã, ele registra em log a seguinte mensagem:

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

A função `postgres_get_av_diag()` exibe uma saída semelhante à saída a seguir quando identifica uma tabela temporária como um bloqueador. Para que a função exiba corretamente a saída relacionada às tabelas temporárias, ela precisa ser executada no mesmo banco de dados em que essas tabelas existem.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

Ação sugerida

Siga as instruções fornecidas na coluna `suggestion` da saída para identificar e remover a tabela temporária que está impedindo a execução do autovacuum. Use o comando a seguir para eliminar a tabela temporária reportada por `postgres_get_av_diag()`. Substitua o nome da tabela com base na saída fornecida pela função `postgres_get_av_diag()`.

```
DROP TABLE my_temp_schema.my_temp_table;
```

A seguinte consulta pode ser usada para identificar tabelas temporárias:

```
SELECT
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

Solucionar bloqueadores de limpeza não identificáveis no Aurora PostgreSQL

Esta seção explora razões adicionais que podem impedir o progresso da limpeza. Esses problemas atualmente não são diretamente identificáveis pela função `postgres_get_av_diag()`.

Tópicos

- Inconsistência do índice
- Taxa de transação excepcionalmente alta

Inconsistência do índice

Um índice logicamente inconsistente pode impedir que o autovacuum progrida. Os seguintes erros ou erros semelhantes são registrados em log durante a fase de limpeza do índice ou quando o índice é acessado por instruções SQL.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

Orientação

Reconstrua o índice ou pule índices usando `INDEX_CLEANUP` no manual `VACUUM FREEZE`.

- Usar a opção CONCURRENTLY: antes da versão 12 do PostgreSQL, a reconstrução de um índice exigia um bloqueio de tabela exclusivo, restringindo o acesso à tabela. Com a versão 12 e posteriores do PostgreSQL, a opção CONCURRENTLY permite o bloqueio em nível de linha, melhorando significativamente a disponibilidade da tabela. Este é o comando:

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  Embora CONCURRENTLY seja menos disruptivo, pode ser mais lento em tabelas acessadas com frequência. Considere construir o índice durante períodos de baixo tráfego, se possível. Para ter mais informações, consulte a documentação REINDEX do PostgreSQL.
- Usar a opção INDEX_CLEANUP FALSE: se os índices forem grandes e, segundo sua avaliação, exigirem muito tempo para serem concluídos, é possível desbloquear o autovacuum executando um VACUUM FREEZE manual ao excluir os índices. Essa funcionalidade está disponível na versão 12 e versões posteriores do PostgreSQL.

  Ignorar os índices permitirá que você pule o processo de limpeza do índice inconsistente e mitigue o problema de conclusão. No entanto, isso não resolverá o problema subjacente da página inválida. Para abordar e resolver totalmente o problema da página inválida, ainda será necessário reconstruir o índice.

Taxa de transação excepcionalmente alta

No PostgreSQL, altas taxas de transação podem afetar significativamente o desempenho do autovacuum, levando a uma limpeza mais lenta das tuplas inativas e ao aumento do risco de conclusão do ID da transação. É possível monitorar a taxa de transação medindo a diferença em `max(age(datfrozenxid))` entre dois períodos, normalmente por segundo. Além disso, é possível usar as métricas de contador do Insights de Performance do RDS a seguir para medir a taxa de transação (a soma de xact_commit e xact_rollback), que é o número total de transações.

Contador	Type	Unidade	Métrica
xact_commit	Transações	Confirmações por segundo	db.Transactions.xact_commit
xact_rollback	Transações	Reversões por segundo	db.Transactions.xact_rollback

Um aumento rápido indica uma alta carga de transações, que pode sobrecarregar o autovacuum, causando sobrecarga, contenção de bloqueios e possíveis problemas de desempenho. Isso pode impactar negativamente o processo de autovacuum de duas maneiras:

- Atividade da tabela: a tabela específica que está sendo limpa pode estar passando por um alto volume de transações, causando atrasos.
- Recursos do sistema: o sistema geral pode estar sobrecarregado, dificultando que o autovacuum acesse os recursos necessários para funcionar com eficiência.

Considere as seguintes estratégias para permitir que o autovacuum opere com mais eficiência e acompanhe suas tarefas:

1. Reduza a taxa de transação, se possível. Considere agrupar transações semelhantes sempre que possível.
2. Defina tabelas atualizadas com frequência com a operação `VACUUM FREEZE` manual noturna, semanal ou quinzenal durante horários de baixo movimento.
3. Considere aumentar a escala verticalmente da classe de instância para alocar mais recursos do sistema para lidar com o alto volume de transações e o autovacuum.

Resolver problemas de desempenho de limpeza no Aurora PostgreSQL

Esta seção discute os fatores que geralmente contribuem para um desempenho mais lento de limpeza e como resolvê-los.

Tópicos

- Limpeza em índices grandes
- Muitas tabelas ou bancos de dados para limpar
- A limpeza agressiva (para evitar conclusão) está em execução

Limpeza em índices grandes

O VACUUM opera em fases sequenciais: inicialização, verificação do heap, índice e limpeza do heap, limpeza de índices, truncamento de heap e limpeza final. Durante a verificação do heap, o processo remove, desfragmenta e congela páginas. Depois que o heap é verificado, o VACUUM limpa os índices, devolve páginas vazias ao sistema operacional e realiza tarefas finais de limpeza, como limpar o mapa de espaço livre e atualizar as estatísticas.

A limpeza de índice pode exigir várias passagens quando `maintenance_work_mem` (ou `autovacuum_work_mem`) é insuficiente para processar o índice. No PostgreSQL 16 e anterior, um limite de memória de 1 GB para armazenar IDs de tuplas mortas geralmente forçava várias passagens em índices grandes. O PostgreSQL 17 introduz o `TidStore`, que aloca memória dinamicamente em vez de usar uma matriz de alocação única. Isso remove a restrição de 1 GB, usa a memória com maior eficiência e reduz a necessidade de várias verificações para cada índice.

Índices grandes ainda podem exigir várias passagens no PostgreSQL 17 se a memória disponível não for suficiente para o processamento completo do índice de uma só vez. Normalmente, os índices maiores contêm mais tuplas mortas que exigem várias passagens.

Detectar operações de limpeza lentas

A função `postgres_get_av_diag()` pode detectar quando a execução de operações de limpeza está lenta devido a memória insuficiente. Para ter mais informações sobre essa função, consulte Instalar ferramentas de monitoramento e diagnóstico de autovacuum no Aurora PostgreSQL.

A função `postgres_get_av_diag()` emite os avisos a seguir quando a memória disponível não é suficiente para concluir a limpeza do índice em uma única passagem.

`rds_tools` 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
       memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
 TRUE) table_name;). For more information, see

       Working with PostgreSQL autovacuum in the Amazon Aurora User Guide.
```

> **Note**
>
> A função `postgres_get_av_diag()` depende do `pg_stat_all_tables.n_dead_tup` para estimar a quantidade de memória necessária para a limpeza do índice.

Quando a função `postgres_get_av_diag()` identificar uma operação de limpeza lenta que requer várias verificações do índice devido a `autovacuum_work_mem` insuficiente, ela gerará a seguinte mensagem:

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
       For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

Orientação

Você pode aplicar as soluções alternativas a seguir usando o `VACUUM FREEZE` manual para acelerar o congelamento da tabela.

Aumentar a memória para limpeza

Conforme sugerido pela função `postgres_get_av_diag()`, é recomendável aumentar o parâmetro `autovacuum_work_mem` para lidar com possíveis restrições de memória no nível da instância. Embora `autovacuum_work_mem` seja um parâmetro dinâmico, é importante observar que, para que a nova configuração de memória entre em vigor, o daemon autovacuum precisa reiniciar seus processos. Para fazer isso:

1. Confirme se a nova configuração está em vigor.
2. Encerre os processos que estão executando o autovacuum.

Essa abordagem garante que a alocação de memória ajustada seja aplicada às novas operações do autovacuum.

Para obter resultados mais imediatos, considere realizar manualmente uma operação `VACUUM FREEZE` com uma configuração `maintenance_work_mem` maior em sua sessão:

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Se você estiver usando o Amazon RDS e notar que precisa de memória adicional para oferecer suporte a valores mais altos para `maintenance_work_mem` ou `autovacuum_work_mem`, considere fazer upgrade para uma classe de instância com mais memória. Isso pode fornecer os recursos necessários para aprimorar as operações de limpeza manuais e automáticas, levando a um melhor desempenho geral da limpeza e do banco de dados.

Desativar INDEX_CLEANUP

O VACUUM manual no PostgreSQL versão 12 e posteriores permite pular a fase de limpeza do índice, enquanto o autovacuum de emergência no PostgreSQL versão 14 e posteriores faz isso automaticamente com base no parâmetro `vacuum_failsafe_age`.

> **Warning**
>
> Pular a etapa de limpeza do índice pode causar inchaço no índice e afetar negativamente o desempenho da consulta. Para mitigar isso, considere reindexar ou fazer limpeza nos índices afetados durante uma janela de manutenção.

Para obter orientação adicional sobre como lidar com índices grandes, consulte a documentação em Gerenciar o autovacuum com grandes índices.

Limpeza paralelo de índices

Desde o PostgreSQL 13, os índices podem ser limpos em paralelo por padrão usando o VACUUM manual, com um processo de operação de limpeza atribuído a cada índice. No entanto, para que o PostgreSQL determine se uma operação de limpeza se qualifica para execução paralela, critérios específicos devem ser atendidos:

- Deve haver pelo menos dois índices.
- O parâmetro `max_parallel_maintenance_workers` deve ser definido como no mínimo 2.
- O tamanho do índice deve exceder o limite `min_parallel_index_scan_size`, que por padrão é 512 KB.

É possível ajustar a configuração `max_parallel_maintenance_workers` com base no número de vCPUs disponíveis na sua instância do Amazon RDS e no número de índices na tabela para otimizar o tempo de resposta da limpeza.

Para obter mais informações, consulte Parallel vacuuming in Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL.

Muitas tabelas ou bancos de dados para limpar

Conforme mencionado na documentação The Autovacuum Daemon do PostgreSQL, o daemon autovacuum é executado por meio de vários processos. Isso inclui um inicializador persistente de

autovacuum, responsável por iniciar os processos de trabalho de autovacuum para cada banco de dados do sistema. O inicializador programa esses trabalhadores para iniciarem aproximadamente a cada `autovacuum_naptime` segundos por banco de dados.

Com 'N' bancos de dados, um novo trabalhador começa aproximadamente a cada [`autovacuum_naptime`/N segundos]. No entanto, o número total de trabalhadores simultâneos é limitado pela configuração `autovacuum_max_workers`. Se o número de bancos de dados ou tabelas que precisam ser limpos exceder esse limite, o próximo banco de dados ou tabela será processado assim que um trabalhador estiver disponível.

Quando muitas tabelas ou bancos de dados grandes precisam ser limpos simultaneamente, todos os trabalhadores de autovacuum disponíveis podem ficar ocupados por um longo período, atrasando a manutenção em outras tabelas e bancos de dados. Em ambientes com altas taxas de transação, esse gargalo pode aumentar rapidamente e potencialmente levar a problemas de conclusão de limpeza em sua instância do Amazon RDS.

Quando o `postgres_get_av_diag()` detecta um grande número de tabelas ou bancos de dados, ele fornece a seguinte recomendação:

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

Orientação

Aumentar autovacuum_max_workers

Para agilizar a limpeza, recomendamos ajustar o parâmetro `autovacuum_max_workers` para permitir mais trabalhadores simultâneos de autovacuum. Se os gargalos de desempenho persistirem, considere aumentar a escala verticalmente de sua instância do Amazon RDS para uma classe com mais vCPUs, o que pode melhorar ainda mais as capacidades de processamento paralelo.

A limpeza agressiva (para evitar conclusão) está em execução

A idade do banco de dados (MaximumUsedTransactionIDs) no PostgreSQL só diminui quando uma limpeza agressiva (para evitar conclusão) é concluída com sucesso. Até que essa limpeza termine, a idade continuará aumentando dependendo da taxa de transação.

A função `postgres_get_av_diag()` gera o seguinte `NOTICE` quando detecta uma limpeza agressiva. No entanto, ela só aciona essa saída depois que a limpeza estiver ativa por pelo menos dois minutos.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Consulte When an aggressive vacuum is already running para obter mais informações sobre a limpeza agressiva.

Use a seguinte consulta para verificar se uma limpeza agressiva está em andamento:

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
```

```
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

É possível determinar se é uma limpeza agressiva (para evitar conclusão) verificando a coluna de consulta na saída. A frase "para evitar conclusão" indica que se trata de uma limpeza agressiva.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

Por exemplo, suponha que você tenha um bloqueador com idade de transação de 1 bilhão e uma tabela exigindo uma limpeza agressiva para evitar conclusão na mesma idade da transação. Além disso, há outro bloqueador com idade de transação de 750 milhões. Depois de eliminar o bloqueador com idade de transação de 1 bilhão, a idade de transação não cairá imediatamente para 750 milhões. Ela permanecerá alta até que a tabela que precise da limpeza agressiva ou qualquer transação com idade superior a 750 milhões seja concluída. Durante esse período, a idade das transações do cluster do PostgreSQL continuará aumentando. Quando o processo de limpeza for concluído, a idade da transação cairá para 750 milhões, mas começará a aumentar novamente até que a limpeza adicional seja concluída. Esse ciclo continuará enquanto essas condições persistirem, até que a idade da transação caia para o nível configurado na instância do Amazon RDS, especificado por `autovacuum_freeze_max_age`.

Explicação das mensagens de AVISO no Aurora PostgreSQL

A função `postgres_get_av_diag()` fornece as seguintes mensagens de AVISO:

Quando a idade ainda não atingiu o limite de monitoramento

O limite de monitoramento de `postgres_get_av_diag()` para identificar bloqueadores é de 500 milhões de transações por padrão. Se `postgres_get_av_diag()` gerar o seguinte AVISO, isso indica que a idade da transação ainda não atingiu esse limite.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

Não conectado ao banco de dados com o ID de transação mais antigo

A função `postgres_get_av_diag()` fornece a saída mais precisa quando conectada ao banco de dados com a idade de ID de transação mais antiga. O banco de dados com a idade de ID de transação mais antiga informada por `postgres_get_av_diag()` será diferente

de “my_database” no seu caso. Se você não estiver conectado ao banco de dados correto, o seguinte AVISO será gerado:

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

Conectar-se ao banco de dados com a idade de transação mais antiga é importante pelos seguintes motivos:

- Identificação de bloqueadores de tabelas temporárias: como os metadados das tabelas temporárias são específicos de cada banco de dados, eles geralmente são encontrados no banco de dados em que foram criados. No entanto, se uma tabela temporária for o principal bloqueador e residir no banco de dados com a transação mais antiga, isso pode ser enganoso. A conexão com o banco de dados correto garante a identificação precisa do bloqueador de tabelas temporárias.
- Diagnóstico de vacuums lentos: os metadados do índice e as informações de contagem de tabelas são específicos do banco de dados e necessários para diagnosticar problemas de vacuum lento.

O banco de dados com a transação mais antiga por idade está em um banco de dados rdsadmin ou template0

Em alguns casos, os bancos de dados `rdsadmin` ou `template0` podem ser identificados como o banco de dados com a idade de ID de transação mais antiga. Se isso acontecer, `postgres_get_av_diag()` emitirá o seguinte AVISO:

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

Verifique se o bloqueador listado não é originário de nenhum desses dois bancos de dados. Se for relatado que o bloqueador está presente em `rdsadmin` ou `template0`, entre em contato com o suporte, pois esses bancos de dados não são acessíveis ao usuário e exigem intervenção.

É altamente improvável que o banco de dados `rdsadmin` ou `template0` contenha um bloqueador principal.

Quando um vacuum agressivo já está em execução

A função `postgres_get_av_diag()` foi projetada para relatar quando um processo de vacuum agressivo está em execução, mas ela só vai acionar essa saída depois que o vacuum estiver

ativo por pelo menos 1 minuto. Esse atraso intencional ajuda a reduzir as chances de falsos positivos. Ao esperar, a função garante que somente vacuums efetivos e significativos sejam relatados, levando a um monitoramento mais preciso e confiável da atividade do vacuum.

A função `postgres_get_av_diag()` gera o seguinte AVISO quando detecta um ou mais vacuums agressivos em andamento.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Conforme indicado no AVISO, continue monitorando o desempenho do vacuum. Para obter mais informações sobre o vacuum agressivo, consulte A limpeza agressiva (para evitar conclusão) está em execução

Quando o autovacuum está desativado

A função `postgres_get_av_diag()` vai gerar o seguinte AVISO se o autovacuum estiver desativado na instância de banco de dados:

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

O autovacuum é um recurso essencial da instância de banco de dados do Aurora PostgreSQL que garante uma operação tranquila do banco de dados. Ele remove automaticamente as versões antigas das linhas, recupera espaço de armazenamento e evita o inchaço das tabelas, ajudando a manter as tabelas e os índices eficientes para um desempenho ideal. Além disso, ele protege contra o wraparound de ID de transação, que pode interromper transações na instância do Amazon RDS. A desativação do autovacuum pode levar a quedas de longo prazo no desempenho e na estabilidade do banco de dados. Sugerimos que você o mantenha ativado sempre. Para ter mais informações, consulte Understanding autovacuum in Aurora PostgreSQL environments.

> **Note**
>
> O desligamento do autovacuum não impede os vacuums agressivos. Estes ainda ocorrerão quando as tabelas atingirem o limite de `autovacuum_freeze_max_age`.

O número de transações restantes é criticamente baixo

A função `postgres_get_av_diag()` gera o seguinte AVISO quando um vacuum wraparound é iminente. Este AVISO é emitido quando a instância do Amazon RDS está a 100 milhões de transações de possivelmente rejeitar novas transações.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

Sua ação imediata é necessária para evitar tempo de inatividade do banco de dados. Você deve monitorar de perto suas operações de vacuum e considerar iniciar manualmente um `VACUUM FREEZE` no banco de dados afetado para evitar falhas nas transações.

Gerenciar a contenção de TOAST OID no Amazon Aurora PostgreSQL

TOAST (The Oversized-Attribute Storage Technique) é um recurso do PostgreSQL projetado para lidar com grandes valores de dados que excedem o tamanho normal de blocos de banco de dados de 8 KB. O PostgreSQL não permite que linhas físicas abranjam vários blocos. O tamanho do bloco atua como um limite superior no tamanho da linha. O TOAST supera essa restrição dividindo grandes valores de campo em partes menores. Ele os armazena separadamente em uma tabela TOAST dedicada vinculada à tabela principal. Para acessar mais informações, consulte a documentação de implementação e mecanismo de armazenamento TOAST do PostgreSQL.

Tópicos

- Noções básicas sobre operações TOAST
- Identificar problemas de performance
- Recomendações
- Monitoramento

Noções básicas sobre operações TOAST

O TOAST executa a compactação e armazena grandes valores de campo fora da linha. O recurso atribui um OID (Identificador de Objeto) exclusivo a cada bloco de dados superdimensionados

armazenado na tabela TOAST. A tabela principal armazena o ID do valor TOAST e o ID da relação na página para fazer referência à linha correspondente na tabela TOAST. Isso permite que o PostgreSQL localize e gerencie com eficiência esses fragmentos do TOAST. No entanto, à medida que a tabela TOAST cresce, o sistema corre o risco de esgotar os OIDs disponíveis, causando a degradação da performance e um possível tempo de inatividade devido ao esgotamento do OID.

Identificadores de objetos no TOAST

Identificador de Objeto (OID) é um identificador exclusivo em todo o sistema usado pelo PostgreSQL para fazer referência a objetos de banco de dados, como tabelas, índices e funções. Esses identificadores desempenham um papel essencial nas operações internas do PostgreSQL, permitindo que o banco de dados localize e gerencie objetos com eficiência.

Para tabelas com conjuntos de dados elegíveis para toasting, o PostgreSQL atribui OIDs para identificar exclusivamente cada fragmento de dado excedente armazenado na tabela TOAST associada. O sistema associa cada fragmento a um `chunk_id`, o que ajuda o PostgreSQL a organizar e localizar esses fragmentos de forma eficiente na tabela TOAST.

Identificar problemas de performance

O gerenciamento de OID do PostgreSQL depende de um contador global de 32 bits para que ele funcione depois de gerar 4 bilhões de valores exclusivos. Embora o cluster de banco de dados compartilhe esse contador, a alocação de OID envolve duas etapas durante as operações TOAST:

- Contador global para alocação: o contador global atribui um novo OID em todo o cluster.
- Pesquisa local de conflitos: a tabela TOAST garante que o novo OID não entre em conflito com os OIDs existentes já usados nessa tabela específica.

A degradação da performance pode ocorrer quando:

- A tabela TOAST tem alta fragmentação ou uso denso de OID, causando atrasos na atribuição do OID.
- O sistema frequentemente aloca e reutiliza OIDs em ambientes com alta rotatividade de dados ou tabelas amplas que usam o TOAST extensivamente.

Para acessar mais informações, consulte a documentação de alocação de OID e limites de tamanho de tabelas TOAST do PostgreSQL:

Um contador global gera os OIDs e envolve cada 4 bilhões de valores, para que, periodicamente, o sistema gere novamente um valor já usado. O PostgreSQL detecta isso e tenta novamente com o próximo OID. Um INSERT lento poderá ocorrer se houver uma série muito longa de valores de OID usados sem lacunas na tabela TOAST. Esses desafios se tornam mais pronunciados à medida que o espaço do OID é preenchido, causando inserções e atualizações mais lentas.

Identificar o problema

- Declarações `INSERT` simples demoram muito mais do que o normal de maneira inconsistente e aleatória.
- Atrasos ocorrem somente para declarações `INSERT` e `UPDATE` envolvendo operações TOAST.
- As seguintes entradas de log aparecem nos logs do PostgreSQL quando o sistema tem dificuldade para encontrar OIDs disponíveis nas tabelas TOAST:

```
LOG: still searching for an unused OID in relation "pg_toast_20815"
DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been
 found yet.
```

- O Insights de Performance indica um alto número médio de sessões ativas (AAS) associadas a eventos de espera `LWLock:buffer_io` e `LWLock:OidGenLock`.

  Você pode executar a seguinte consulta SQL para identificar transações INSERT de longa duração com eventos de espera:

```
SELECT
    datname AS database_name,
    usename AS database_user,
    pid,
    now() - pg_stat_activity.xact_start AS transaction_duration,
    concat(wait_event_type, ':', wait_event) AS wait_event,
    substr(query, 1, 30) AS TRANSACTION,
    state
FROM
    pg_stat_activity
WHERE (now() - pg_stat_activity.xact_start) > INTERVAL '60 seconds'
    AND state IN ('active', 'idle in transaction', 'idle in transaction (aborted)',
 'fastpath function call', 'disabled')
    AND pid <> pg_backend_pid()
AND lower(query) LIKE '%insert%'
ORDER BY
```

```
    transaction_duration DESC;
```

Exemplos de resultado de consulta exibindo operações INSERT com tempos de espera estendidos:

```
 database_name |  database_user  |  pid  | transaction_duration |     wait_event
 |          transaction           | state
---------------+-----------------+-------+----------------------
+---------------------+---------------------------------+--------
 postgres       | db_admin_user| 70965 | 00:10:19.484061      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 69878 | 00:06:14.976037      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 68937 | 00:05:13.942847      | :                   |
 INSERT INTO "products" (......... | active
```

Isolar o problema

- Testar a inserção pequena: insira um registro menor que o limite `toast_tuple_target`. Lembre-se de que a compactação é aplicada antes do armazenamento TOAST. Se isso funcionar sem problemas de performance, o problema está relacionado às operações TOAST.
- Testar nova tabela: crie uma tabela com a mesma estrutura e insira um registro maior que `toast_tuple_target`. Se isso funcionar sem problemas, o problema estará localizado na alocação de OID da tabela original.

Recomendações

As abordagens a seguir podem ajudar a resolver problemas de contenção do TOAST OID.

- Limpeza e arquivamento de dados: analise e exclua quaisquer dados obsoletos ou desnecessários para liberar OIDs para uso futuro ou arquivar os dados. Considere as seguintes limitações:
  - Escalabilidade limitada, pois a limpeza futura nem sempre é possível.
  - Possível operação VACUUM de longa duração para remover as tuplas inativas resultantes.
- Gravar em uma nova tabela: crie uma tabela para futuras inserções e use uma visualização `UNION ALL` para combinar dados antigos e novos para consultas. Essa visualização apresenta os dados combinados das tabelas antigas e novas, permitindo que as consultas os acessem como uma única tabela. Considere as seguintes limitações:

  - As atualizações na tabela antiga ainda podem causar o esgotamento de OIDs.
- Partição ou fragmento: particione a tabela ou fragmente os dados para ter melhor escalabilidade e performance. Considere as seguintes limitações:
  - Maior complexidade na lógica de consulta e na manutenção, possível necessidade de alterações na aplicação para lidar corretamente com dados particionados.

Monitoramento

Usar tabelas do sistema

Você pode usar as tabelas do sistema do PostgreSQL para monitorar o crescimento do uso de OIDs.

> **Warning**
>
> Dependendo do número de OIDs na tabela TOAST, pode levar algum tempo para ser concluído. Recomendamos que você agende o monitoramento fora do horário comercial para minimizar o impacto.

O bloco anônimo a seguir conta o número de OIDs distintos usados em cada tabela TOAST e exibe as informações da tabela principal:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
```

```
            -- Fetch the number of distinct used OIDs (chunk IDs) from the TOAST table
            EXECUTE 'SELECT COUNT(DISTINCT chunk_id) FROM ' || r.toast_table INTO o;
            -- If there are used OIDs, find the associated parent table and its schema
            IF o <> 0 THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table: %
 | Number of used OIDs: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Exemplo de saída exibindo estatísticas de uso de OIDs pela tabela TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Number of used OIDs: 45,623,317
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Number of used OIDs: 10,000
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Number of used OIDs: 1,000,000
DO
```

O bloco anônimo a seguir recupera o OID máximo atribuído para cada tabela TOAST não vazia:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
```

```
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the max(chunk_id) from the TOAST table
            EXECUTE 'SELECT max(chunk_id) FROM ' || r.toast_table INTO o;
            -- If there's at least one TOASTed chunk, find the associated parent table
 and its schema
            IF o IS NOT NULL THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table:
 % | Max chunk_id: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Exemplo de saída exibindo o máximo de IDs de blocos para tabelas TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Max chunk_id: 45,639,907
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Max chunk_id: 45,649,929
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Max chunk_id: 46,649,935
DO
```

Usar o Performance Insights

Os eventos de espera `LWLock:buffer_io` e `LWLock:OidGenLock` aparecem no Insights de Performance durante operações que exigem a atribuição de novos identificadores de objetos (OIDs). As sessões ativas de alta média (AAS) para esses eventos geralmente apontam para contenção durante a atribuição de OIDs e o gerenciamento de recursos. Isso é particularmente comum em ambientes com alta rotatividade de dados, uso extensivo de grandes volumes de dados ou criação frequente de objetos.

LWLock:buffer_io

`LWLock:buffer_io` é um evento de espera que ocorre quando uma sessão do PostgreSQL aguarda a conclusão das operações de E/S em um buffer compartilhado. Isso geralmente acontece quando o banco de dados lê dados do disco para a memória ou grava páginas modificadas da memória para o disco. O evento de espera `BufferIO` garante a consistência ao impedir que vários processos acessem ou modifiquem o mesmo buffer enquanto as operações de E/S estão em andamento. Altas ocorrências desse evento de espera podem indicar gargalos no disco ou atividade excessiva de E/S na workload do banco de dados.

Durante as operações TOAST:

- O PostgreSQL aloca OIDs para objetos grandes e garante sua exclusividade examinando o índice da tabela TOAST.
- Índices TOAST grandes podem exigir o acesso a várias páginas para verificar a exclusividade do OID. Isso gera aumento de E/S de disco, principalmente quando o grupo de buffers não consegue armazenar em cache todas as páginas necessárias.

O tamanho do índice afeta diretamente o número de páginas de buffer que precisam ser acessadas durante essas operações. Mesmo que o índice não esteja inchado, seu tamanho pode aumentar a E/S do buffer, principalmente em ambientes de alta simultaneidade ou alta rotatividade. Para acessar mais informações, consulte o Guia de solução de problemas com o evento de espera LWLock:BufferIO.

LWLock:OidGenLock

`OidGenLock` é um evento de espera que ocorre quando uma sessão do PostgreSQL está aguardando para alocar um novo identificador de objeto (OID). Esse bloqueio garante que os OIDs sejam gerados sequencialmente e com segurança, permitindo que somente um processo gere OIDs por vez.

Durante as operações TOAST:

- Alocação de OID para blocos na tabela TOAST: o PostgreSQL atribui OIDs a blocos nas tabelas TOAST ao gerenciar grandes registros de dados. Cada OID deve ser exclusivo para evitar conflitos no catálogo do sistema.
- Alta simultaneidade: como o acesso ao gerador de OIDs é sequencial, quando várias sessões estão criando simultaneamente objetos que exigem OIDs, pode ocorrer contenção para `OidGenLock`. Isso aumenta a probabilidade de sessões esperando a conclusão da alocação de OIDs.
- Dependência do acesso ao catálogo do sistema: a alocação de OIDs requer atualizações nas tabelas compartilhadas do catálogo do sistema, como `pg_class` e `pg_type`. Se essas tabelas apresentarem muita atividade (devido às operações frequentes de DDL), isso poderá aumentar a contenção de bloqueio para `OidGenLock`.
- Alta demanda de alocação de OIDs: workloads intensas envolvendo TOAST com grandes registros de dados exigem alocação constante de OIDs, aumentando a contenção.

Fatores adicionais que aumentam a contenção de OIDs:

- Criação frequente de objetos: workloads trabalho que frequentemente criam e descartam objetos, como tabelas temporárias, amplificam a contenção no contador global de OIDs.
- Bloqueio global do contador: o contador global de OIDs é acessado em série para garantir a exclusividade, criando um único ponto de contenção em ambientes de alta simultaneidade.

Usar o Babelfish para Aurora PostgreSQL

O Babelfish para Aurora PostgreSQL amplia seu cluster de banco de dados do Aurora PostgreSQL com a capacidade de aceitar conexões de banco de dados provenientes de clientes do SQL Server. Com o Babelfish, as aplicações originalmente criadas para o SQL Server podem funcionar diretamente com o Aurora PostgreSQL com poucas alterações de código em comparação com uma migração tradicional e sem alterar drivers de banco de dados. Para obter mais informações sobre a migração, consulte Migrar um banco de dados do SQL Server para o Babelfish para Aurora PostgreSQL.

O Babelfish fornece um endpoint adicional para um cluster de bancos de dados Aurora PostgreSQL com o qual ele pode compreender o protocolo em nível de transmissão do SQL Server e as instruções do SQL Server comumente utilizadas. As aplicações cliente que usam o protocolo de transmissão Tabular Data Stream (TDS) podem se conectar nativamente à porta do ouvinte TDS no Aurora PostgreSQL. Para saber mais sobre o TDS, consulte [MS-TDS]: Tabular Data Stream Protocol no site da Microsoft.

> **Note**
>
> O Babelfish para Aurora PostgreSQL é compatível com as versões 7.1 a 7.4 do TDS.

O Babelfish também fornece acesso aos dados usando a conexão do PostgreSQL. Por padrão, os dois dialetos SQL compatíveis com o Babelfish estão disponíveis por meio de seus protocolos nativos nas seguintes portas:

- Dialeto do SQL Server (T-SQL), os clientes conectam-se à porta 1433.
- Dialeto PostgreSQL (PL/pgSQL), os clientes conectam-se à porta 5432.

O Babelfish executa a linguagem Transact-SQL (T-SQL) com algumas diferenças. Para obter mais informações, consulte Diferenças entre o Babelfish para Aurora PostgreSQL e o SQL Server.

Nas seções a seguir, você encontra informações sobre como configurar e utilizar um cluster de banco de dados do Babelfish para Aurora PostgreSQL.

Tópicos

- Limitações do Babelfish
- Noções básicas da arquitetura e da configuração do Babelfish

- Criar um cluster de banco de dados do Babelfish para Aurora PostgreSQL
- Migrar um banco de dados do SQL Server para o Babelfish para Aurora PostgreSQL
- Autenticação de banco de dados com o Babelfish para Aurora PostgreSQL
- Conectar-se a um cluster de banco de dados do Babelfish
- Trabalhar com o Babelfish
- Solução de problemas do Babelfish
- Desativar o Babelfish
- Gerenciar as atualizações da versão do Babelfish para Aurora PostgreSQL
- Referência do Babelfish para Aurora PostgreSQL

Limitações do Babelfish

No momento, as seguintes limitações se aplicam ao Babelfish para Aurora PostgreSQL:

- O Babelfish não comporta os seguintes recursos do Aurora:
  - AWS Identity and Access Management
  - Fluxos de atividades do banco de dados (DAS)
  - API de dados do RDS com o Aurora PostgreSQL Sem Servidor v2 e provisionado
  - RDS Proxy com RDS para SQL Server
  - Salted challenge response authentication mechanism (SCRAM)
  - Editor de consultas
  - Integrações ETL zero
- O Babelfish não oferece suporte à seguinte API de driver de cliente:
  - Não há suporte para solicitações de API com os atributos de conexão relacionados ao Microsoft Distributed Transaction Coordinator (MSDTC). Isso inclui chamadas XA pela classe SQLServerXAResource no driver JDBC do servidor SQL.
- No momento, o Babelfish não é compatível com os seguintes recursos do Aurora PostgreSQL:
  - `bloom`
  - `btree_gin`
  - `btree_gist`
  - `citext`
  - `cube`
  - `hstore`
  - `hypopg`
  - Replicação lógica usando o `pglogical`
  - `ltree`
  - `pgcrypto`
  - Gerenciamento de planos de consultas usando `apg_plan_mgmt`

  Para saber mais sobre as extensões do PostgreSQL, consulte Trabalhar com extensões e invólucros de dados externos.
- O driver jTDS de código aberto projetado como uma alternativa ao driver Microsoft JDBC não é compatível.

Noções básicas da arquitetura e da configuração do Babelfish

Você gerencia o cluster de banco de dados de edição compatível com Aurora PostgreSQL que executa o Babelfish da mesma forma que faria com qualquer cluster de banco de dados do Aurora. Ou seja, você se beneficia da escalabilidade, alta disponibilidade com compatibilidade com failover e replicação integrada fornecida por um cluster de banco de dados do Aurora. Para saber mais sobre esses recursos, consulte Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora, Alta disponibilidade do Amazon Aurora e Replicação com o Amazon Aurora. Você também tem acesso a muitas outras ferramentas e utilitários da AWS, inclusive o seguinte:

- O Amazon CloudWatch é um serviço de monitoramento e capacidade de observação que fornece dados e informações úteis. Para ter mais informações, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.
- O Performance Insights é um recurso de ajuste e monitoramento de performance do banco de dados que ajuda você a avaliar rapidamente a carga em seu banco de dados. Para saber mais, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.
- Os bancos de dados globais do Aurora abrangem várias Regiões da AWS, permitindo leituras globais de baixa latência e fornecendo recuperação rápida de qualquer interrupção que possa afetar toda uma Região da AWS. Para ter mais informações, consulte Usar o Amazon Aurora Global Database.
- A aplicação automática de patches de software mantém seu banco de dados atualizado com os patches de segurança e os recursos mais recentes quando estes se tornam disponíveis.
- Os eventos do Amazon RDS notificam você por e-mail ou mensagem SMS sobre eventos importantes do banco de dados, como um failover automatizado. Para ter mais informações, consulte Monitorar eventos do Amazon Aurora.

A seguir, você pode aprender sobre a arquitetura do Babelfish e como os bancos de dados do SQL Server migrados são processados pelo Babelfish. Ao criar seu cluster de banco de dados do Babelfish, você precisa tomar algumas decisões antecipadamente sobre um único ou vários bancos de dados, agrupamentos e outros detalhes.

Tópicos

- Arquitetura do Babelfish
- Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish
- Noções básicas sobre agrupamentos no Babelfish para Aurora PostgreSQL

- Gerenciar o tratamento de erros do Babelfish com hatches de escape

Arquitetura do Babelfish

Quando você cria um cluster do Aurora PostgreSQL com o Babelfish habilitado, o Aurora provisiona o cluster com um banco de dados PostgreSQL denominado `babelfish_db`. Esse banco de dados é o local onde residem todos os objetos e estruturas do SQL Server migrados.

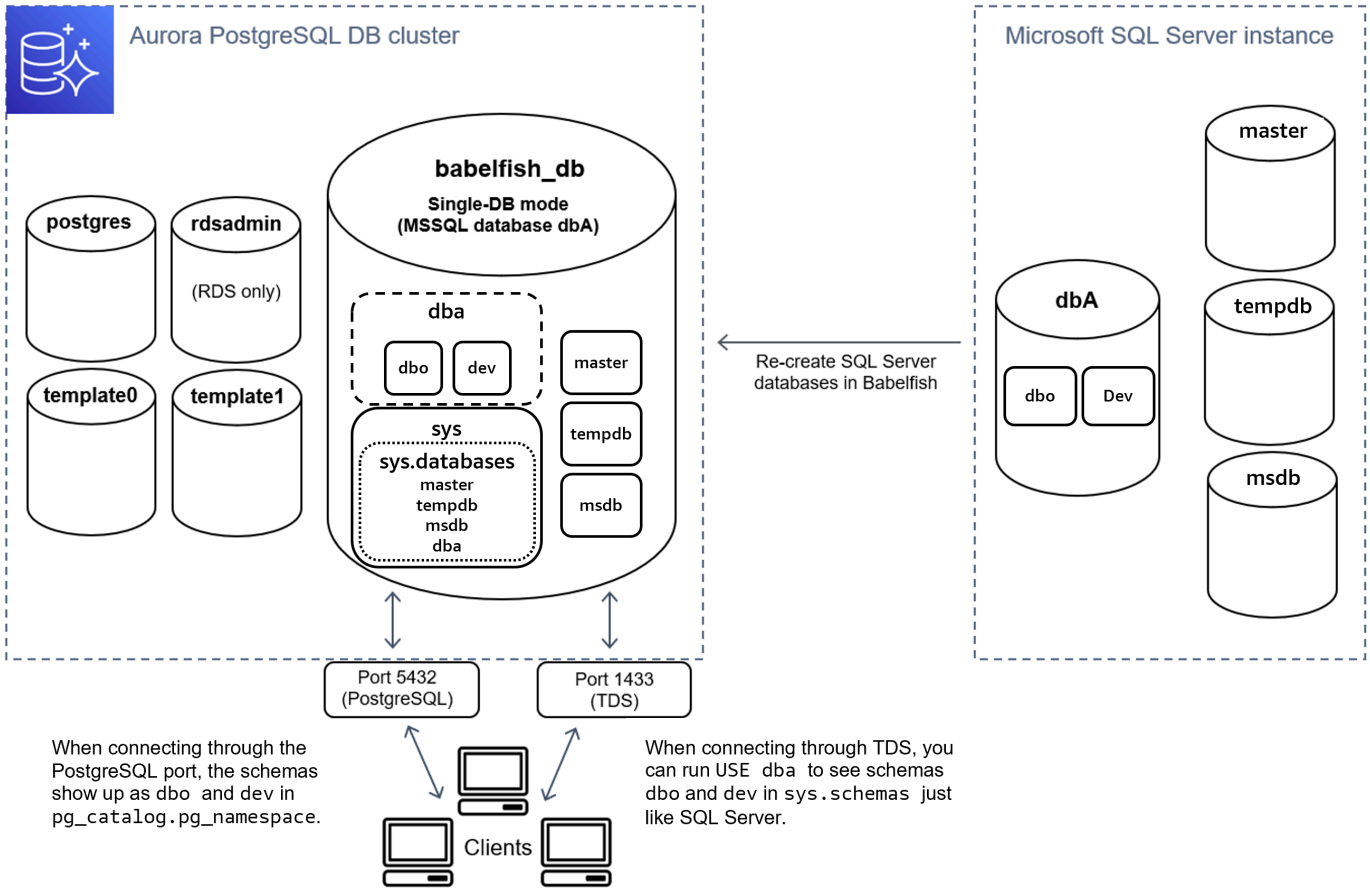
> **Note**
>
> Em um cluster do Aurora PostgreSQL, o nome do banco de dados `babelfish_db` é reservado para Babelfish. A criação de seu próprio banco de dados "babelfish_db" em um cluster de banco de dados do Babelfish impede o Aurora de provisionar com êxito o Babelfish.

Quando você se conecta à porta do TDS, a sessão é colocada no banco de dados `babelfish_db`. No T-SQL, a estrutura parece semelhante a uma conexão com uma instância do SQL Server. É possível ver os bancos de dados `master`, `msdb` e `tempdb` e o catálogo `sys.databases`. É possível criar bancos de dados de usuário adicionais e alternar entre eles com a instrução USE. Quando você cria um banco de dados de usuários do SQL Server, ele é nivelado no banco de dados PostgreSQL `babelfish_db`. Seu banco de dados retém a sintaxe entre bancos de dados e uma semântica igual ou semelhante àquelas fornecidas pelo SQL Server.

Utilizar o Babelfish com um único banco de dados ou vários bancos de dados

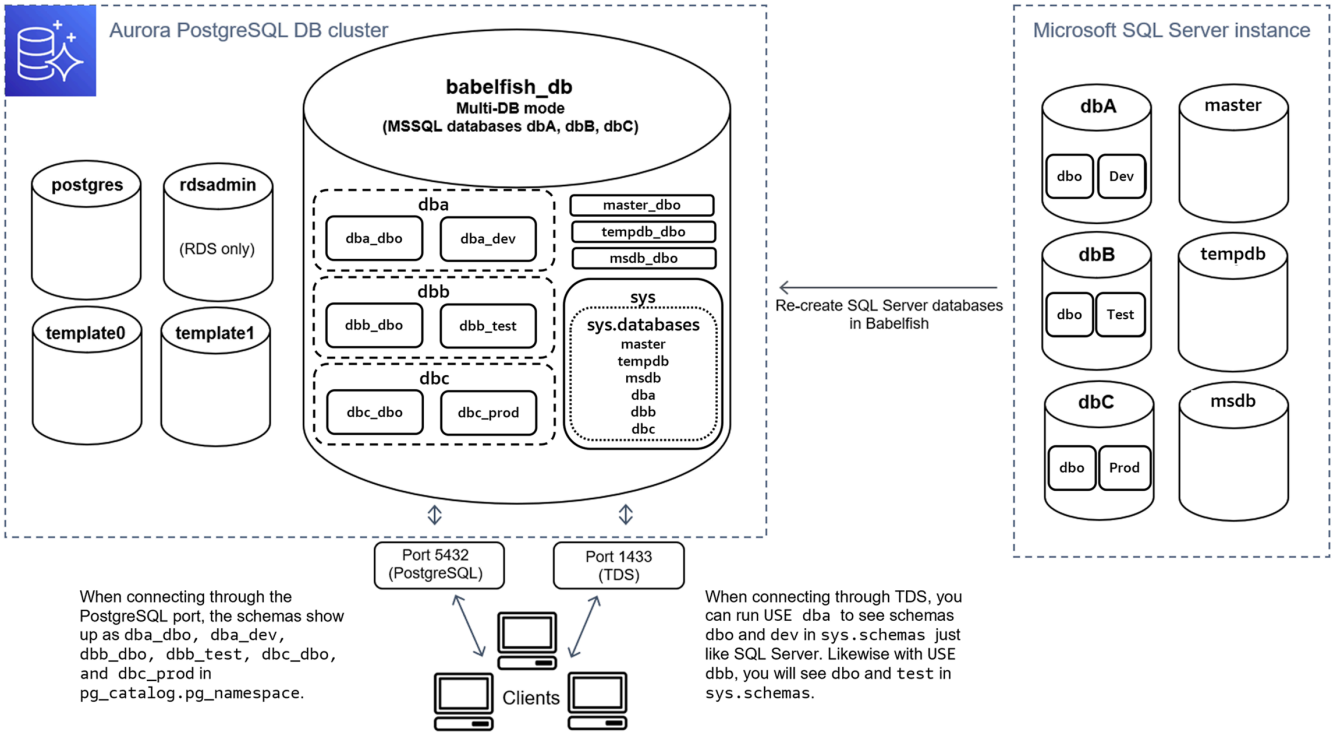
Ao criar um cluster do Aurora PostgreSQL para uso com o Babelfish, você escolhe entre utilizar um único banco de dados SQL Server em seus próprios bancos de dados SQL Server ou em vários bancos de dados SQL Server juntos. Sua escolha afeta como os nomes dos esquemas do SQL Server dentro do banco de dados `babelfish_db` aparecem no Aurora PostgreSQL. O modo de migração é armazenado no parâmetro `migration_mode`. Você não deve alterar esse parâmetro depois de criar seu cluster, pois pode perder o acesso a todos os objetos SQL criados anteriormente.

No modo single-db, os nomes do esquema do banco de dados SQL Server permanecem os mesmos no banco de dados `babelfish_db` do PostgreSQL. Se você optar por migrar somente um único banco de dados, os nomes dos esquemas do banco de dados de usuário migrado poderão ser referenciados no PostgreSQL com os mesmos nomes usados no SQL Server. Por exemplo, os esquemas `smith` e `dbo` e residem no banco de dados `dbA` .



Ao se conectar via TDS, é possível executar `USE dba` para ver esquemas `dbo` e `dev` do T-SQL, como você faria no SQL Server. Os nomes de esquema inalterados também são visíveis no PostgreSQL.

No modo de vários bancos de dados, os nomes dos esquemas dos bancos de dados de usuários se tornam `dbname_schemaname` quando acessados do PostgreSQL. Os nomes dos esquemas permanecem os mesmos quando acessados do T-SQL.



Como mostrado na imagem, o modo de vários bancos de dados e o modo de banco de dados único são os mesmos que do SQL Server ao se conectar pela porta TDS e usar o T-SQL. Por exemplo, `USE dbA` lista os esquemas `dbo` e `dev` assim como acontece no SQL Server. Os nomes dos esquemas mapeados, como `dba_dbo` e `dba_dev`, são visíveis a partir do PostgreSQL.

Cada banco de dados ainda contém seus esquemas. O nome de cada banco de dados é anexado ao nome do esquema do SQL Server utilizando um sublinhado como delimitador, por exemplo:

- `dba` contém `dba_dbo` e `dba_dev`.
- `dbb` contém `dbb_dbo` e `dbb_test`.
- `dbc` contém `dbc_dbo` e `dbc_prod`.

No interior do banco de dados `babelfish_db`, o usuário T-SQL ainda precisa executar `USE dbname` para alterar o contexto do banco de dados e, portanto, a aparência permanece semelhante à do SQL Server.

Escolher um modo de migração

Cada modo de migração tem vantagens e desvantagens. Escolha o modo de migração com base no número de bancos de dados de usuários que você possui e seus planos de migração. Depois

de criar um cluster para uso com o Babelfish, você não deve alterar o modo de migração, pois pode perder o acesso a todos os objetos SQL criados anteriormente. Ao escolher um modo de migração, considere os requisitos dos seus bancos de dados de usuários e clientes.

Quando você cria um cluster para uso com o Babelfish, o Aurora PostgreSQL cria os bancos de dados do sistema, `master` e `tempdb`. Se você tiver criado ou modificado objetos nos bancos de dados do sistema (`master` ou `tempdb`), certifique-se de recriar esses objetos no novo cluster. Ao contrário do SQL Server, o Babelfish não reinicializa `tempdb` após uma reinicialização do cluster.

Utilize o modo de migração de banco de dados único nos seguintes casos:

- Se estiver migrando um único banco de dados do SQL Server. No modo de banco de dados único, os nomes de esquemas migrados, quando acessados do PostgreSQL, são idênticos aos nomes de esquemas originais do SQL Server. Isso reduzirá as alterações de código nas consultas SQL existentes se você quiser otimizá-las para execução com uma conexão PostgreSQL.
- Se o seu objetivo final for uma migração completa para o Aurora PostgreSQL nativo. Antes de migrar, consolide seus esquemas em um único esquema (`dbo`) e depois migre para um único cluster para reduzir as alterações necessárias.

Utilize o modo de migração de vários bancos de dados nos seguintes casos:

- Se você quiser a experiência padrão do SQL Server com vários bancos de dados de usuários na mesma instância.
- Se vários bancos de dados de usuários precisarem ser migrados juntos.

Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish

Ao criar um cluster de banco de dados do Aurora PostgreSQL e escolher Turn on Babelfish (Ativar Babelfish), um grupo de parâmetros de cluster de banco de dados é criado para você automaticamente ao escolher Create new (Criar). Esse grupo de parâmetros de cluster de banco de dados é baseado no grupo de parâmetros de cluster de banco de dados do Aurora PostgreSQL para a versão do Aurora PostgreSQL escolhida para a instalação, por exemplo, o Aurora PostgreSQL versão 14. Ele é nomeado usando o seguinte padrão geral:

```
custom-aurora-postgresql14-babelfish-compat-3
```

Você pode alterar as configurações a seguir durante o processo de criação do cluster, mas algumas delas não podem ser alteradas depois de armazenadas no grupo de parâmetros personalizado, então escolha com cuidado:

- Banco de dados único ou vários bancos de dados
- Localidade de agrupamentos padrão
- Nome do agrupamento
- Grupo de parâmetros de banco de dados

Para usar um grupo de parâmetros de cluster de banco de dados do Aurora PostgreSQL versão 13 ou superior, edite o grupo e defina o parâmetro `babelfish_status` como on. Especifique todas as opções do Babelfish antes de criar o cluster do Aurora PostgreSQL. Para saber mais, consulte Grupos de parâmetros para Amazon Aurora.

Os parâmetros a seguir controlam as preferências do Babelfish. Salvo indicação em contrário na Descrição, os parâmetros podem ser modificados. O valor padrão está incluído na descrição. Para ver os valores permitidos para qualquer parâmetro, faça o seguinte:

> **Note**
>
> Ao associar um novo grupo de parâmetros de banco de dados a uma instância de banco de dados, os parâmetros estáticos e dinâmicos modificados serão aplicados somente após a reinicialização da instância de banco de dados. No entanto, se você modificar parâmetros dinâmicos no grupo de parâmetros de banco de dados depois de associá-lo à instância de banco de dados, essas alterações serão aplicadas imediatamente sem uma reinicialização.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Parameter groups (Grupos de parâmetros) no menu de navegação.
3. Escolha o grupo de parâmetros do cluster de banco de dados `default.aurora-postgresql14` na lista.
4. Insira o nome de um parâmetro no campo de pesquisa. Por exemplo, insira `babelfishpg_tsql.default_locale` no campo de pesquisa para exibir esse parâmetro e seu valor padrão e configurações permitidas.

> **Note**
>
> Os bancos de dados globais do Babelfish para Aurora PostgreSQL funcionarão em regiões secundárias somente se os parâmetros a seguir estiverem ativados nessas regiões.

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tsql.apg_enable_correlated_scalar_transform	Permite que o planejador transforme a subconsulta escalar correlacionada no Babelfish. (Padrão: ativado) (Permitido: ativado, desativado)	dinâmico	true
babelfishpg_tsql.apg_enable_subquery_cache	Permite o uso de cache para subconsultas escalares correlacionadas. (Padrão: ativado) (Permitido: ativado, desativado)	dinâmico	true
babelfishpg_tds.tds_default_numeric_scale	Define a escala padrão do tipo numérico a ser	dinâmico	true

Parameter	Descrição	Aplicar tipo	É modificável
	enviada nos metadados da coluna TDS se o mecanismo não especificar uma. (Padrão: 8) (Permitido: 0 a 38)		
babelfishpg_tds.tds_default_numeric_precision	Um valor inteiro que define a precisão padrão do tipo numérico a ser enviada nos metadados da coluna do TDS se o mecanismo não especificar uma. (Padrão: 38) (Permitido: 1 a 38)	dinâmico	true
babelfishpg_tds.tds_default_packet_size	Um valor inteiro que define o tamanho do pacote padrão para conectar clientes do SQL Server. (Padrão: 4096) (Permitido: 512 a 32767)	dinâmico	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tds.tds_default_protocol_version	Um valor inteiro que define uma versão padrão do protocolo TDS para conectar clientes. (Padrão: DEFAULT) (Permitido: TDSv7.0, TDSv7.1, TDSv7.1.1, TDSv7.2, TDSv7.3A, TDSv7.3B, TDSv7.4, DEFAULT)	dinâmico	true
babelfishpg_tds.default_server_name	Uma string que identifica o nome padrão do servidor do Babelfish. (Padrão: Microsoft SQL Server) (Permitido: null)	dinâmico	true
babelfishpg_tds.tds_debug_log_level	Um valor inteiro que define o nível de registro em log no TDS; 0 desativa o registro em log. (Padrão: 1) (Permitido: 0, 1, 2, 3)	dinâmico	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tds.listen_addresses	Uma string que define o nome do host ou um ou mais endereços IP nos quais ouvir o TDS. Esse parâmetro não pode ser modificado após a criação do cluster de banco de dados do Babelfish. (Padrão: *) (Permitido: null)	–	false
babelfishpg_tds.port	Um valor inteiro que especifica a porta TCP utilizada para solicitações na sintaxe do SQL Server. (Padrão: 1433) (Permitido: 1 a 65535)	estático	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tds.tds_ssl_encrypt	Um valor booliano que ativa (0) ou desativa (1) a criptografia de dados que atravessam a porta do ouvinte do TDS. Para obter informações detalhadas sobre como utilizar o SSL para conexões de cliente, consulte Configurações SSL e conexões do cliente do Babelfish. (Padrão: 0) (Permitido: 0, 1)	dinâmico	true
babelfishpg_tds.tds_ssl_max_protocol_version	Uma string que especifica a versão mais alta do protocolo SSL/TLS a ser utilizada na sessão do TDS. (Padrão: “TLSv1.2”) (Permitido: “TLSv1”, “TLSv1.1”, “TLSv1.2”)	dinâmico	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tds.tds_ssl_min_protocol_version	Uma string que especifica a versão mínima do protocolo SSL/TLS a ser utilizada na sessão do TDS. (Padrão: “TLSv1.2” do Aurora PostgreSQL versão 16, “TLSv1.1” para versões anteriores ao Aurora PostgreSQL versão 16) (Permitido: “TLSv1”, “TLSv1.1”, “TLSv1.2”).	dinâmico	true
babelfishpg_tds.unix_socket_directories	Uma string que identifica o diretório de soquetes Unix do servidor do TDS. Esse parâmetro não pode ser modificado após a criação do cluster de banco de dados do Babelfish. (Padrão: /tmp) (Permitido: null)	–	false

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tds.unix_socket_group	Uma string que identifica o grupo de soquetes Unix do servidor do TDS. Esse parâmetro não pode ser modificado após a criação do cluster de banco de dados do Babelfish. (Padrão: rdsdb) (Permitido: null)	–	false
babelfishpg_tsql.default_locale	Uma string que especifica a localidade padrão usada para agrupamentos do Babelfish. A localidade padrão é apenas a localidade e não inclui qualificadores. Defina esse parâmetro ao provisionar um cluster de banco de dados do Babelfish. Depois que o cluster de banco de dados for provisionado, as alterações nesse parâmetro serão ignoradas. (Padrão: en_US) (Permitido: consulte tabelas)	estático	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tsql.migration_mode	Uma lista não modificável que especifica a compatibilidade com bancos de dados de usuário único ou vários usuários. Defina esse parâmetro ao provisionar um cluster de banco de dados do Babelfish. Após o provisionamento do cluster de banco de dados, não é possível modificar o valor desse parâmetro. (Padrão: multi-db do Aurora PostgreSQL versão 16, single-db para versões anteriores à versão 16 do Aurora PostgreSQL) (Permitidos: single-db, multi-db, null)	estático	true

Parameter	Descrição	Aplicar tipo	É modificável
babelfishpg_tsql.server_collation_name	A string que especifica o nome do agrupamento utilizado para ações em nível de servidor. Defina esse parâmetro ao provisionar um cluster de banco de dados do Babelfish. Após o provisionamento do cluster de banco de dados, não modifique o valor desse parâmetro. (Padrão: bbf_unicode_general_ci_as) (Permitido: consulte tabelas)	estático	true
babelfishpg_tsql.version	Uma string que define a saída da variável @@VERSION. Não modifique esse valor para clusters de bancos de dados Aurora PostgreSQL. (Padrão: null) (Permitido: padrão)	dinâmico	true

Parameter	Descrição	Aplicar tipo	É modificável
rds.babelfish_status	Uma string que define o estado da funcionalidade do Babelfish. Quando esse parâmetro está definido como `datatypesonly` , o Babelfish está desativado, mas os tipos de dados do SQL Server ainda estão disponíveis. (Padrão: desativado) (Permitido: ativado, desativado, datatypes only)	estático	true
unix_socket_permissions	Um número inteiro que define as permissões de soquete Unix do servidor do TDS. Esse parâmetro não pode ser modificado após a criação do cluster de banco de dados do Babelfish . (Padrão: 0700) (Permitido: 0 a 511)	–	false

Configurações SSL e conexões do cliente do Babelfish

Para exigir conexões SSL/TLS com o cluster de bancos de dados do Babelfish para Aurora PostgreSQL, use o parâmetro `rds.force_ssl`.

- Para exigir conexões SSL/TLS, defina o valor do parâmetro `rds.force_ssl` como 1 (ativado).
- Para desativar as conexões SSL/TLS obrigatórias, defina o valor do parâmetro `rds.force_ssl` como 0 (desativado).

O valor padrão desse parâmetro depende da versão do Aurora PostgreSQL:

- Para o Aurora PostgreSQL versões 17 e posteriores: o valor padrão é 1 (ativado).
- Para o Aurora PostgreSQL versões 16 e posteriores: o valor padrão é 0 (desativado).

> **Note**
>
> Quando você realiza uma atualização de versão principal do Aurora PostgreSQL versão 16 ou anterior para a versão 17 ou posterior, o valor padrão do parâmetro muda de 0 (desativado) para 1 (ativado). Essa alteração pode causar falhas de conectividade em aplicações que não estão configuradas para SSL. Você pode reverter para o comportamento padrão anterior definindo esse parâmetro como 0 (desativado).

Para obter detalhes específicos do driver, consulte Conectar-se a um cluster de banco de dados do Babelfish.

Quando um cliente se conecta à porta do TDS (`1433` padrão), o Babelfish compara a configuração do Secure Sockets Layer (SSL) enviada durante o handshake do cliente com a configuração do parâmetro SSL do Babelfish (`tds_ssl_encrypt`). O Babelfish então determina se uma conexão é permitida. Em caso positivo, o comportamento de criptografia pode ser aplicado ou não, dependendo das configurações de parâmetros e do suporte à criptografia oferecido pelo cliente.

A seguinte tabela mostra como o Babelfish se comporta para cada combinação.

Configuração SSL do cliente	Configuração SSL do Babelfish	rds.force_ssl	Conexão permitida?	Valor retornado ao cliente
ENCRYPT_ON	Any	Any	Permitido, a conexão inteira é criptografada	ENCRYPT_ON
ENCRYPT_OFF	tds_ssl_encrypt=1	Any	Permitido, a conexão inteira é criptografada	ENCRYPT_REQ
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=0	Permitida, o pacote de login é criptografado	ENCRYPT_OFF
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=1	Não, conexão encerrada	ENCRYPT_OFF
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=0	Sim	ENCRYPT_NOT_SUP
ENCRYPT_NOT_SUP	tds_ssl_encrypt=1	Any	Não, conexão encerrada	ENCRYPT_REQ
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=1	Não, conexão encerrada	ENCRYPT_NOT_SUP
ENCRYPT_CLIENT_CERT	Any	Any	Não, conexão encerrada	Sem suporte

Noções básicas sobre agrupamentos no Babelfish para Aurora PostgreSQL

Ao criar um cluster de banco de dados do Aurora PostgreSQL com o Babelfish, você escolhe um agrupamento para seus dados. Um agrupamento especifica a ordem de classificação e os padrões de bits que produzem o texto ou os caracteres em determinada linguagem humana escrita. Um agrupamento inclui regras que comparam dados de determinado conjunto de padrões de bits. O agrupamento está relacionado à localização. Localidades diferentes afetam o mapeamento de caracteres, a ordem de classificação e similares. Os atributos de agrupamento são refletidos nos nomes de vários agrupamentos. Para obter informações sobre atributos, consulte Babelfish collation attributes table.

O Babelfish mapeia agrupamentos do SQL Server para agrupamentos comparáveis que são fornecidos pelo Babelfish. O Babelfish predefine agrupamentos Unicode com comparações de string culturalmente sensíveis e ordens de classificação. O Babelfish também fornece uma maneira de converter os agrupamentos no banco de dados SQL Server para o agrupamento Babelfish de correspondência mais próxima. Agrupamentos específicos de localidade são fornecidos para diferentes idiomas e regiões.

Alguns agrupamentos definem uma página de código que corresponde a uma codificação no lado do cliente. O Babelfish converte automaticamente da codificação do servidor na codificação do cliente, dependendo do agrupamento de cada coluna de saída.

O Babelfish é compatível com os agrupamentos listados na Babelfish supported collations table. O Babelfish mapeia agrupamentos do SQL Server para agrupamentos comparáveis que são fornecidos pelo Babelfish.

O Babelfish usa a versão 153.80 da biblioteca de agrupamentos International Components for Unicode (ICU). Para obter mais informações sobre agrupamentos ICU, consulte Collation (Agrupamento)) na documentação da ICU. Para saber mais sobre o PostgreSQL e agrupamentos, consulte Collation Support (Compatibilidade com agrupamentos) na documentação do PostgreSQL.

Tópicos

- Parâmetros de cluster de banco de dados que controlam o agrupamento e a localidade
- Agrupamentos determinísticos e não determinísticos no Babelfish
- Agrupamentos permitidos no nível do banco de dados no Babelfish
- Agrupamentos de servidor e objetos no Babelfish
- Comportamento padrão de agrupamentos no Babelfish

- Gerenciar agrupamentos
- Limitações de agrupamentos e diferenças de comportamento

Parâmetros de cluster de banco de dados que controlam o agrupamento e a localidade

Os parâmetros a seguir afetam o comportamento do agrupamento.

babelfishpg_tsql.default_locale

Esse parâmetro especifica a localidade padrão usada pelo agrupamento. Esse parâmetro é utilizado em combinação com os atributos listados na Babelfish collation attributes table para personalizar agrupamentos para um idioma e uma região específicos. O valor padrão desse parâmetro é `en-US`.

A localidade padrão aplica-se a todos os nomes de agrupamentos do Babelfish que começam com “BBF” e a todos os agrupamentos do SQL Server mapeados para agrupamentos do Babelfish. Alterar a configuração para esse parâmetro em um cluster de banco de dados do Babelfish existente não afeta a localidade dos agrupamentos existentes. Para ver a lista de agrupamentos, consulte a Babelfish supported collations table.

babelfishpg_tsql.server_collation_name

Esse parâmetro especifica o agrupamento padrão para o servidor (instância de cluster de banco de dados do Aurora PostgreSQL) e do banco de dados. O valor padrão é `sql_latin1_general_cp1_ci_as`. `server_collation_name` deve ser um agrupamento `CI_AS`, pois no T-SQL o agrupamento de servidor determina como os identificadores são comparados.

Ao criar seu cluster de banco de dados do Babelfish, escolha o Collation name (Nome do agrupamento) na lista selecionável. Estes incluem os agrupamentos listados na Babelfish supported collations table. Não modifique `server_collation_name` depois que o banco de dados Babelfish for criado.

As configurações escolhidas ao criar seu cluster de banco de dados do Babelfish para Aurora PostgreSQL são armazenadas no grupo de parâmetros do cluster de banco de dados associado ao cluster para esses parâmetros e definem seu comportamento de agrupamento.

Agrupamentos determinísticos e não determinísticos no Babelfish

O Babelfish oferece suporte para agrupamentos determinísticos e não determinísticos:

- Um agrupamento determinístico avalia caracteres que têm sequências de bytes idênticas como iguais. Isso significa que x e X não são iguais em um agrupamento determinístico. Agrupamentos determinísticos fazem distinção de maiúsculas e minúsculas (CS) e dialetos (AS).
- Um agrupamento não determinístico não precisa de correspondência idêntica. Um agrupamento não determinístico avalia x e X como iguais. Agrupamentos não determinísticos não fazem distinção de maiúsculas e minúsculas (CI) e de acentos (IA).

Na tabela a seguir, você pode encontrar algumas diferenças de comportamento entre o Babelfish e o PostgreSQL ao usar agrupamentos não determinísticos.

Babelfish	PostgreSQL
É compatível com a cláusula LIKE para agrupamentos CI_AS.	Não é compatível com a cláusula LIKE em agrupamentos não determinísticos.
É compatível com a cláusula LIKE somente nos seguintes agrupamentos de IA do Babelfish versão 4.2.0: • bbf_unicode_cp1250_ci_ai • bbf_unicode_cp1250_cs_ai • bbf_unicode_cp1257_ci_ai • bbf_unicode_cp1257_cs_ai • bbf_unicode_cp1_ci_ai • bbf_unicode_cp1_cs_ai • estonian_ci_ai • finnish_swedish_ci_ai • french_ci_ai • latin1_general_ci_ai • latin1_general_cs_ai • modern_spanish_ci_ai • polish_ci_ai • sql_latin1_general_cp1_ci_ai • sql_latin1_general_cp1_cs_ai	Não é compatível com a cláusula LIKE em agrupamentos não determinísticos.

Babelfish	PostgreSQL
• traditional_spanish_ci_ai	

Para obter uma lista de outras limitações e diferenças de comportamento do Babelfish em comparação com o SQL Server e o PostgreSQL, consulte Limitações de agrupamentos e diferenças de comportamento.

O Babelfish e o SQL Server seguem uma convenção de nomenclatura para agrupamentos que descrevem os atributos de agrupamento, conforme mostrado na tabela a seguir.

Atributo	Descrição
AI	Não tem distinção entre dialetos.
AS	Tem distinção entre dialetos.
BIN2	O BIN2 solicita que os dados sejam classificados em ordem de pontos de código. A ordem dos pontos de código Unicode é a mesma ordem de caracteres de codificações UTF-8, UTF-16 e UCS-2. A ordem de pontos de código é um agrupamento determinístico rápido.
CI	Não tem distinção entre maiúsculas e minúsculas.
CS	Tem distinção entre maiúsculas e minúsculas.
PREF	Para classificar letras maiúsculas antes de letras minúsculas, utilize um agrupamento PREF. Se a comparação não diferenciar maiúsculas de minúsculas, a versão em maiúsculas de uma letra será classificada antes da versão em minúscula, caso não haja outra distinção. A biblioteca ICU oferece suporte à preferência de maiúsculas com `colCaseFirst=upper` , mas não para agrupamentos CI_AS. PREF só pode ser aplicado a agrupamentos determinísticos `CS_AS`.

Agrupamentos permitidos no nível do banco de dados no Babelfish

Os seguintes agrupamentos são permitidos no nível do banco de dados no Babelfish:

- bbf_unicode_bin2
- bbf_unicode_cp1_ci_ai
- bbf_unicode_cp1_ci_as
- bbf_unicode_cp1250_ci_ai
- bbf_unicode_cp1250_ci_as
- bbf_unicode_cp1257_ci_ai
- bbf_unicode_cp1257_ci_as
- estonian_ci_ai
- estonian_ci_as
- finnish_swedish_ci_ai
- finnish_swedish_ci_as
- french_ci_ai
- french_ci_as
- latin1_general_bin2
- latin1_general_ci_ai
- latin1_general_ci_as
- latin1_general_90_bin2
- latin1_general_100_bin2
- latin1_general_140_bin2
- modern_spanish_ci_ai
- modern_spanish_ci_as
- polish_ci_ai
- polish_ci_as
- sql_latin1_general_cp1_ci_ai
- sql_latin1_general_cp1_ci_as
- sql_latin1_general_cp1250_ci_as
- sql_latin1_general_cp1251_ci_as
- sql_latin1_general_cp1257_ci_as
- traditional_spanish_ci_ai
- traditional_spanish_ci_as

> **Note**
>
> Para usar um agrupamento diferente no nível do banco de dados, certifique-se de que ele corresponda ao agrupamento no nível do servidor. Para ter mais informações, consulte Agrupamentos de servidor e objetos no Babelfish.

Agrupamentos de servidor e objetos no Babelfish

Use os seguintes agrupamentos como um agrupamento de servidor ou um agrupamento de objetos.

ID de agrupamento	Observações
bbf_unicode_general_ci_as	Oferece suporte à comparação sem distinção entre maiúsculas e minúsculas e operador LIKE.
bbf_unicode_cp1_ci_as	Agrupamento não determinístico, também conhecido como CP1252.
bbf_unicode_CP1250_ci_as	Agrupamento não determinístico utilizado para representar textos em idiomas da Europa Central e do Leste Europeu que utilizam o script latino.
bbf_unicode_CP1251_ci_as	Agrupamento não determinístico para linguagens que utilizam o script cirílico.
bbf_unicode_cp1253_ci_as	Agrupamento não determinístico utilizado para representar o grego moderno.
bbf_unicode_cp1254_ci_as	Agrupamento não determinístico que é compatível com o idioma turco.
bbf_unicode_cp1255_ci_as	Agrupamento não determinístico que é compatível com o idioma hebraico.

ID de agrupamento	Observações
bbf_unicode_cp1256_ci_as	Agrupamento não determinístico utilizado para escrever idiomas que utilizam o script árabe.
bbf_unicode_cp1257_ci_as	Agrupamento não determinístico utilizado para suporte aos idiomas estoniano, letão e lituano.
bbf_unicode_cp1258_ci_as	Agrupamento não determinístico utilizado para escrever caracteres em vietnamita.
bbf_unicode_cp874_ci_as	Agrupamento não determinístico utilizado para escrever caracteres em tailandês.
sql_latin1_general_cp1250_ci_as	Codificação de caracteres de byte único não determinística utilizado para representar caracteres latinos.
sql_latin1_general_cp1251_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1253_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1254_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1255_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.

ID de agrupamento	Observações
sql_latin1_general_cp1256_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1257_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
sql_latin1_general_cp1258_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.
chinese_prc_ci_as	Agrupamento não determinístico que é compatível com o idioma chinês (RPC).
cyrillic_general_ci_as	Agrupamento não determinístico que é compatível com o idioma cirílico.
finnish_swedish_ci_as	Agrupamento não determinístico que é compatível com o idioma finlandês.
french_ci_as	Agrupamento não determinístico que é compatível com o idioma francês.
japanese_ci_as	Agrupamento não determinístico que é compatível com o idioma japonês. Compatível com o Babelfish 2.1.0 e versões posteriores.
korean_wansung_ci_as	Agrupamento não determinístico que é compatível com o idioma coreano (com classificação de dicionário).
latin1_general_ci_as	Agrupamento não determinístico que oferece suporte a caracteres latinos.

ID de agrupamento	Observações
modern_spanish_ci_as	Agrupamento não determinístico que é compatível com o idioma espanhol moderno.
polish_ci_as	Agrupamento não determinístico que é compatível com o idioma polonês.
thai_ci_as	Agrupamento não determinístico que é compatível com o idioma tailandês.
traditional_spanish_ci_as	Agrupamento não determinístico que é compatível com o idioma espanhol (tipo tradicional).
turkish_ci_as	Agrupamento não determinístico que é compatível com o idioma turco.
ukrainian_ci_as	Agrupamento não determinístico que é compatível com o idioma ucraniano.
vietnamese_ci_as	Agrupamento não determinístico que é compatível com o idioma vietnamita.

Os seguintes agrupamentos podem ser utilizados como agrupamentos de objetos.

Dialeto	Opções determinísticas	Opções não determinísticas
Árabe	Arabic_CS_AS	Arabic_CI_AS Arabic_CI_AI
Alfabeto árabe	BBF_Unicode_CP1256_CS_AS	BBF_Unicode_CP1256_CI_AI

Dialeto	Opções determinísticas	Opções não determinísticas
	BBF_Unicode_Pref_CP1256_CS_AS	BBF_Unicode_CP1256_CS_AI
Binário	latin1_general_bin2 BBF_Unicode_BIN2	–
Idiomas da Europa Central e do Leste Europeu que utilizam o alfabeto latino	BBF_Unicode_CP1250_CS_AS BBF_Unicode_Pref_CP1250_CS_AS	BBF_Unicode_CP1250_CI_AI BBF_Unicode_CP1250_CS_AI
Chinês	Chinese_PRC_CS_AS	Chinese_PRC_CI_AS Chinese_PRC_CI_AI
Cyrillic_General	Cyrillic_General_CS_AS	Cyrillic_General_CI_AS Cyrillic_General_CI_AI
Alfabeto cirílico	BBF_Unicode_CP1251_CS_AS BBF_Unicode_Pref_CP1251_CS_AS	BBF_Unicode_CP1251_CI_AI BBF_Unicode_CP1251_CS_AI
Estoniano	Estonian_CS_AS	Estonian_CI_AS Estonian_CI_AI
Estoniano, letão e lituano	BBF_Unicode_CP1257_CS_AS BBF_Unicode_Pref_CP1257_CS_AS	BBF_Unicode_CP1257_CI_AI BBF_Unicode_CP1257_CS_AI

Dialeto	Opções determinísticas	Opções não determinísticas
Finnish_Swedish	Finnish_Swedish_CS_AS	Finnish_Swedish_CI_AS Finnish_Swedish_CI_AI
Francês	French_CS_AS	French_CI_AS French_CI_AI
Grego	Greek_CS_AS	Greek_CI_AS Greek_CI_AI
Hebraico	BBF_Unicode_CP1255_CS_AS BBF_Unicode_Pref_CP1255_CS_AS Hebrew_CS_AS	BBF_Unicode_CP1255_CI_AI BBF_Unicode_CP1255_CS_AI Hebrew_CI_AS Hebrew_CI_AI
Japonês (Babelfish 2.1.0 e posteriores)	Japanese_CS_AS	Japanese_CI_AI Japanese_CI_AS
Korean_Wamsung	Korean_Wamsung_CS_AS	Korean_Wamsung_CI_AS Korean_Wamsung_CI_AI

Dialeto	Opções determinísticas	Opções não determinísticas
Caracteres latinos para a página de código CP1252	latin1_general_cs_as BBF_Unicode_General_CS_AS BBF_Unicode_General_Pref_CS_AS BBF_Unicode_Pref_CP1_CS_AS BBF_Unicode_CP1_CS_AS	latin1_general_ci_as latin1_general_ci_ai latin1_general_cs_ai BBF_Unicode_General_CI_AI BBF_Unicode_General_CS_AI BBF_Unicode_CP1_CI_AI BBF_Unicode_CP1_CS_AI
Grego moderno	BBF_Unicode_CP1253_CS_AS BBF_Unicode_Pref_CP1253_CS_AS	BBF_Unicode_CP1253_CI_AI BBF_Unicode_CP1253_CS_AI
Modern_Spanish	Modern_Spanish_CS_AS	Modern_Spanish_CI_AS Modern_Spanish_CI_AI
Mongol	Mongolian_CS_AS	Mongolian_CI_AS Mongolian_CI_AI
Polonês	Polish_CS_AS	polish_ci_as Polish_CI_AI
Tailandês	BBF_Unicode_CP874_CS_AS BBF_Unicode_Pref_CP874_CS_AS Thai_CS_AS	BBF_Unicode_CP874_CI_AI BBF_Unicode_CP874_CS_AI Thai_CI_AS, Thai_CI_AI

Dialeto	Opções determinísticas	Opções não determinísticas
Tradition al_Spanish	Traditional_Spanish_CS_AS	Traditional_Spanish_CI_AS Traditional_Spanish_CI_AI
Turco	BBF_Unicode_CP1254_CS_AS BBF_Unicode_Pref_CP1254_CS_AS Turkish_CS_AS	BBF_Unicode_CP1254_CI_AI BBF_Unicode_CP1254_CS_AI Turkish_CI_AS, Turkish_CI_AI
Ucraniano	Ukranian_CS_AS	Ukranian_CI_AS Ukranian_CI_AI
Vietnamita	BBF_Unicode_CP1258_CS_AS BBF_Unicode_Pref_CP1258_CS_AS Vietnamese_CS_AS	BBF_Unicode_CP1258_CI_AI BBF_Unicode_CP1258_CS_AI Vietnamese_CI_AS Vietnamese_CI_AI

Comportamento padrão de agrupamentos no Babelfish

Antes, o agrupamento padrão dos tipos de dados agrupáveis era `pg_catalog.default`. Os tipos de dados e os objetos que dependem desses tipos de dados seguem um agrupamento que diferencia maiúsculas de minúsculas. Essa condição pode afetar os objetos T-SQL do conjunto de dados com agrupamento que não diferencia maiúsculas de minúsculas. A partir do Babelfish 2.3.0, o agrupamento padrão para os tipos de dados agrupáveis (exceto TEXT e NTEXT) é o mesmo que o agrupamento no parâmetro `babelfishpg_tsql.server_collation_name`. Quando você atualiza para o Babelfish 2.3.0, o agrupamento padrão é escolhido automaticamente no momento da criação do cluster de banco de dados, o que não cria nenhum impacto visível.

Gerenciar agrupamentos

A biblioteca ICU fornece rastreamento de versão de agrupamento para garantir que os índices que dependem de agrupamentos possam ser reindexados quando uma nova versão do ICU estiver disponível. Para ver se o banco de dados atual tem agrupamentos que precisam ser atualizados, você pode usar a seguinte consulta depois de se conectar usando `psql` ou `pgAdmin`:

```
SELECT pg_describe_object(refclassid, refobjid,
    refobjsubid) AS "Collation",
    pg_describe_object(classid, objid, objsubid) AS "Object"
    FROM pg_depend d JOIN pg_collation c ON refclassid = 'pg_collation'::regclass
    AND refobjid = c.oid WHERE c.collversion <> pg_collation_actual_version(c.oid)
    ORDER BY 1, 2;
```

Essa consulta retorna o seguinte resultado:

```
 Collation | Object
-----------+--------
(0 rows)
```

Neste exemplo, nenhum agrupamento precisa ser atualizado.

Para obter uma lista dos agrupamentos predefinidos em seu banco de dados do Babelfish, você pode usar `psql` ou `pgAdmin` com a seguinte consulta:

```
SELECT * FROM pg_collation;
```

Agrupamentos predefinidos são armazenados na tabela `sys.fn_helpcollations`. É possível utilizar o seguinte comando para exibir informações sobre um agrupamento (como os sinalizadores lcid, style e collate). Para obter uma lista de todos os agrupamentos usando `sqlcmd`, conecte-se à porta T-SQL (1433, por padrão) e execute a seguinte consulta:

```
1> :setvar SQLCMDMAXVARTYPEWIDTH 40
2> :setvar SQLCMDMAXFIXEDTYPEWIDTH 40
3> SELECT * FROM fn_helpcollations()
4> GO
name                                 description
------------------------------------ ----------------------------------------
arabic_cs_as                         Arabic, case-sensitive, accent-sensitive
arabic_ci_ai                         Arabic, case-insensitive, accent-insensi
arabic_ci_as                         Arabic, case-insensitive, accent-sensiti
```

```
bbf_unicode_bin2                          Unicode-General, case-sensitive, accent-
bbf_unicode_cp1250_ci_ai                  Default locale, code page 1250, case-ins
bbf_unicode_cp1250_ci_as                  Default locale, code page 1250, case-ins
bbf_unicode_cp1250_cs_ai                  Default locale, code page 1250, case-sen
bbf_unicode_cp1250_cs_as                  Default locale, code page 1250, case-sen
bbf_unicode_pref_cp1250_cs_as             Default locale, code page 1250, case-sen
bbf_unicode_cp1251_ci_ai                  Default locale, code page 1251, case-ins
bbf_unicode_cp1251_ci_as                  Default locale, code page 1251, case-ins
bbf_unicode_cp1254_ci_ai                  Default locale, code page 1254, case-ins
...
(124 rows affected)
```

As linhas 1 e 2 mostradas no exemplo restringem a saída apenas para fins de legibilidade da documentação.

```
1> SELECT SERVERPROPERTY('COLLATION')
2> GO
serverproperty
------------------------------------------------------------------------
sql_latin1_general_cp1_ci_as

(1 rows affected)
1>
```

Limitações de agrupamentos e diferenças de comportamento

O Babelfish usa a biblioteca ICU para suporte a agrupamentos. O PostgreSQL é construído com uma versão específica do ICU e pode corresponder no máximo uma versão de um agrupamento. Variações entre versões são inevitáveis, assim como pequenas variações ao longo do tempo à medida que os idiomas evoluem. Na lista a seguir, você pode encontrar algumas limitações conhecidas e variações de comportamento dos agrupamentos do Babelfish:

- Índices e dependência de tipo de agrupamento: um índice em um tipo definido pelo usuário que depende da biblioteca de agrupamentos International Components for Unicode (ICU) (biblioteca utilizada pelo Babelfish) não é invalidado quando a versão da biblioteca é alterada.
- Função COLLATIONPROPERTY: propriedades de agrupamento são implementadas apenas para os agrupamentos BBF do Babelfish compatíveis. Para obter mais informações, consulte Babelfish supported collations table.
- Diferenças de regras de classificação Unicode: agrupamentos SQL para o SQL Server classificam dados codificados em Unicode (`nchar` e `nvarchar`) de forma diferente dos dados que não

são codificados em Unicode (`char` e `varchar`). Os bancos de dados do Babelfish são sempre codificados em UTF-8 e sempre aplicam regras de classificação Unicode de maneira consistente, independentemente do tipo de dados, portanto, a ordem de classificação para `char` ou `varchar` é a mesmo que para `nchar` ou `nvarchar`.

- Agrupamentos de igualdade secundária e comportamento da classificação: o agrupamento padrão ICU de igualdade secundária Unicode (`CI_AS`) classifica sinais de pontuação e outros caracteres não alfanuméricos antes dos caracteres numéricos e os caracteres numéricos antes dos caracteres alfabéticos. No entanto, a ordem de pontuação e outros caracteres especiais é diferente.
- Agrupamentos terciários, solução alternativa para ORDER BY: agrupamentos SQL, como `SQL_Latin1_General_Pref_CP1_CI_AS`, são compatíveis com a função `TERTIARY_WEIGHTS` e a capacidade de classificar strings que se comparam igualmente em um agrupamento `CI_AS` para classificação inicialmente em maiúsculas: `ABC`, `ABc`, `AbC`, `Abc`, `aBC`, `aBc`, `abC` e finalmente `abc`. Assim, a função `DENSE_RANK OVER (ORDER BY column)` analítica avalia essas strings como tendo a mesma classificação, mas as ordena em maiúsculas primeiro dentro de uma partição.

  Você pode obter um resultado semelhante com o Babelfish adicionando uma cláusula `COLLATE` à cláusula `ORDER BY` que especifica um agrupamento `CS_AS` terciário que especifica `@colCaseFirst=upper`. No entanto, o modificador `colCaseFirst` aplica-se somente a strings com igualdade terciária (em vez de igualdade secundária, como um agrupamento `CI_AS`). Por isso, você não pode emular agrupamentos SQL terciários utilizando um único agrupamento ICU.

  Como solução alternativa, convém aplicações que utilizam o agrupamento `SQL_Latin1_General_Pref_CP1_CI_AS` de forma a utilizar o agrupamento `BBF_SQL_Latin1_General_CP1_CI_AS` primeiro. Em seguida, adicione `COLLATE BBF_SQL_Latin1_General_Pref_CP1_CS_AS` a qualquer cláusula `ORDER BY` para essa coluna.
- Expansão de caracteres: uma expansão de caracteres trata um único caractere como igual a uma sequência de caracteres em nível primário. O agrupamento padrão do SQL Server `CI_AS` é compatível com a expansão de caracteres. Os agrupamentos de ICU são compatíveis com a expansão de caracteres somente para agrupamentos que não distinguem dialetos.

  Quando a expansão de caracteres for necessária, utilize um agrupamento `AI` para comparações. No entanto, esses agrupamentos atualmente não têm suporte pelo operador LIKE.
- Codificação char e varchar: quando agrupamentos SQL são utilizados para tipos de dados `char` ou `varchar`, a ordem de classificação dos caracteres antes de ASCII 127 é determinada pela

página de código específica desse agrupamento SQL. Para agrupamentos SQL, as strings declaradas como `char` ou `varchar` podem ser classificadas de maneira diferente de strings declaradas como `nchar` ou `nvarchar`.

O PostgreSQL codifica todas as strings com a codificação do banco de dados, para que todos os caracteres sejam convertidos em UTF-8 e classificados utilizando regras Unicode.

Como os agrupamentos SQL classificam tipos de dados nchar e nvarchar utilizando regras Unicode, o Babelfish codifica todas as strings no servidor utilizando UTF-8. O Babelfish classifica strings nchar e nvarchar da mesma maneira que classifica strings char e varchar, utilizando regras Unicode.

- Caractere suplementar: as funções do SQL Server `NCHAR`, `UNICODE` e `LEN` são compatíveis com caracteres para pontos de código fora do Unicode Basic Multilingual Plane (BMP). Em contraste, agrupamentos não SC utilizam caracteres de pares substitutos para lidar com caracteres suplementares. Para tipos de dados Unicode, o SQL Server pode representar até 65.535 caracteres utilizando UCS-2 ou o intervalo Unicode completo (1.114.111 caracteres) se forem utilizados caracteres suplementares.
- Agrupamentos que fazem distinção do Kana (KS): um agrupamento que faz distinção do Kana (KS) é aquele que trata caracteres Kana japoneses `Hiragana` e `Katakana` de forma diferente. O ICU é compatível com o padrão de agrupamento japonês `JIS X 4061`. O agora obsoleto modificador de localidade `colhiraganaQ [on | off]` pode fornecer a mesma funcionalidade que agrupamentos KS. No entanto, agrupamentos KS com o mesmo nome que os do SQL Server atualmente não têm suporte pelo Babelfish.
- Agrupamentos sensíveis à largura (WS): quando um caractere de byte único (meia largura) e o mesmo caractere representado como um caractere de byte duplo (largura total) são tratados de maneiras diferentes, o agrupamento é chamado de sensível à largura (WS). Agrupamentos WS com o mesmo nome que os do SQL Server atualmente não têm suporte pelo Babelfish.
- Agrupamentos que fazem distinção do seletor de variação (VSS): agrupamentos que fazem distinção do seletor de variação (VSS) distinguem entre seletores de variação ideográfica em agrupamentos `Japanese_Bushu_Kakusu_140` e `Japanese_XJIS_140` do japonês. Uma sequência de variação é composta por um caractere base mais um seletor de variação adicional. Se você não selecionar a opção `_VSS`, o seletor de variação não será considerado na comparação.

  Atualmente, agrupamentos VSS não têm suporte pelo Babelfish.

- Agrupamentos BIN e BIN2: um agrupamento BIN2 classifica os caracteres de acordo com a ordem de pontos de código. A ordem binária byte a byte do UTF-8 preserva a ordem dos pontos de código Unicode e, portanto, é provável que esse também seja o agrupamento com a melhor performance. Se a ordem de pontos de código Unicode funcionar para uma aplicação, considere utilizar um agrupamento BIN2. No entanto, o uso de um agrupamento BIN2 pode resultar na exibição de dados no cliente em uma ordem culturalmente inesperada. Como novos mapeamentos para caracteres em minúsculas são adicionados ao Unicode com o passar do tempo, a função `LOWER` pode operar de maneira diferente em diferentes versões do ICU. Esse é um caso especial do problema mais geral de versionamento de agrupamentos, e não algo específico do agrupamento BIN2.

  O Babelfish fornece o agrupamento `BBF_Latin1_General_BIN2` com a distribuição do Babelfish para agrupar em ordem de pontos de código Unicode. Em um agrupamento BIN, somente o primeiro caractere é classificado como um wchar. Os caracteres restantes são classificados byte a byte, efetivamente na ordem de pontos de código, de acordo com a codificação. Essa abordagem não segue regras de agrupamento Unicode e não tem suporte pelo Babelfish.

- Agrupamentos não determinísticos e limitação CHARINDEX: para versões do Babelfish anteriores à versão 2.1.0, você não pode usar CHARINDEX com agrupamentos não determinísticos. Por padrão, o Babelfish usa um agrupamento que não faz distinção de maiúsculas e minúsculas (não determinístico). Usar CHARINDEX para versões anteriores do Babelfish gera o seguinte erro de tempo de execução:

  ```
  nondeterministic collations are not supported for substring searches
  ```

  > **Note**
  >
  > Essa limitação e solução alternativa se aplicam somente ao Babelfish versão 1.x (versões 13.x do Aurora PostgreSQL). O Babelfish 2.1.0 e versões posteriores não têm esse problema.

  Para contornar este problema, execute um dos seguintes procedimentos:

  - Converta explicitamente a expressão em um agrupamento que diferencie letras maiúsculas de minúsculas e deixe em minúsculas ambos os argumentos aplicando LOWER ou UPPER. Por exemplo, `SELECT charindex('x', a) FROM t1` transforma-se no seguinte:

```
SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
```

- Crie a função f_charindex do SQL e substitua chamadas CHARINDEX por chamadas para a seguinte função:

```
CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) RETURNS int
AS
BEGIN
declare @i int = 1
WHILE len(@s2) >= len(@s1)
BEGIN
  if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
  set @i += 1
  set @s2 = substring(@s2,2,999999999)
END
return 0
END
go
```

Gerenciar o tratamento de erros do Babelfish com hatches de escape

O Babelfish imita o comportamento do SQL para fluxo de controle e estado de transações sempre que possível. Quando o Babelfish encontra um erro, ele retorna um código de erro semelhante ao código de erro do SQL Server. Se o Babelfish não conseguir mapear o erro para um código do SQL Server, ele retornará um código de erro fixo (`33557097`) e realizará ações específicas com base no tipo de erro, da seguinte forma:

- Se for um erro de tempo de compilação, o Babelfish reverterá a transação.
- No caso de erros de tempo de execução, o Babelfish encerrará o lote e reverterá a transação.
- Para um erro de protocolo entre o cliente e o servidor, a transação não será revertida.

Se um código de erro não puder ser mapeado para um código equivalente, e o código de um erro semelhante estiver disponível, o código do erro será mapeado para esse código alternativo. Por exemplo, os comportamentos que causam os códigos `8143` e `8144` do SQL Server são ambos mapeados para `8143`.

Erros que não podem ser mapeados não respeitam uma construção `TRY... CATCH`.

Você pode usar `@@ERROR` para retornar um código de erro do SQL Server ou a função `@@PGERROR` para retornar um código de erro do PostgreSQL. Também pode utilizar a função `fn_mapped_system_error_list` para retornar uma lista de códigos de erro mapeados. Para obter informações sobre códigos de erro do PostgreSQL, consulte o site do PostgreSQL.

Modificar as configurações do hatch de escape do Babelfish

Para lidar com instruções que podem falhar, o Babelfish define certas opções denominadas hatches de escape. Um hatch de escape é uma opção que especifica o comportamento do Babelfish quando ele se depara com um recurso ou uma sintaxe sem suporte.

Você pode utilizar o procedimento armazenado `sp_babelfish_configure` para controlar as configurações de um hatch de escape. Use o script para definir o hatch de escape como `ignore` ou `strict`. Se estiver definido como `strict`, o Babelfish retornará um erro que você precisará corrigir antes de continuar.

Para aplicar as alterações à sessão atual e no nível do cluster, inclua a palavra-chave `server`.

O uso é o seguinte:

- Para listar todos os hatches de escape e seu status, além de informações de uso, execute `sp_babelfish_configure`.
- Para listar os hatches de escape nomeados e seus valores, seja para a sessão atual ou em todo o cluster, execute o comando `sp_babelfish_configure '`*`hatch_name`*`'`, em que *`hatch_name`* é o identificador de um ou mais hatches de escape. *`hatch_name`* pode utilizar curingas SQL, como "%".
- Para definir um ou mais hatches de escape para o valor especificado, execute `sp_babelfish_configure ['`*`hatch_name`*`' [, 'strict'|'ignore' [, 'server']]]`. Para tornar as configurações permanentes em todo o cluster, inclua a palavra-chave `server`, conforme mostrado a seguir:

  ```
  EXECUTE sp_babelfish_configure 'escape_hatch_unique_constraint', 'ignore', 'server'
  ```

  Para configurá-las apenas para a sessão atual, não use `server`.
- Para redefinir todos os hatches de escape para seus valores padrão, execute `sp_babelfish_configure 'default'` (Babelfish 1.2.0 e posterior).

A string que identifica o hatch (ou os hatches) pode conter curingas SQL. Por exemplo, o seguinte define todos os hatches de escape de sintaxe como `ignore` para o cluster do Aurora PostgreSQL.

```
EXECUTE sp_babelfish_configure '%', 'ignore', 'server'
```

Na tabela a seguir, você pode encontrar descrições e valores padrão para os hatches de escape predefinidos do Babelfish.

Hatch de escape	Descrição	Padrão
`escape_hatch_checkpoint`	Permite o uso da instrução CHECKPOINT no código processual, mas a instrução CHECKPOINT não está implement ada no momento.	ignorar
`escape_hatch_constraint_nam e_for_default`		ignorar

Hatch de escape	Descrição	Padrão
	Controla o comportamento do Babelfish relacionado a nomes de restrições padrão.	
`escape_hatch_database_misc_options`	Controla o comportamento do Babelfish relacionado às seguintes opções em CREATE DATABASE: CONTAINMENT, DB_CHAINING, TRUSTWORTHY, PERSISTENT_LOG_BUFFER.	ignorar
`escape_hatch_for_replication`	Controla o comportamento do Babelfish relacionado à cláusula [NOT] FOR REPLICATION ao criar ou modificar uma tabela.	strict
`escape_hatch_fulltext`	Controla o comportamento do Babelfish relacionado a recursos FULLTEXT, como DEFAULT_FULLTEXT_LANGUAGE in CREATE/ALTER DATABASE, CREATE FULLTEXT INDEX ou sp_fulltext_database.	ignorar
`escape_hatch_ignore_dup_key`	Controla o comportamento do Babelfish relacionado a CREATE/ALTER TABLE e CREATE INDEX. Quando IGNORE_DUP_KEY=ON, é gerado um erro quando definido como `strict` (o padrão) ou o erro é ignorado quando definido como `ignore` (Babelfish versão 1.2.0 e posterior).	strict

Hatch de escape	Descrição	Padrão
`escape_hatch_index_clustering`	Controla o comportamento do Babelfish relacionado às palavras-chave CLUSTERED ou NONCLUSTERED para índices e restrições PRIMARY KEY ou UNIQUE. Quando CLUSTERED é ignorado, o índice ou a restrição ainda é criado como se NONCLUSTERED tivesse sido especificada.	ignorar
`escape_hatch_index_columnstore`	Controla o comportamento do Babelfish relacionado à cláusula COLUMNSTORE. Se você especificar `ignore`, o Babelfish criará um índice normal de árvore B.	strict
`escape_hatch_join_hints`	Controla o comportamento de palavras-chave em um operador JOIN: LOOP, HASH, MERGE, REMOTE, REDUCE, REDISTRIB UTE, REPLICATE.	ignorar

Hatch de escape	Descrição	Padrão
`escape_hatch_language_non_english`	Controla o comportamento do Babelfish relacionado a idiomas diferentes do inglês para mensagens na tela. No momento, o Babelfish oferece suporte apenas a `us_english` para mensagens na tela. SET LANGUAGE pode utilizar uma variável contendo o nome do idioma e, portanto, o idioma real que está sendo definido apenas pode ser detectado em tempo de execução.	strict
`escape_hatch_login_hashed_password`	Quando ignorada, suprime o erro para a palavra-chave `HASHED` para `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_misc_options`	Quando ignorada, suprime o erro para outras palavras-chave além de `HASHED`, `MUST_CHANGE` , `OLD_PASSWORD` e `UNLOCK`para `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_old_password`	Quando ignorada, suprime o erro para a palavra-chave `OLD_PASSWORD` para `CREATE LOGIN` e `ALTER LOGIN`.	strict

Hatch de escape	Descrição	Padrão
`escape_hatch_login_password_must_change`	Quando ignorada, suprime o erro para a palavra-chave `MUST_CHANGE` para `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_password_unlock`	Quando ignorada, suprime o erro para a palavra-chave `UNLOCK` para `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_nocheck_add_constraint`	Controla o comportamento do Babelfish relacionado à cláusula WITH CHECK ou NOCHECK para restrições.	strict
`escape_hatch_nocheck_existing_constraint`	Controla o comportamento do Babelfish relacionado a restrições FOREIGN KEY ou CHECK.	strict
`escape_hatch_query_hints`	Controla o comportamento do Babelfish relacionado a dicas de consultas. Quando essa opção é definida como ignore, o servidor ignora as dicas que usam a cláusula OPTION (...) para especificar aspectos de processamento de consultas . Exemplos incluem SELECT FROM... OPTION(MERGE JOIN HASH, MAXRECURSION 10)).	ignorar

Hatch de escape	Descrição	Padrão
`escape_hatch_rowversion`	Controla o comportamento dos tipos de dados ROWVERSION e TIMESTAMP. Para ter mais informações, consulte Usar recursos do Babelfish com implementação limitada.	strict
`escape_hatch_schemabinding_function`	Controla o comportamento do Babelfish referente à cláusula WITH SCHEMABINDING. Por padrão, a cláusula WITH SCHEMABINDING é ignorada ao ser especificada com o comando CREATE ou ALTER FUNCTION.	ignorar
`escape_hatch_schemabinding_procedure`	Controla o comportamento do Babelfish referente à cláusula WITH SCHEMABINDING. Por padrão, a cláusula WITH SCHEMABINDING é ignorada ao ser especificada com o comando CREATE ou ALTER PROCEDURE.	ignorar
`escape_hatch_rowguidcol_column`	Controla o comportamento do Babelfish relacionado à cláusula ROWGUIDCOL ao criar ou modificar uma tabela.	strict

Hatch de escape	Descrição	Padrão
escape_hatch_schemabinding_trigger	Controla o comportamento do Babelfish referente à cláusula WITH SCHEMABINDING. Por padrão, a cláusula WITH SCHEMABINDING é ignorada ao ser especificada com o comando CREATE ou ALTER TRIGGER.	ignorar
escape_hatch_schemabinding_view	Controla o comportamento do Babelfish referente à cláusula WITH SCHEMABINDING. Por padrão, a cláusula WITH SCHEMABINDING é ignorada ao ser especificada com o comando CREATE ou ALTER VIEW.	ignorar
escape_hatch_session_settings	Controla o comportamento do Babelfish em relação a instruções SET em nível de sessão incompatíveis.	ignorar
escape_hatch_showplan_all	Controla o comportamento do Babelfish com relação a SET SHOWPLAN_ALL e SET STATISTICS PROFILE. Quando definidos para serem ignorados, eles se comportam como SET BABELFISH_SHOWPLAN_ALL e SET BABELFISH_STATISTICS PROFILE; quando definidos para serem estritos, são ignorados silenciosamente.	strict

Hatch de escape	Descrição	Padrão
`escape_hatch_storage_on_partition`	Controla o comportamento do Babelfish relacionado à cláusula `ON partition_scheme column` ao definir o particionamento. Atualmente, o Babelfish não implementa particionamento.	strict
`escape_hatch_storage_options`	Hatch de escape em qualquer opção de armazenamento utilizada em CREATE, ALTER DATABASE, TABLE, INDEX. Isso inclui cláusulas (LOG) ON, TEXTIMAGE_ON e FILESTREAM_ON que definem locais de armazenamento (partições e grupos de arquivos) para tabelas, índices e restrições e também para um banco de dados. Essa configuração de hatch de escape aplica-se a todas essas cláusulas (incluindo ON [PRIMARY] e ON “DEFAULT”). A exceção é quando uma partição é especificada para uma tabela ou um índice com ON partition_scheme (coluna).	ignorar
`escape_hatch_table_hints`	Controla o comportamento de dicas de tabelas especificadas utilizando a cláusula WITH (...).	ignorar

Hatch de escape	Descrição	Padrão
`escape_hatch_unique_constraint`	Quando definido como estrito, uma diferença semântica obscura entre o SQL Server e o PostgreSQL no tratamento de valores NULL em colunas indexadas pode gerar erros. A diferença semântica só surge em casos de uso não realistas, para que você possa definir esse hatch de escape como “ignorar” para evitar ver o erro. Obsoleto nas seguintes versões: 3.6.0 e versões posteriores, 4.2.0 e versões posteriores.	strict

Criar um cluster de banco de dados do Babelfish para Aurora PostgreSQL

O Babelfish para Aurora PostgreSQL é compatível com o Aurora PostgreSQL versão 13.4 e posteriores.

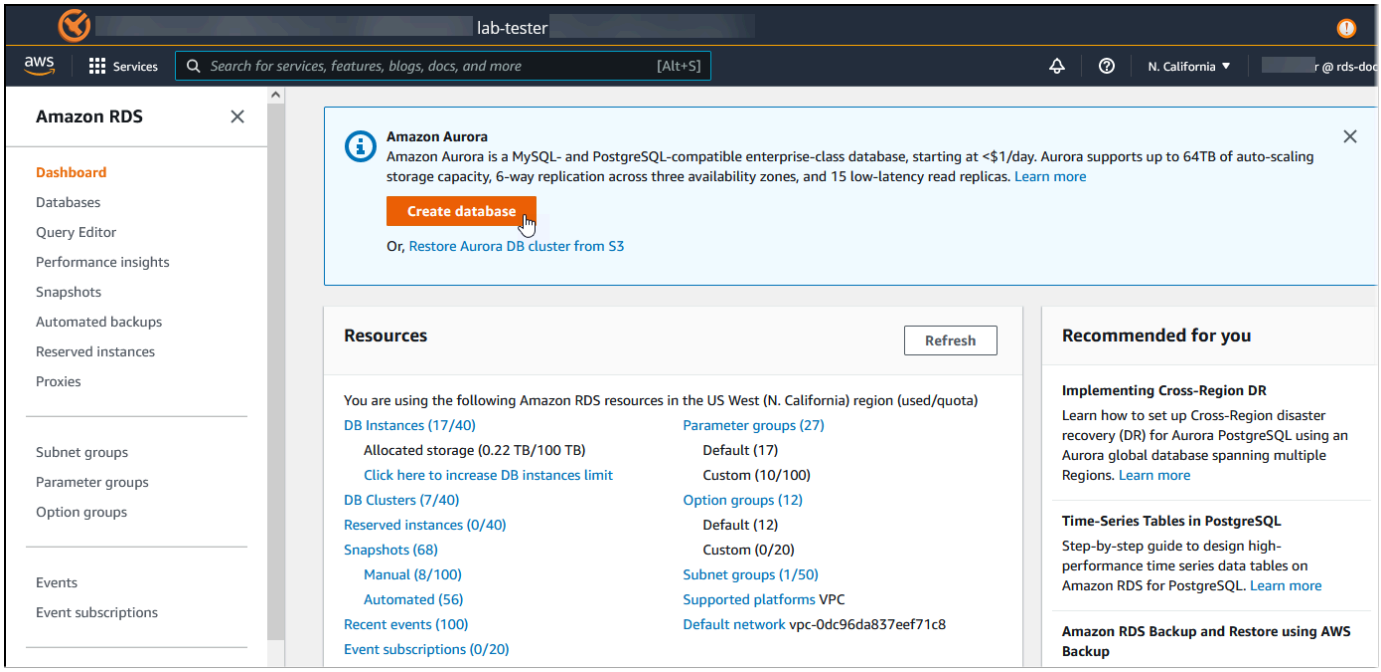
Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para criar um cluster do Aurora PostgreSQL com o Babelfish.

> **Note**
>
> Em um cluster do Aurora PostgreSQL, o nome do banco de dados `babelfish_db` é reservado para Babelfish. Criar seu próprio banco de dados “babelfish_db” em um Babelfish for Aurora PostgreSQL impede o Aurora de provisionar com êxito o Babelfish.

Console

Para criar um cluster com o Babelfish em execução com o Console de gerenciamento da AWS

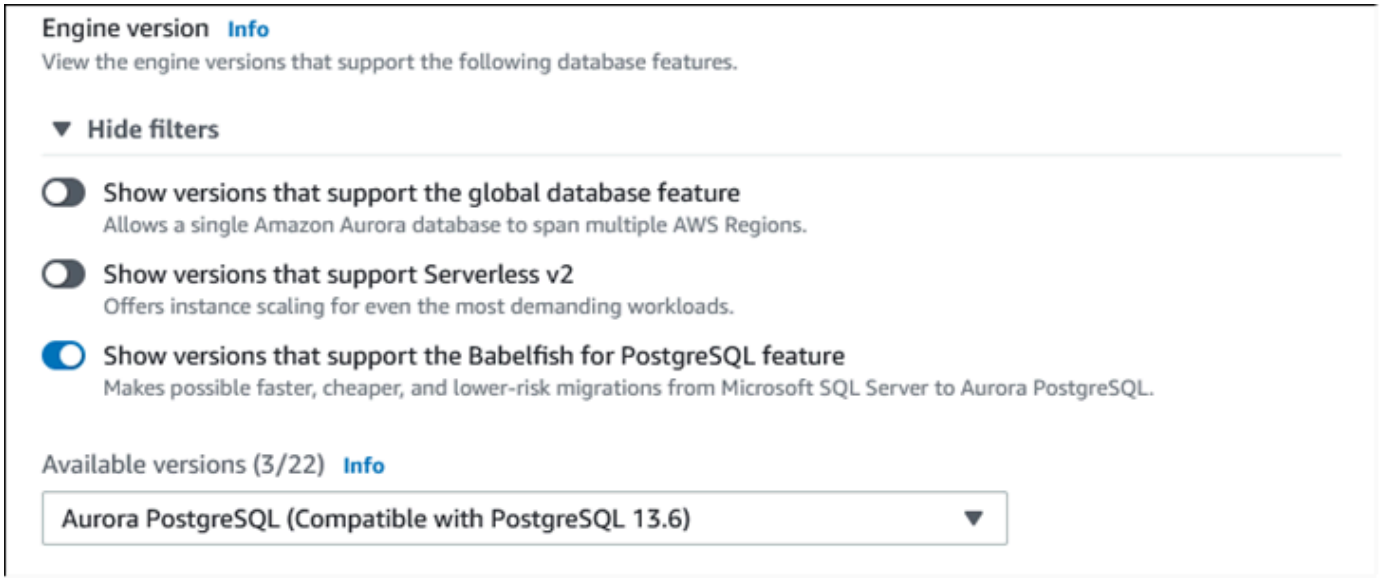
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/ e escolha Create database (Criar banco de dados).



2. Em Choose a database creation method (Escolha um método de criação de banco de dados), siga um destes procedimentos:

- Para especificar opções detalhadas do mecanismo, escolha Standard create (Criação padrão).
- Para utilizar opções pré-configuradas que oferecem suporte a práticas recomendadas para um cluster do Aurora, escolha Easy create (Criação fácil).

3. Em Tipo de mecanismo, escolha Aurora (compatível com PostgreSQL).
4. Selecione Show filters (Mostrar filtros) e depois Show versions that support the Babelfish for PostgreSQL feature (Mostrar versões compatíveis com o recurso Babelfish for PostgreSQL) para listar os tipos de motores que oferecem suporte para o Babelfish. Atualmente, o Babelfish é compatível com o Aurora PostgreSQL 13.4 e versões posteriores.
5. Para Available versions (Versões disponíveis), escolha uma versão do Aurora PostgreSQL. Para obter os recursos mais recentes do Babelfish, escolha a versão principal mais recente do Aurora PostgreSQL.

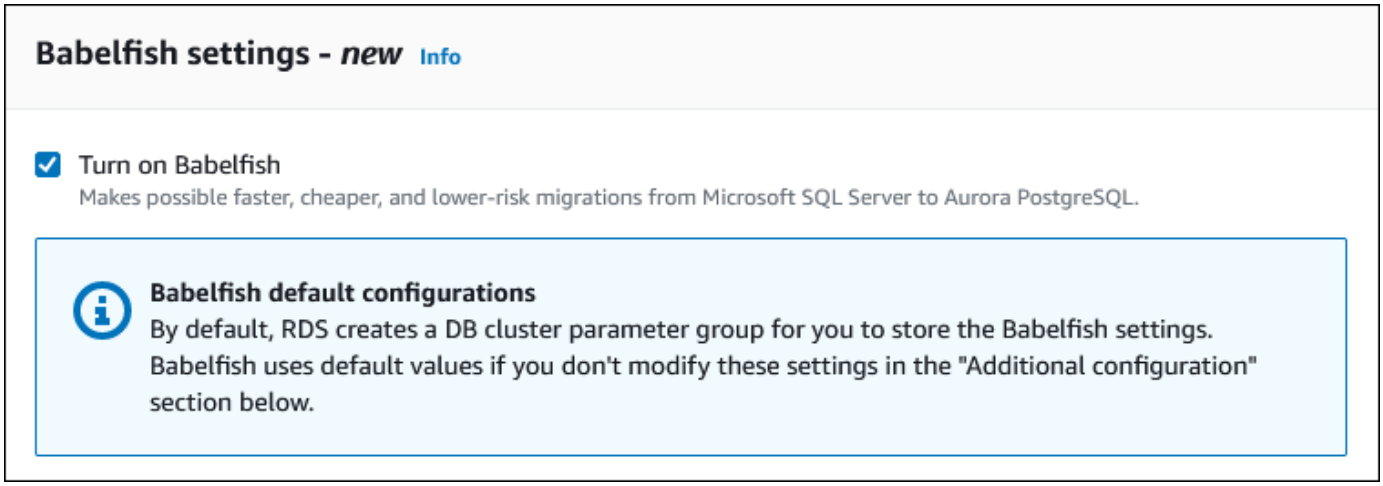


6. Para Templates (Modelos), escolha o modelo que corresponde ao seu caso de uso.
7. Para DB cluster identifier (Identificador do cluster de banco de dados), insira um nome que seja possível encontrar facilmente mais tarde na lista de clusters de banco de dados.
8. Para Master username (Nome de usuário primário), insira um nome de usuário de administrador. O valor padrão do Aurora PostgreSQL é `postgres`. Você pode aceitar o padrão ou escolher um nome diferente. Por exemplo, para seguir a convenção de nomenclatura usada em seus bancos de dados do SQL Server, você pode inserir `sa`(administrador do sistema) para o nome de usuário principal.

Se você não criar um usuário chamado `sa` nessa ocasião, poderá criá-lo mais tarde com sua opção de cliente. Depois de criar o usuário, use o comando `ALTER SERVER ROLE` para adicioná-lo ao grupo `sysadmin` (perfil) para o cluster.

> **Warning**
>
> O nome do usuário principal deve sempre usar caracteres minúsculos, caso contrário, o cluster de banco de dados não conseguirá se conectar ao Babelfish por meio da porta TDS.

9. Em Master password (Senha principal), crie uma senha forte e confirme-a.
10. Para as opções seguintes, até a seção Babelfish settings (Configurações do Babelfish), especifique as configurações do seu cluster de banco de dados. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.
11. Para disponibilizar a funcionalidade do Babelfish, marque a caixa Turn on Babelfish (Ativar o Babelfish).



12. Para DB cluster parameter group (Grupo de parâmetros de cluster de banco de dados), siga um destes procedimentos:

    - Escolha Create new (Criar novo) para criar um novo grupo de parâmetros com o Babelfish ativado.
    - Selecione Choose existing (Escolher existente) para utilizar um grupo de parâmetros existente. Se você utilizar um grupo existente, modifique esse grupo antes de criar o cluster e adicionar valores para os parâmetros do Babelfish. Para obter informações sobre os

parâmetros do Babelfish, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish.

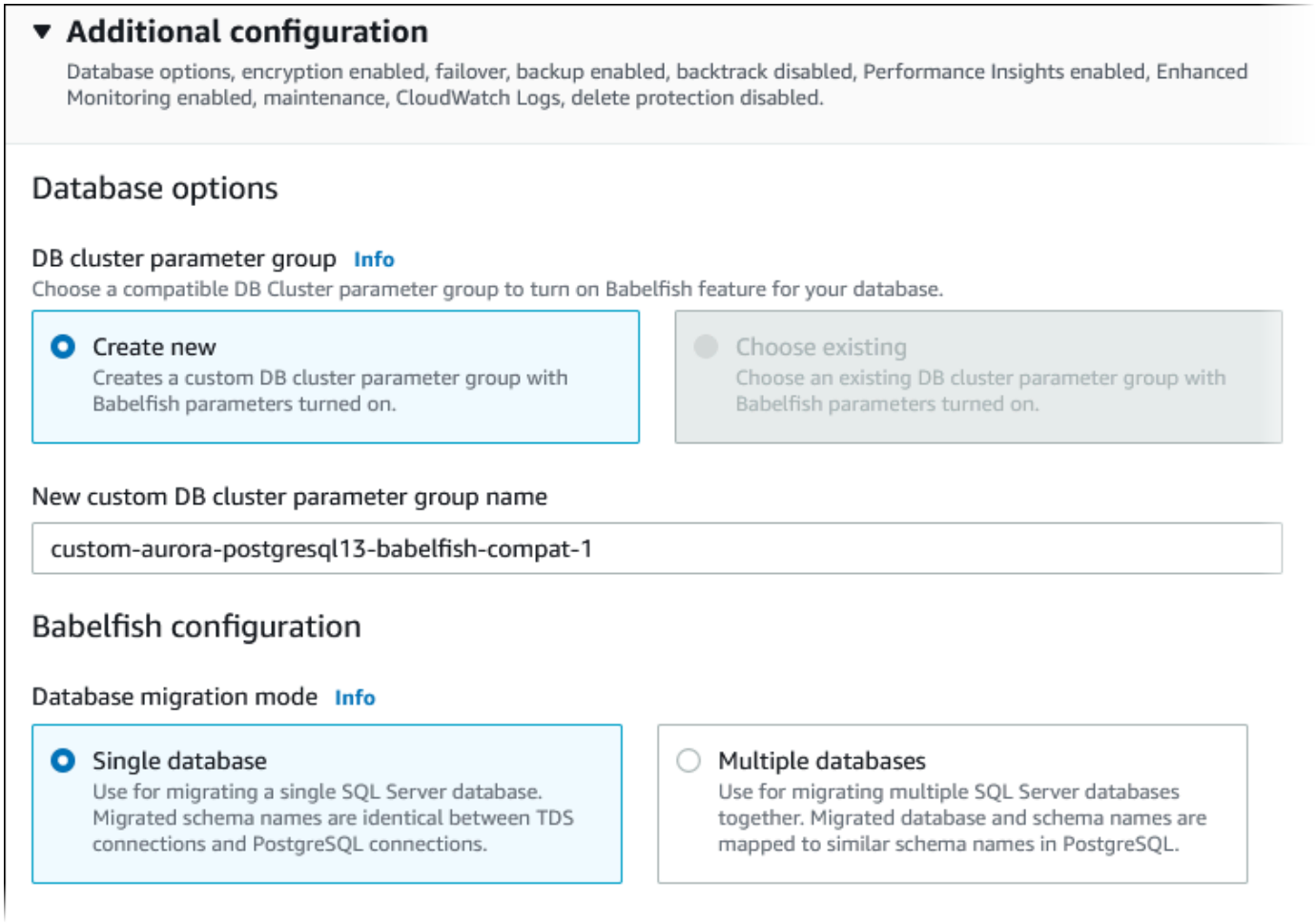
Se você utilizar um grupo existente, forneça o nome desse grupo na caixa seguinte.

13. Para Database migration mode (Modo de migração de banco de dados), escolha uma das seguintes opções:

    - Single database (Banco de dados único) para migrar um único banco de dados SQL Server.

      Em alguns casos, é possível migrar vários bancos de dados de usuários juntos quando seu objetivo final é uma migração completa para o Aurora PostgreSQL nativo sem o Babelfish. Se as aplicações finais exigirem esquemas consolidados (um único esquema dbo), certifique-se de primeiro consolidar seus bancos de dados do SQL Server em um único banco de dados do SQL Server. Em seguida, migre para o Babelfish utilizando o modo Single database (Banco de dados único).
    - Multiple databases (Vários bancos de dados) para migrar vários bancos de dados do SQL Server (originados de uma única instalação do SQL Server). O modo de banco de dados múltiplo não consolida vários bancos de dados que não são provenientes de uma única instalação do SQL Server. Para obter informações sobre como migrar vários bancos de dados, consulte Utilizar o Babelfish com um único banco de dados ou vários bancos de dados.

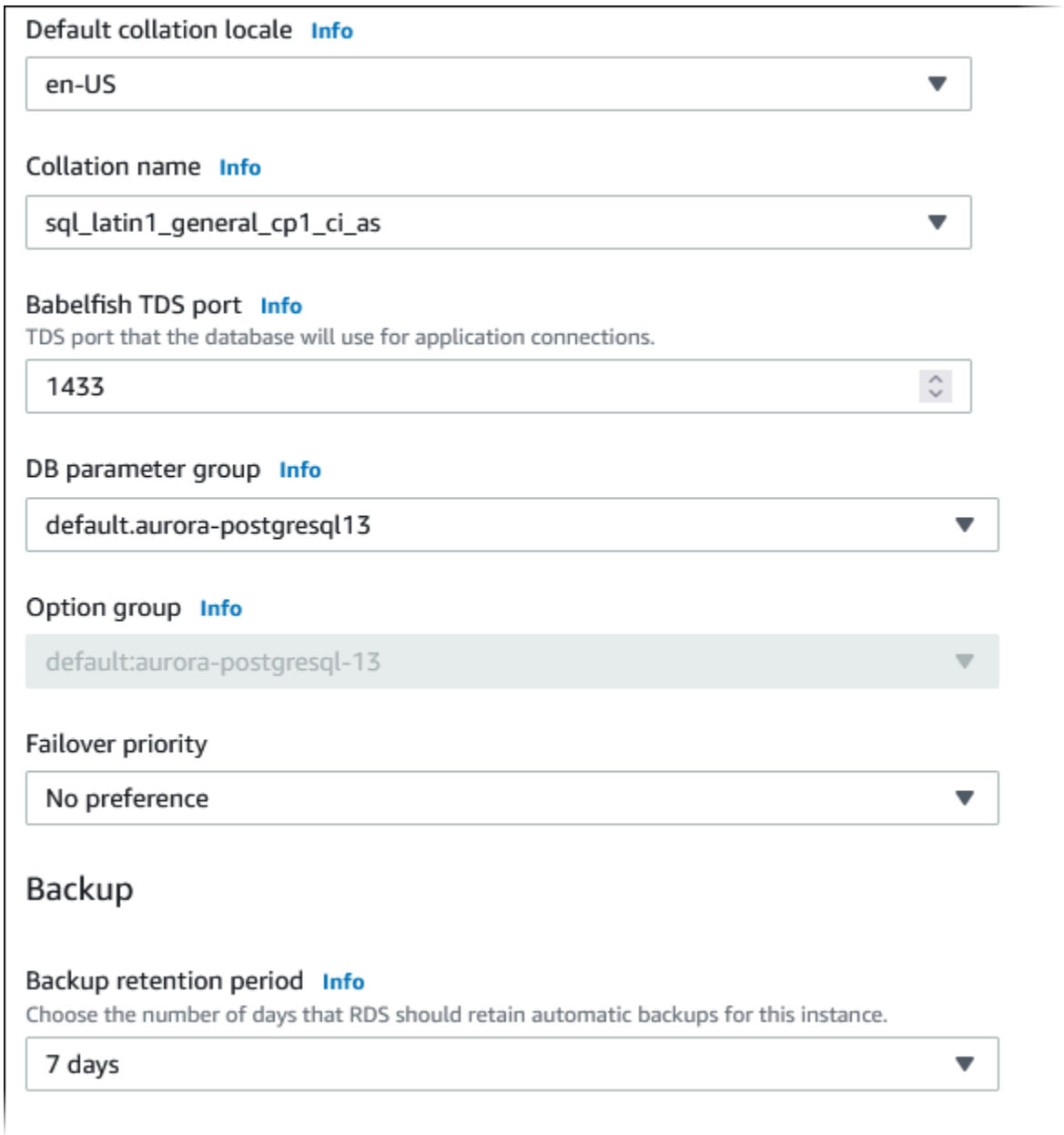
> **Note**
>
> A partir do Aurora PostgreSQL versão 16, vários bancos de dados são escolhidos por padrão como o modo de migração de banco de dados.



14. Para Default collation locale (Localidade do agrupamento padrão), insira a localidade do servidor. O padrão é `en-US`. Para obter informações detalhadas sobre agrupamentos, consulte Noções básicas sobre agrupamentos no Babelfish para Aurora PostgreSQL.
15. Para o campo Collation name (Nome do agrupamento), insira o agrupamento padrão. O padrão é `sql_latin1_general_cp1_ci_as`. Para obter informações detalhadas, consulte Noções básicas sobre agrupamentos no Babelfish para Aurora PostgreSQL.
16. Em Porta TDS do Babelfish, insira a porta padrão `1433`. No momento, o Babelfish só oferece suporte à porta `1433` para seu cluster de banco de dados.
17. Para DB parameter group (Grupo de parâmetros de banco de dados), escolha um grupo de parâmetros ou faça com que o Aurora crie um novo grupo para você com configurações padrão.
18. Para Failover priority (Prioridade de failover), escolha uma prioridade de failover para a instância. Se você não escolher um valor, o padrão será `tier-1`. Essa prioridade determina a ordem em que as réplicas do são promovidas durante a recuperação de uma falha de instância

primária. Para obter mais informações, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

19. Para Backup retention period (Período de retenção de backup), escolha por quanto tempo (de 1 a 35 dias) o Aurora retém as cópias de backup do banco de dados. Você pode usar cópias de backup para restaurações point-in-time (PITR) do banco de dados, contabilizando até os segundos. O período de retenção padrão é de sete dias.



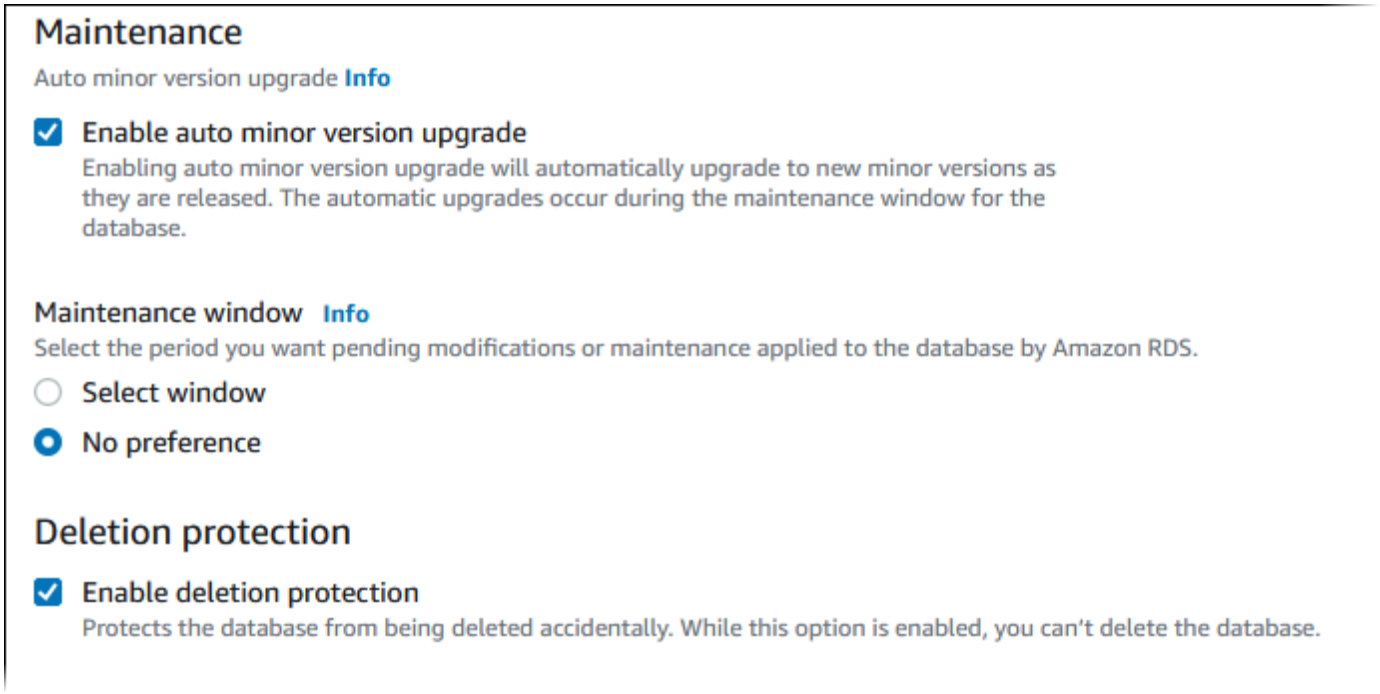
20. Escolha Copy tags to snapshots (Copiar etiquetas para snapshots) para copiar qualquer etiqueta da instância de banco de dados para um snapshot de banco de dados quando você cria um snapshot.

> **Note**
>
> Ao restaurar um cluster de banco de dados a partir de um snapshot, ele não é restaurado como um cluster de banco de dados do Babelfish para Aurora PostgreSQL. É necessário ativar os parâmetros que controlam as preferências do Babelfish no grupo de parâmetros do cluster de banco de dados para ativar o Babelfish novamente. Para ter mais informações sobre os parâmetros do Babelfish, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish.

21. Escolha Enable encryption (Habilitar criptografia) para ativar a criptografia em repouso (criptografia de armazenamento do Aurora) neste cluster de banco de dados.
22. Escolha Enable Performance Insights (Habilitar Performance Insights) para ativar o Amazon RDS Performance Insights.
23. Escolhe Enable enhanced monitoring (Habilitar monitoramento avançado) para iniciar a coleta de métricas em tempo real do sistema operacional em que o cluster de banco de dados é executado.
24. Escolha PostgreSQL log (Log do PostgreSQL) para publicar os arquivos de log no Amazon CloudWatch Logs.
25. Escolha Enable auto minor version upgrade (Habilitar upgrade automático da versão secundária) para atualizar automaticamente o cluster de bancos de dados Aurora quando um upgrade de versão secundária estiver disponível.
26. Para Maintenance window (Janela de manutenção), faça o seguinte:

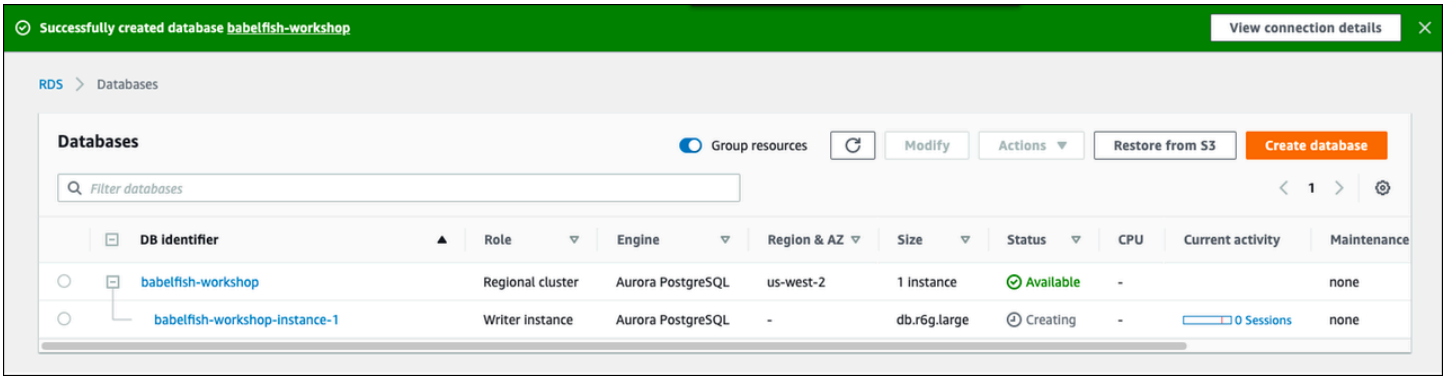
    - Para escolher um horário para o Amazon RDS fazer modificações ou manutenções, escolha Select window (Selecionar janela).
    - Para fazer a manutenção do Amazon RDS em um horário não programado, escolha No preference (Sem preferência).
27. Marque a caixa Enable deletion protection (Habilitar proteção contra exclusão) para evitar que o seu banco de dados seja excluído por acidente.

    Se você ativar esse recurso, não poderá excluir o banco de dados diretamente. Em vez disso, você precisará modificar o cluster de banco de dados e desativar esse recurso antes de excluir o banco de dados.



28. Selecione Criar banco de dados.

Você pode encontrar seu novo banco de dados configurado para o Babelfish na lista Databases (Bancos de dados). A coluna Status mostrará Available (Disponível) quando a implantação estiver concluída.



AWS CLI

Quando você cria um Babelfish para Aurora PostgreSQL usando aAWS CLI, precisa passar ao comando o nome do grupo de parâmetros de cluster de banco de dados a ser usado para o cluster. Para obter mais informações, consulte Pré-requisitos do cluster de banco de dados.

Para poder utilizar a AWS CLI para criar um cluster do Aurora PostgreSQL com o Babelfish, faça o seguinte:

- Escolha o URL do endpoint na lista de serviços em Endpoints e cotas do Amazon Aurora.
- Crie um grupo de parâmetros para o cluster. Para obter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.
- Modifique o grupo de parâmetros adicionando o parâmetro que ativa o Babelfish.

Para criar um cluster de bancos de dados Aurora PostgreSQL com o Babelfish utilizando a AWS CLI

Os exemplos a seguir usam o nome de usuário principal padrão, `postgres`. Se necessário, substitua-o pelo nome de usuário criado para o cluster de banco de dados, como `sa` ou qualquer nome de usuário escolhido se não aceitou o padrão.

1. Criar um grupo de parâmetros.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --db-parameter-group-family aurora-postgresql14 \
   --description "description"
   ```

   Para Windows:

   ```
   aws rds create-db-cluster-parameter-group ^
   --endpoint-url endpoint-URL ^
   --db-cluster-parameter-group-name parameter-group ^
   --db-parameter-group-family aurora-postgresql14 ^
   --description "description"
   ```

2. Modifique seu grupo de parâmetros para ativar o Babelfish.

   Para Linux, macOS ou Unix:

   ```
   aws rds modify-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --parameters
    "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
   ```

   Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
--endpoint-url endpoint-url ^
--db-cluster-parameter-group-name paramater-group ^
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

3. Identifique o grupo de sub-redes de banco de dados e o ID do grupo de segurança da nuvem privada virtual (VPC) do seu novo cluster de banco de dados. Em seguida, chame o comando create-db-cluster.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster \
   --db-cluster-identifier  cluster-name\
   --master-username postgres \
   --manage-master-user-password \
   --engine aurora-postgresql \
   --engine-version 14.3            \
   --vpc-security-group-ids security-group \
   --db-subnet-group-name subnet-group-name \
   --db-cluster-parameter-group-name parameter-group
   ```

   Para Windows:

   ```
   aws rds create-db-cluster ^
   --db-cluster-identifier cluster-name ^
   --master-username postgres ^
   --manage-master-user-password ^
   --engine aurora-postgresql ^
   --engine-version 14.3 ^
   --vpc-security-group-ids security-group ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-parameter-group-name parameter-group
   ```

   Este exemplo especifica a opção `--manage-master-user-password` para gerar a senha mestra do usuário e gerenciá-la no Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

4. Crie explicitamente a instância primária do cluster de banco de dados. Use o nome do cluster criado na etapa 3 para o argumento `--db-cluster-identifier` ao chamar o comando create-db-instance, conforme mostrado a seguir.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-instance \
   --db-instance-identifier instance-name \
   --db-instance-class db.r6g \
   --db-subnet-group-name subnet-group \
   --db-cluster-identifier cluster-name \
   --engine aurora-postgresql
   ```

   Para Windows:

   ```
   aws rds create-db-instance ^
   --db-instance-identifier instance-name ^
   --db-instance-class db.r6g ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-identifier cluster-name ^
   --engine aurora-postgresql
   ```

Migrar um banco de dados do SQL Server para o Babelfish para Aurora PostgreSQL

É possível usar o Babelfish para Aurora PostgreSQL para migrar um banco de dados do SQL Server para um cluster de bancos de dados do Amazon Aurora PostgreSQL. Antes da migração, leia Utilizar o Babelfish com um único banco de dados ou vários bancos de dados.

Tópicos

- Visão geral do processo de migração
- Avaliar e lidar com diferenças entre o SQL Server e o Babelfish
- Ferramentas de importação/exportação para realizar a migração do SQL Server para o Babelfish

Visão geral do processo de migração

O resumo a seguir lista as etapas necessárias para migrar com êxito sua aplicação SQL Server e fazê-la funcionar com o Babelfish. Para obter informações sobre as ferramentas que você pode usar para os processos de exportação e importação e para obter mais detalhes, consulte Ferramentas de importação/exportação para realizar a migração do SQL Server para o Babelfish. Para carregar os dados, recomendamos usar o AWS DMS com um cluster de banco de dados do Aurora PostgreSQL como o endpoint de destino.

1. Crie um cluster de banco de dados do Aurora PostgreSQL com o Babelfish ativado. Para saber como, consulte Criar um cluster de banco de dados do Babelfish para Aurora PostgreSQL.

   Para importar os vários artefatos SQL exportados do banco de dados do SQL Server, conecte-se ao cluster do Babelfish usando uma ferramenta do SQL Server, como sqlcmd. Para obter mais informações, consulte Utilizar um cliente SQL Server para se conectar ao seu cluster de banco de dados.
2. No banco de dados do SQL Server que você deseja migrar, exporte a linguagem de definição de dados (DDL). A DDL é um código SQL que descreve objetos de banco de dados que contêm dados de usuários (como tabelas, índices e visualizações) e código de banco de dados escrito pelo usuário (como procedimentos armazenados, funções definidas pelo usuário e acionadores).

   Para obter mais informações, consulte Usar o SQL Server Management Studio (SSMS) para migrar para o Babelfish.
3. Execute uma ferramenta de avaliação para avaliar o escopo de quaisquer alterações que talvez você precise fazer para que o Babelfish possa oferecer compatibilidade com a aplicação em

execução no SQL Server de forma eficiente. Para obter mais informações, consulte Avaliar e lidar com diferenças entre o SQL Server e o Babelfish.

4. Analise as limitações do endpoint de destino do AWS DMS e atualize o script de DDL conforme necessário. Consulte mais informações em “Limitations to using a PostgreSQL target endpoint with Babelfish tables” em Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service.
5. No novo cluster de banco de dados do Babelfish, execute a DDL em seu banco de dados T-SQL especificado para criar apenas os esquemas, os tipos de dados definidos pelo usuário e as tabelas com suas restrições de chave primária.
6. Use o AWS DMS para migrar seus dados do SQL Server para as tabelas do Babelfish. Para replicação contínua usando o SQL Server Change Data Capture ou o SQL Replication, use o Aurora PostgreSQL em vez do Babelfish como endpoint. Para fazer isso, consulte Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service.
7. Quando o carregamento de dados for concluído, crie todos os objetos T-SQL restantes que são compatíveis com a aplicação em seu cluster do Babelfish.
8. Reconfigure sua aplicação cliente para se conectar ao endpoint do Babelfish em vez do banco de dados SQL Server. Para obter mais informações, consulte Conectar-se a um cluster de banco de dados do Babelfish.
9. Modifique sua aplicação se necessário e teste novamente. Para obter mais informações, consulte Diferenças entre o Babelfish para Aurora PostgreSQL e o SQL Server.

Você ainda precisa avaliar suas consultas SQL do lado do cliente. Os esquemas gerados a partir da instância do SQL Server convertem somente o código SQL do lado do servidor. Recomendamos que você execute as seguintes etapas:

- Capture consultas do lado do cliente usando o SQL Server Profiler com o modelo predefinido TSQL_Replay. Esse modelo captura informações de instrução T-SQL que podem ser reproduzidas para ajuste e teste iterativos. Você pode iniciar o profiler no SQL Server Management Studio, no menu Tools (Ferramentas). Selecione SQL Server Profiler para abrir o profiler e escolher o modelo TSQL_replay.

  Para usar para a migração do Babelfish, inicie um rastreamento e execute a aplicação usando seus testes funcionais. O profiler captura as instruções T-SQL. Ao terminar o teste, interrompa o rastreamento. Salve o resultado em um arquivo XML com suas consultas do lado do cliente (File (Arquivo) > Save as (Salvar como) > Trace XML File for Replay (Rastrear arquivo XML para repetição)).

Para obter mais informações, consulte SQL Server Profiler na documentação da Microsoft. Para obter mais informações sobre o modelo TSQL_Replay, consulte Modelos do SQL Server Profiler.

- Para aplicações com consultas SQL complexas no lado do cliente, recomendamos que você use o Babelfish Compass para analisar a compatibilidade delas com o Babelfish. Se a análise indicar que as instruções SQL no lado do cliente contêm recursos SQL incompatíveis, revise os aspectos SQL na aplicação cliente e faça modificações, se necessário.
- Você também pode capturar as consultas SQL como eventos estendidos (formato .xel). Para isso, use o SSMS XEvent Profiler. Depois de gerar o arquivo .xel, extraia as instruções SQL em arquivos .xml que o Compass poderá processar. Para obter mais informações, consulte Use the SSMS XEvent Profler (Usar o SSMS XEvent Profler) na documentação da Microsoft.

Quando estiver satisfeito com todos os testes, análises e quaisquer modificações necessárias para a aplicação migrada, você poderá começar a usar o banco de dados do Babelfish para produção. Para fazer isso, interrompa o banco de dados original e redirecione aplicações cliente ativas para utilizar a porta do TDS do Babelfish.

> **Note**
>
> Agora o AWS DMS permite a replicação de dados do Babelfish. Consulte mais informações em AWS DMS now supports Babelfish for Aurora PostgreSQL as a source.

Avaliar e lidar com diferenças entre o SQL Server e o Babelfish

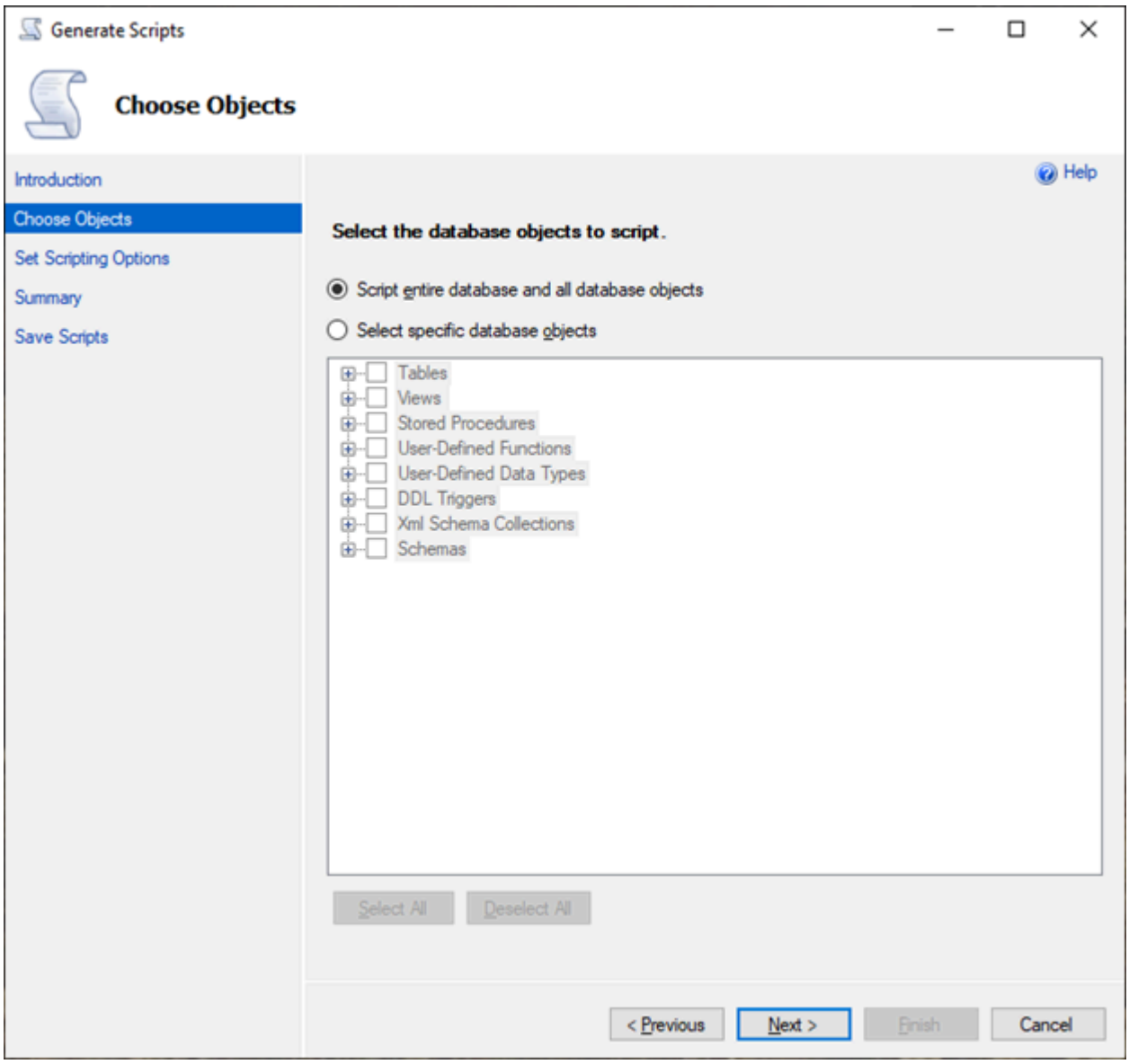
Para obter melhores resultados, recomendamos que você avalie a DDL/DML gerada e o código de consulta do cliente antes de realmente migrar sua aplicação de banco de dados SQL Server para o Babelfish. Dependendo da versão do Babelfish e dos recursos específicos do SQL Server implementados pela aplicação, talvez seja necessário refatorar sua aplicação ou usar alternativas para funcionalidades que ainda não são totalmente compatíveis com o Babelfish.

- Para avaliar o código de sua aplicação SQL Server, use o Babelfish Compass na DDL gerada para determinar quanto código T-SQL é compatível com o Babelfish. Identifique o código T-SQL que talvez precise de modificações antes de ser executado no Babelfish. Para obter mais informações sobre essa ferramenta, consulte Ferramenta Babelfish Compass no GitHub.

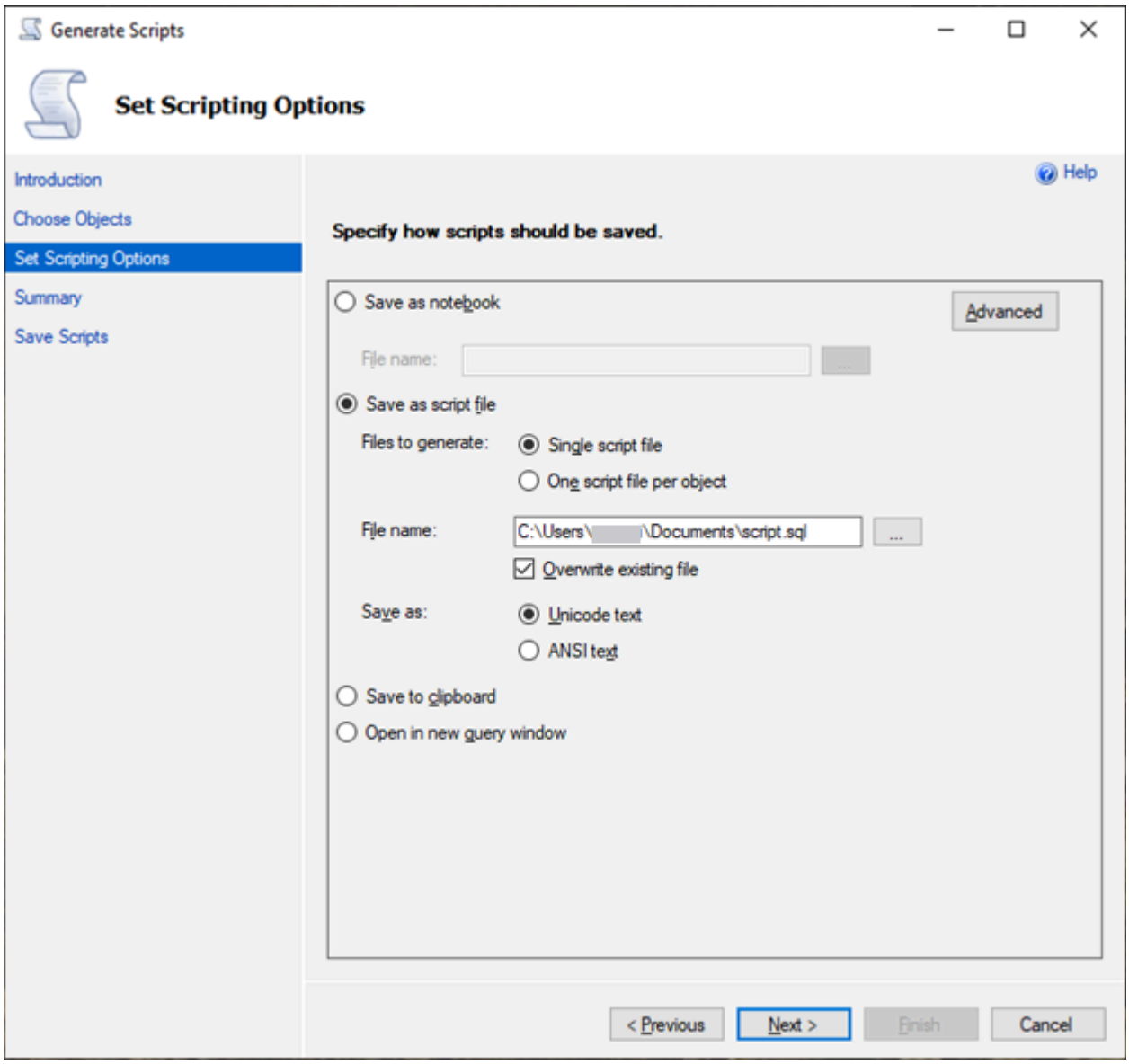
> **Note**
>
> O Babelfish Compass é uma ferramenta de software livre. Relate quaisquer problemas com o Babelfish Compass pelo GitHub e não pelo AWS Support.

Você pode usar o Generate Script Wizard com o SQL Server Management Studio (SSMS) para gerar o arquivo SQL que é avaliado pelo Babelfish Compass ou a AWS Schema Conversion Tool CLI. Recomendamos as etapas a seguir para agilizar a avaliação.

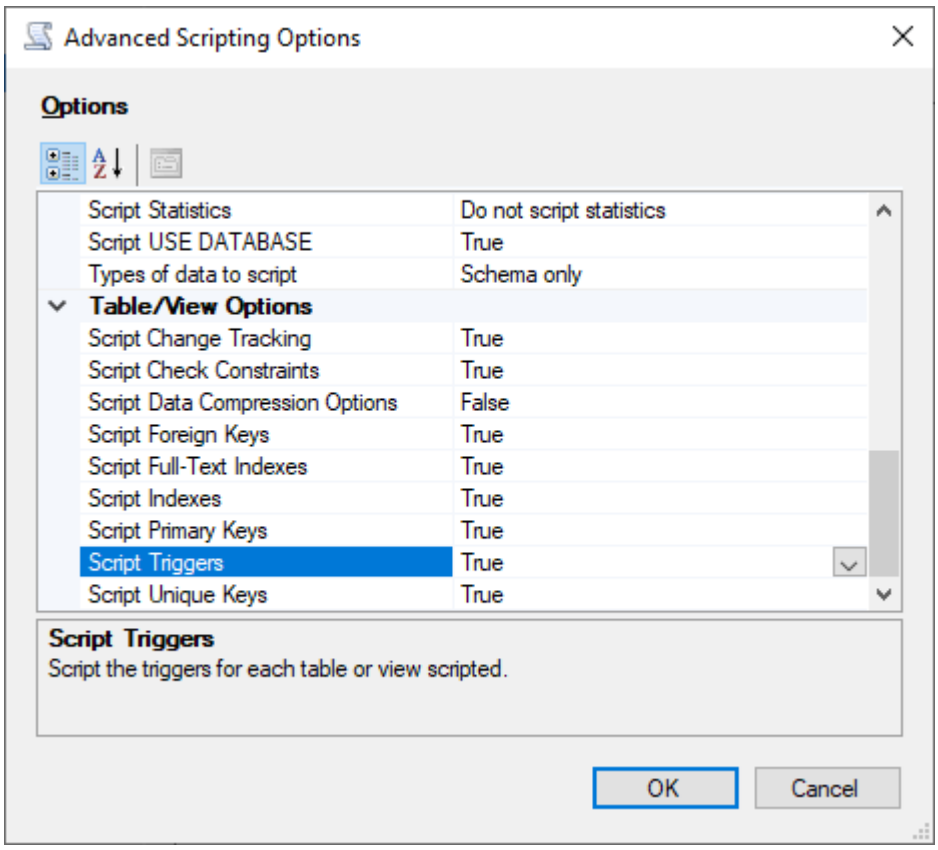
1. Na página Choose Objects (Escolher objetos), selecione Script entire database and all database objects (Criar script para todo o banco de dados e todos os objetos do banco de dados).



2. Em Set Scripting Options (Definir opções de script), selecione Save as script file (Salvar como arquivo de script) como um Single script file (Único arquivo de script).



3. Selecione Advanced (Avançado) para alterar as opções de script padrão para identificar recursos que normalmente são definidos como falsos para uma avaliação completa:
   - Rastreamento de alterações de script como True
   - Índices de texto de script completo como True
   - O script se torna verdadeiro
   - O script faz login em True
   - Proprietário do script para True
   - Permissões em nível de objeto de script como True
   - Agrupamentos de scripts como True



4. Execute as etapas restantes no assistente para gerar o arquivo.

Ferramentas de importação/exportação para realizar a migração do SQL Server para o Babelfish

Recomendamos usar o AWS DMS como ferramenta principal para realizar a migração do SQL Server para o Babelfish. No entanto, o Babelfish é compatível com várias outras formas de migrar dados usando as ferramentas do SQL Server que incluem as seguintes.

- SQL Server Integration Services (SSIS) para todas as versões do Babelfish. Para obter mais informações, consulte Migrate from SQL Server to Aurora PostgreSQL using SSIS and Babelfish (Migrar do SQL Server para o Aurora PostgreSQL usando SSIS e Babelfish).
- Use o SSMS Import/Export Wizard para as versões 2.1.0 e posteriores do Babelfish. Essa ferramenta está disponível por meio do SSMS, mas também está disponível como uma ferramenta autônoma. Para obter mais informações, consulte Welcome to SQL Server Import and Export Wizard (Bem-vindo ao assistente de importação e exportação do SQL Server) na documentação da Microsoft.

- O utilitário Microsoft bcp permite copiar dados de uma instância do Microsoft SQL Server em um arquivo de dados no formato especificado. Para obter mais informações, consulte bcp Utility na documentação da Microsoft. O Babelfish agora é compatível com a migração de dados usando o cliente BCP, e o utilitário bcp agora é compatível com o sinalizador `-E` (para colunas de identidade) e o sinalizador -b (para inserções em lote). Algumas opções do bcp não são compatíveis, por exemplo, `-C`, `-T`, `-G`, `-K`, `-R`, `-V` e `-h`.

Usar o SQL Server Management Studio (SSMS) para migrar para o Babelfish

Recomendamos gerar arquivos separados para cada um dos tipos de objeto específicos. Você pode usar primeiro o Generate Scripts Wizard no SSMS para cada conjunto de instruções DDL e depois modificar os objetos como um grupo para corrigir quaisquer problemas encontrados durante a avaliação.

Execute estas etapas para migrar os dados usando o AWS DMS ou outros métodos de migração de dados. Execute primeiro esses tipos de script de criação para uma abordagem melhor e mais rápida de carregar os dados nas tabelas do Babelfish no Aurora PostgreSQL.

1. Execute instruções `CREATE SCHEMA`.
2. Execute instruções `CREATE TYPE` para criar tipos de dados definidos pelo usuário.
3. Execute instruções `CREATE TABLE` básicas com as chaves primárias ou restrições exclusivas.

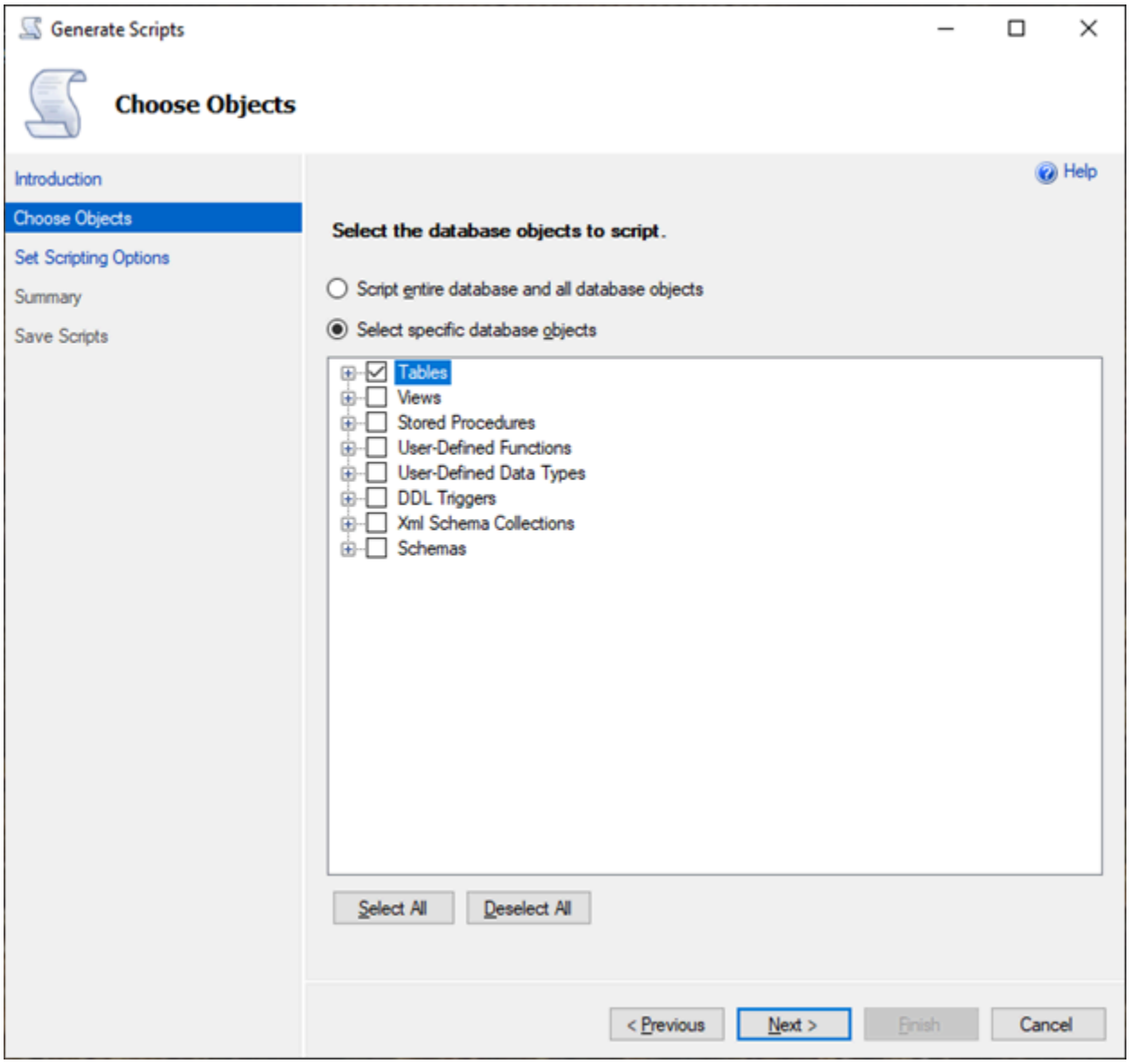
Execute o carregamento de dados usando a ferramenta de importação/exportação recomendada. Execute os scripts modificados para as etapas a seguir para adicionar os objetos restantes do banco de dados. Você precisa das instruções create table para executar esses scripts para as restrições, acionadores e índices. Depois que os scripts forem gerados, exclua as instruções create table.

1. Execute instruções `ALTER TABLE` para as restrições de verificação, restrições de chave externa e restrições padrão.
2. Execute instruções `CREATE TRIGGER`.
3. Execute instruções `CREATE INDEX`.
4. Execute instruções `CREATE VIEW`.
5. Execute instruções `CREATE STORED PROCEDURE`.

Como gerar scripts para cada tipo de objeto

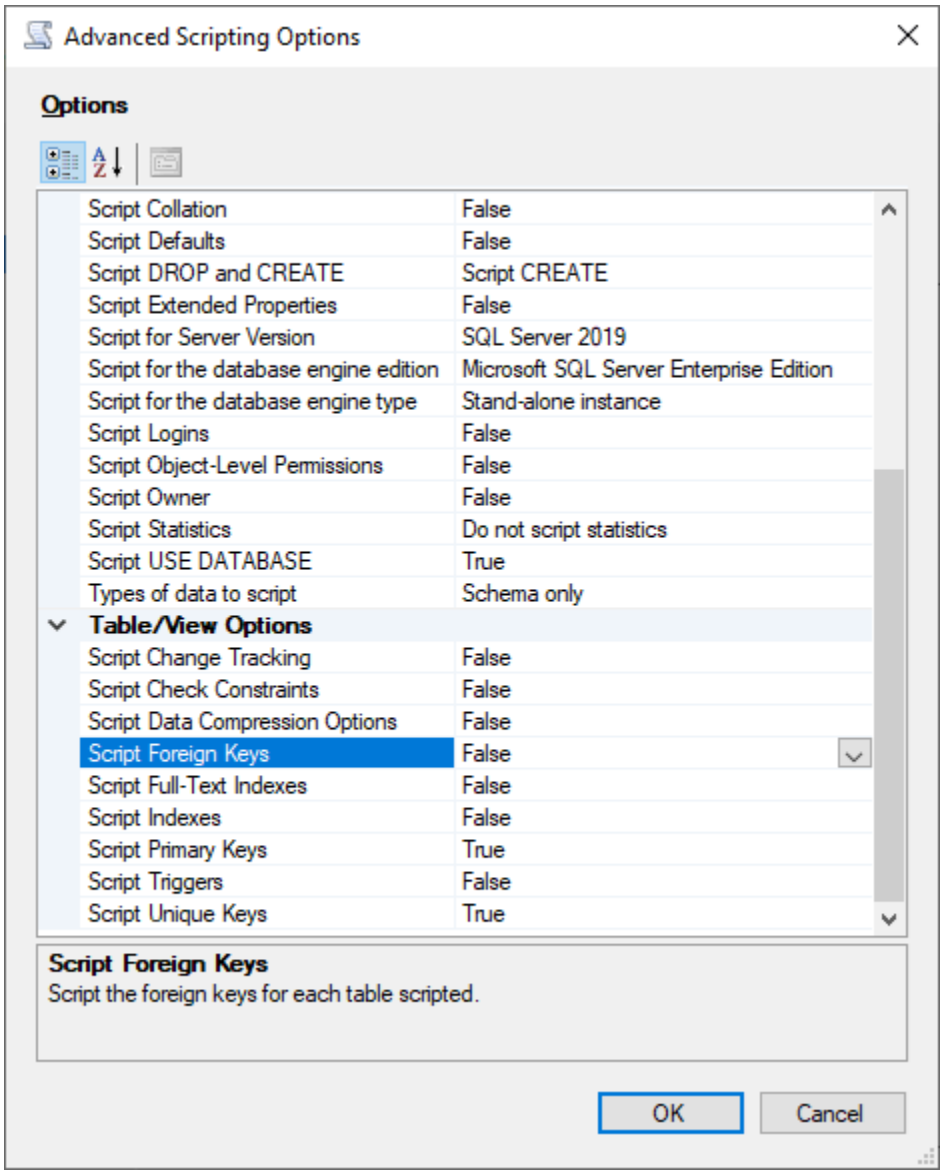
Use as etapas a seguir para criar instruções básicas de criação de tabela usando o Generate Scripts Wizard no SSMS. Siga estas etapas para gerar scripts para os diferentes tipos de objeto.

1. Conecte-se à sua instância do SQL Server existente.
2. Abra o menu de contexto (clique com o botão direito do mouse) para um nome de banco de dados.
3. Selecione Tasks (Tarefas) e, depois, Generate Scripts... (Gerar scripts...).
4. No painel Choose Objects (Escolher objetos), escolha Select specific database objects (Selecionar objetos de banco de dados específicos). Escolha Tables (Tabelas), selecione todas as tabelas. Escolha Próximo para continuar.



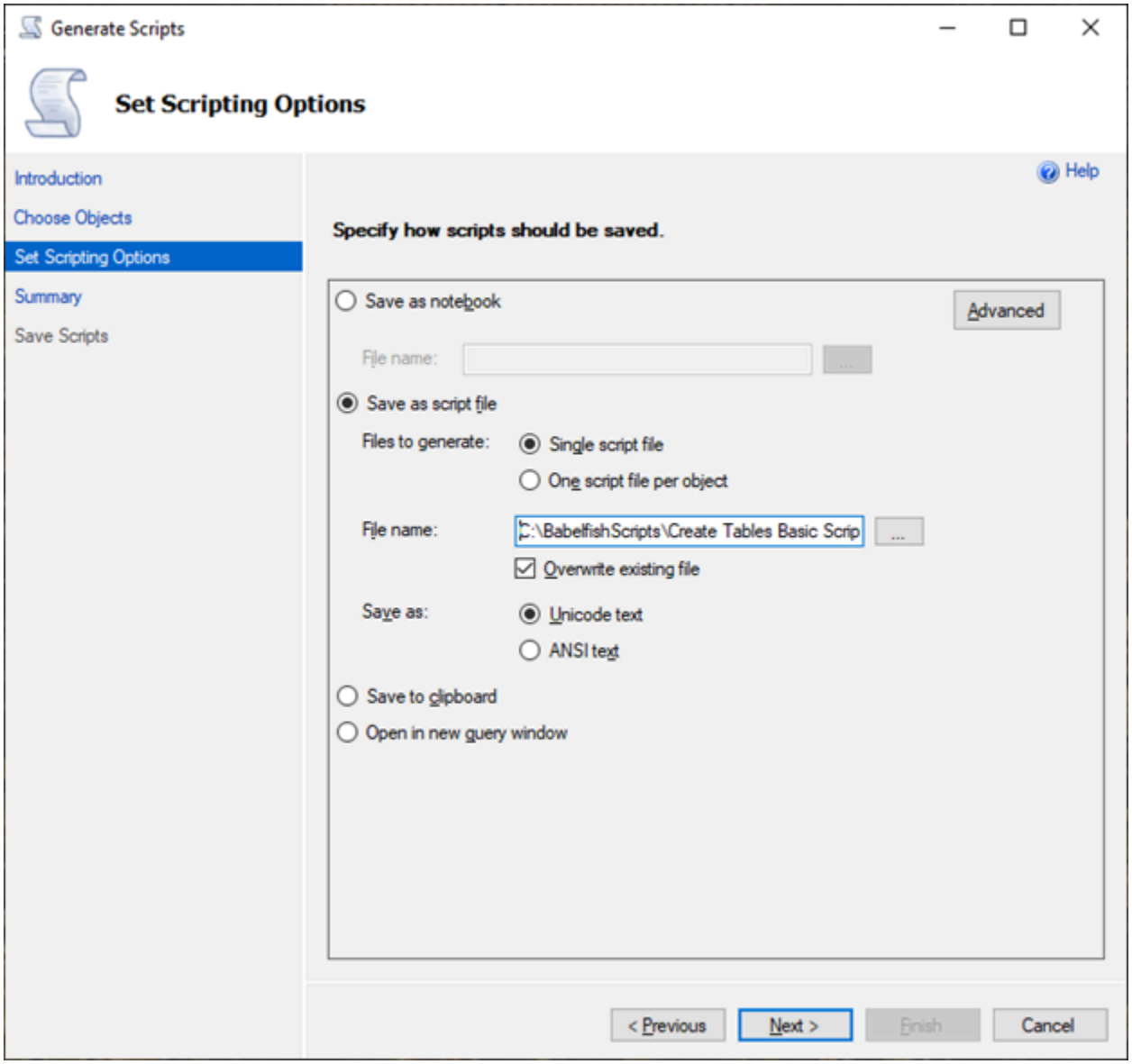
5. Na página Set Scripting Options (Definir opções de script), escolha Advanced (Avançado) para abrir as configurações de Options (Opções). Para gerar as instruções create table básicas, altere os seguintes valores padrão:

   - O script assume como padrão False.
   - Propriedades estendidas de script como False. O Babelfish não é compatível com propriedades estendidas.
   - Restrições de conferência de script como False. Chaves externas de script como False.



6. Escolha OK.

7. Na página Set Scripting Options (Definir opções de script), selecione Save as script file (Salvar como arquivo de script) e Single script file (Único arquivo de script). Insira seu File name (Nome de arquivo).



8. Selecione Next (Próximo) para visualizar a página Summary wizard (Assistente de resumo).
9. Selecione Next (Próximo) para iniciar a geração do script.

   É possível continuar gerando scripts para os outros tipos de objetos no assistente. Em vez de selecionar Finish (Concluir) depois que o arquivo for salvo, selecione o botão Previous (Anterior) três vezes para voltar à página Choose Objects (Selecionar objetos). Depois, repita as etapas no assistente para gerar scripts para os outros tipos de objeto.

Autenticação de banco de dados com o Babelfish para Aurora PostgreSQL

O Babelfish para Aurora PostgreSQL permite duas maneiras de autenticar usuários do banco de dados. A autenticação por senha está disponível por padrão para todos os clusters de banco de dados do Babelfish. Também é possível adicionar a autenticação do Kerberos para o mesmo cluster de banco de dados.

Tópicos

- Autenticação de senha com o Babelfish
- Autenticação do Kerberos com o Babelfish
- Configurar a autenticação do Kerberos usando grupos de segurança do Active Directory para Babelfish

Autenticação de senha com o Babelfish

O Babelfish para Aurora PostgreSQL é compatível com a autenticação por senha. As senhas são armazenadas em formato criptografado no disco. Para obter mais informações sobre a autenticação em um cluster do Aurora PostgreSQL, consulte Segurança com o Amazon Aurora PostgreSQL.

Podem ser solicitadas credenciais sempre que você se conectar ao Babelfish. Qualquer usuário migrado ou criado no Aurora PostgreSQL pode utilizar as mesmas credenciais na porta do SQL Server e na porta do PostgreSQL. O Babelfish não impõe políticas de senha, mas recomendamos o seguinte:

- Exija uma senha complexa com pelo menos oito (8) caracteres.
- Aplique uma política de validade de senha.

Para rever uma lista completa de usuários de banco de dados, utilize o comando `SELECT * FROM pg_user;`.

Autenticação do Kerberos com o Babelfish

A versão 15.2 do Babelfish para Aurora PostgreSQL é compatível com a autenticação em seu cluster de banco de dados usando Kerberos. Esse método possibilita usar a autenticação do Microsoft Windows para autenticar usuários quando eles se conectam ao banco de dados do Babelfish. Para fazer isso, configure o cluster de banco de dados para usar o AWS Directory Service for Microsoft Active Directory para autenticação Kerberos. Para ter mais informações, consulte O que é Directory Service? no Guia de administração do AWS Directory Service.

Configurar a autenticação Kerberos

O cluster de banco de dados do Babelfish para Aurora PostgreSQL pode se conectar usando duas portas diferentes, mas a configuração da autenticação Kerberos é um processo único. Portanto, você precisa primeiro configurar a autenticação Kerberos para seu cluster de banco de dados. Para ter mais informações, consulte Configurar a autenticação Kerberos. Depois de concluir a configuração, verifique se você pode se conectar a um cliente PostgreSQL usando o Kerberos. Para ter mais informações, consulte Conectar-se com a autenticação Kerberos.

Login e provisionamento de usuários no Babelfish

Os logins do Windows criados por meio da porta Tabular Data Stream (TDS) podem ser usados com a porta TDS ou a porta PostgreSQL. Primeiro, o login que podem usar o Kerberos para autenticação devem ser provisionados por meio da porta TDS antes de serem usados por usuários e aplicações do T-SQL para conexão com um banco de dados Babelfish. Ao criar logins do Windows, os administradores podem fornecer o login usando o nome do domínio DNS ou o nome do domínio NetBIOS. Normalmente, o domínio NetBIOS é o subdomínio do nome do domínio DNS. Por exemplo, se o nome do domínio DNS for `CORP.EXAMPLE.COM`, o domínio do NetBIOS poderá ser `CORP`. Se o formato do nome de domínio do NetBIOS for fornecido para um login, deverá existir um mapeamento para o nome do domínio DNS.

Gerenciar o nome de domínio do NetBIOS para o mapeamento do nome do domínio DNS

Para gerenciar mapeamentos entre o nome de domínio do NetBIOS e o nome do domínio DNS, o Babelfish fornece procedimentos armazenados no sistema para adicionar, remover e truncar mapeamentos. Somente um usuário com um perfil `sysadmin` pode executar esses procedimentos.

Para criar mapeamento entre o NetBIOS e o nome do domínio DNS, use o procedimento armazenado do sistema fornecido pelo Babelfish `babelfish_add_domain_mapping_entry`. Os dois argumentos devem ter um valor válido e não ser NULL.

Example

```
EXEC babelfish_add_domain_mapping_entry 'netbios_domain_name',
 'fully_qualified_domain_name'
```

O exemplo a seguir mostra como criar o mapeamento entre o nome NetBIOS CORP e o nome do domínio DNS CORP.EXAMPLE.COM.

Example

```
EXEC babelfish_add_domain_mapping_entry 'corp', 'corp.example.com'
```

Para excluir uma entrada de mapeamento existente, use o procedimento armazenado no sistema babelfish_remove_domain_mapping_entry.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'netbios_domain_name'
```

O exemplo a seguir mostra como remover o mapeamento para o NetBIOS nome CORP.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'corp'
```

Para remover todas as entradas de mapeamento existentes, use o procedimento armazenado no sistema babelfish_truncate_domain_mapping_table:

Example

```
EXEC babelfish_truncate_domain_mapping_table
```

Para visualizar todos os mapeamentos entre o NetBIOS e o nome do domínio DNS, use a consulta a seguir.

Example

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

Gerenciar logins

Criar logins

Conecte-se ao banco de dados por meio do endpoint TDS usando um login que tenha as permissões corretas. Se não houver nenhum usuário do banco de dados criado para o login, o login será mapeado para o usuário convidado. Se o usuário convidado não estiver habilitado, a tentativa de login falhará.

Crie um login do Windows usando a consulta a seguir. A opção `FROM WINDOWS` permite a autenticação usando o Active Directory.

```
CREATE LOGIN login_name FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Example

O exemplo a seguir mostra a criação de um login para o usuário do Active Directory [corp\test1] com um banco de dados padrão de db1.

```
CREATE LOGIN [corp\test1] FROM WINDOWS WITH DEFAULT_DATABASE=db1
```

Este exemplo pressupõe que haja um mapeamento entre o domínio do NetBIOS CORP e o nome do domínio DNS CORP.EXAMPLE.COM. Se não houver mapeamento, você deverá fornecer o nome do domínio DNS [CORP.EXAMPLE.COM\test1].

> **Note**
>
> Os logins baseados em usuários do Active Directory são limitados a nomes com até 21 caracteres.

Cancelar o login

Para cancelar um login, use a mesma sintaxe de qualquer login, conforme mostrado no seguinte exemplo:

```
DROP LOGIN [DNS domain name\login]
```

Alterar login

Para cancelar um login, use a mesma sintaxe de qualquer login, conforme mostrado no seguinte exemplo:

```
ALTER LOGIN [DNS domain name\login] { ENABLE|DISABLE|WITH DEFAULT_DATABASE=[master] }
```

O comando ALTER LOGIN é compatível com opções limitadas para logins do Windows, inclusive as seguintes:

- DISABLE: para desabilitar um login. Você não pode usar um login desabilitado para autenticação.
- ENABLE: para habilitar um login desabilitado.
- DEFAULT_DATABASE: para alterar o banco de dados padrão de um login.

> **Note**
>
> Todo gerenciamento de senhas é executado pelo Directory Service; portanto, o comando ALTER LOGIN não permite que administradores de banco de dados alterem ou definam senhas para logins do Windows.

Conectar-se ao Babelfish para Aurora PostgreSQL com autenticação Kerberos

Normalmente, os usuários de banco de dados que se autenticam usando o Kerberos o fazem nas máquinas cliente. Essas máquinas são membros do domínio do Active Directory. Elas usam a Autenticação do Windows por meio de suas aplicações cliente para acessar o servidor do Babelfish para Aurora PostgreSQL na porta TDS.

Conectar-se ao Babelfish para Aurora PostgreSQL na porta PostgreSQL com autenticação Kerberos

É possível usar logins criados por meio da porta TDS com a porta TDS ou a porta PostgreSQL. No entanto, o PostgreSQL usa comparações com distinção entre maiúsculas e minúsculas por padrão para nomes de usuário. Para que o Aurora PostgreSQL interprete os nomes de usuário do Kerberos sem fazer distinção de maiúsculas e minúsculas, você deve definir o parâmetro `krb_caseins_users` como `true` no grupo de parâmetros de cluster personalizado do Babelfish. Esse parâmetro é definido como `false` por padrão. Para ter mais informações, consulte Configuração de nomes de usuário que não fazem distinção de maiúsculas e minúsculas. Além disso, você deve especificar o nome de usuário de login no formato <login@nome do domínio DNS> das aplicações cliente PostgreSQL. Você não pode usar o formato <nome do domínio DNS\login>.

Erros que ocorrem com frequência

Você não pode configurar uma relação de confiança de floresta entre o Microsoft Active Directory on-premises e o AWS Managed Microsoft AD. Para ter mais informações, consulte Criar uma relação de confiança. Depois, você deve se conectar usando um endpoint específico de domínio especializado em vez de usar o domínio da Amazon `rds.amazonaws.com` no endpoint de host. Se você não usar o endpoint específico do domínio correto, poderá encontrar o seguinte erro:

```
Error: “Authentication method "NTLMSSP" not supported (Microsoft SQL Server, Error:
 514)"
```

Esse erro ocorre quando o cliente TDS não consegue armazenar em cache o tíquete de serviço do URL do endpoint fornecido. Para ter mais informações, consulte Conectar-se com o Kerberos.

Configurar a autenticação do Kerberos usando grupos de segurança do Active Directory para Babelfish

A partir da versão 4.2.0 do Babelfish, é possível configurar a autenticação do Kerberos para o Babelfish com grupos de segurança do Active Directory. Veja a seguir os pré-requisitos a serem atendidos para configurar a autenticação do Kerberos usando o Active Directory:

- É necessário seguir todas as etapas mencionadas em Autenticação do Kerberos com o Babelfish.
- A instância de banco de dados deve estar associada ao Active Directory. Para verificar isso, é possível visualizar o status da associação ao domínio no console ou executando o comando describe-db-instances da AWS CLI.

  O status da instância de banco de dados deve ser habilitado para Kerberos. Para saber mais sobre a associação ao domínio, consulte Compreensão da associação de domínio.
- Verifique os mapeamentos entre o nome de domínio NetBIOS e o nome de domínio DNS usando a seguinte consulta:

  ```
  SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
  ```
- Antes de continuar, verifique se a autenticação do Kerberos usando o login individual funciona conforme o esperado. A conexão usando a autenticação do Kerberos como usuário do Active Directory deve ser bem-sucedida. Se você enfrentar algum problema, consulte Erros que ocorrem com frequência.

Configurar a extensão pg_ad_mapping

É necessário seguir todas as etapas mencionadas em Configurar a extensão pg_ad_mapping. Para verificar se a extensão está instalada, execute a seguinte consulta no endpoint do TDS:

```
1> SELECT extname, extversion FROM pg_extension where extname like 'pg_ad_mapping';
2> GO
extname       extversion
------------- ----------
pg_ad_mapping 0.1

(1 rows affected)
```

Gerenciar logins de grupo

Crie logins de grupo seguindo as etapas mencionadas em Gerenciar logins. Recomendamos que o nome de login seja igual ao nome do grupo de segurança do Active Directory (AD) para facilitar a manutenção, embora não seja obrigatório. Por exemplo:

```
CREATE LOGIN [corp\accounts-group] FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Associar logins de grupos do T-SQL ao grupo de segurança do AD

É necessário provisionar explicitamente o login do grupo do T-SQL Windows para cada grupo de segurança do AD que exija acesso ao servidor de banco de dados. Um usuário do AD, que faz parte de pelo menos um grupo de segurança do AD provisionado, terá acesso ao servidor de banco de dados.

> **Note**
>
> Esse login do T-SQL não pode mais ser autenticado usando uma autenticação baseada em senha.

Por exemplo, accounts-group é um grupo de segurança no AD. Se você quiser provisionar esse grupo de segurança no Babelfish, deverá usar o formato [corp\accounts-group].

- Grupo de segurança do AD: accounts-group
- Login do TSQL: [corp\accounts-group]

- Perfil do PG equivalente para determinado login do TSQL: accounts-group@CORP.EXAMPLE.COM

Agora o administrador pode continuar criando o mapeamento entre o grupo de segurança do AD e o login do T-SQL por meio do endpoint do PostgreSQL pelo comando psql a seguir. Para ter mais informações sobre o uso de funções, consulte Usar funções da extensão `pg_ad_mapping`.

> **Note**
>
> O login do T-SQL deve ser especificado no formato login_name@FQDN ao adicionar o mapeamento. Os pesos são ignorados quando a conexão é feita por meio do endpoint do TDS. Para ter mais informações sobre o uso de pesos, consulte Conectar-se ao Babelfish via endpoint do PostgreSQL na porta do PostgreSQL.

```
postgres=>select pgadmap_set_mapping('accounts-group', 'accounts-
group@CORP.EXAMPLE.COM', <SID>, <Weight>);
```

Para ter informações sobre como recuperar o SID do grupo de segurança do AD, consulte Recuperar o SID do grupo do Active Directory no PowerShell.

A tabela a seguir mostra um exemplo de mapeamento dos grupos de segurança do AD para logins do T-SQL:

Grupos de segurança do AD	Logins do TSQL	Perfil do PG equivalente para determinado login do TSQL	Weight
accounts-group	[corp\accounts-group]	accounts-group@CORP.EXAMPLE.COM	7
sales-group	[corp\sales-group]	sales-group@CORP.EXAMPLE.COM	10

Grupos de segurança do AD	Logins do TSQL	Perfil do PG equivalente para determinado login do TSQL	Weight
dev-group	[corp\dev-group]	dev-group @CORP.EXAMPLE.COM	7

```
postgres=> select admap.ad_sid, admap.ad_grp, lgn.orig_loginname, lgn.rolname,
 admap.weight from pgadmap_read_mapping() as admap, sys.babelfish_authid_login_ext as
 lgn where admap.pg_role = lgn.rolname;
    ad_sid    |     ad_grp     |    orig_loginname   |             rolname
 | weight
--------------+----------------+---------------------+--------------------------------
+--------
 S-1-5-67-890 | accounts-group | corp\accounts-group  | accounts-group@CORP.EXAMPLE.COM
 |     7
 S-1-2-34-560 | sales-group    | corp\sales-group     | sales-group@CORP.EXAMPLE.COM
 |     10
 S-1-8-43-612 | dev-group      | corp\dev-group       | dev-group@CORP.EXAMPLE.COM
 |     7
 (7 rows)
```

Conectar-se ao Babelfish via endpoint do TDS

No exemplo a seguir, user1 é membro de accounts-group e sales-group, ao passo que user2 é membro de accounts-group e dev-group.

Nome de usuário	Associação a grupos de segurança do AD
user1	accounts-group, sales-group
user2	accounts-group, dev-group

Conecte-se ao servidor de banco de dados do Babelfish usando o utilitário sqlcmd. É possível verificar se um usuário (neste exemplo, user1) foi autenticado usando o Kerberos seguindo o exemplo abaixo:

```
1> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();
2>  GO
principal               gss_authenticated
----------------------  -----------------
user1@CORP.EXAMPLE.COM  1

((1 rows affected))
1> select suser_name();
2>  GO
suser_name
----------
corp\user1

(1 rows affected)
```

Neste exemplo, user1 herdará os privilégios de accounts-group e sales-group. É possível verificar a associação ao grupo usando a visualização `sys.login_token` do sistema.

```
1> SELECT name, type FROM sys.login_token;
2>  GO
name                type
------------------- ----
corp\accounts-group WINDOWS GROUP
corp\sales-group    WINDOWS GROUP

(2 rows affected)
```

Auditoria e registro em log

Para determinar a identidade principal de segurança do AD, use os seguintes comandos:

```
1> select suser_name();
2> GO
suser_name
----------
```

```
corp\user1

(1 rows affected)
```

No momento, a identidade do usuário do AD não está visível nos logs. É possível ativar o parâmetro `log_connections` para registrar o estabelecimento da sessão de banco de dados. Para ter mais informações, consulte log_connections. A saída inclui a identidade do usuário do AD como entidade principal, como mostra o exemplo a seguir. O PID de back-end associado a essa saída pode então ajudar a atribuir ações de volta ao usuário real do AD.

```
bbf_group_ad_login@babelfish_db:[615]:LOG: connection authorized:
 user=bbf_group_ad_login database=babelfish_db application_name=sqlcmd GSS
 (authenticated=yes, encrypted=yes, principal=user1@CORP.EXAMPLE.COM)
```

Utilizar os privilégios da associação ao grupo de segurança do AD

Herdar privilégios em nível de servidor

Os usuários do AD que são membros de determinado grupo de segurança do AD herdarão os privilégios em nível de servidor concedidos ao login do grupo mapeado do Windows. Por exemplo, pense no grupo de segurança `accounts-group` do AD, ao qual é concedida a associação ao perfil do servidor `sysadmin` no Babelfish. É possível herdar os privilégios em nível de servidor usando o seguinte comando:

```
1> ALTER SERVER ROLE sysadmin ADD MEMBER [corp\accounts-group];
```

Consequentemente, qualquer usuário do Active Directory que seja membro do grupo de segurança `accounts-group` do AD herdará os privilégios em nível de servidor associados ao perfil `sysadmin`. Isso significa que um usuário como `corp\user1`, sendo membro do `accounts-group`, agora terá a capacidade de realizar operações em nível de servidor no Babelfish.

> **Note**
>
> Para executar DDLs em nível de servidor, deve existir um login do Windows para um usuário individual do AD. Para obter mais informações, consulte Limitações.

Herdar privilégios em nível de banco de dados

Para conceder privilégios em nível de banco de dados, um usuário do banco de dados precisa ser criado e associado ao login do grupo do Windows. Os usuários do AD que são membros de determinado grupo de segurança do AD herdarão os privilégios em nível de banco de dados concedidos a esse usuário do banco de dados. No exemplo a seguir, é possível ver como são atribuídos privilégios em nível de banco de dados para o grupo do Windows [corp\accounts-group].

```
1> CREATE DATABASE db1;
2> GO
1> USE db1;
2> GO
Changed database context to 'db1'.
1> CREATE TABLE dbo.t1(a int);
2> GO
```

Crie um usuário de banco de dados [corp\sales-group] para o login do grupo do Windows [corp \accounts-group]. Para executar essa etapa, conecte-se por meio do endpoint do TDS usando o login que é membro do sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group];
2> GO
```

Agora se conecte como user1 do AD para conferir acesso à tabela t1. Como ainda não concedemos os privilégios em nível de banco de dados, isso gerará um erro de permissão negada.

```
1> SELECT * FROM dbo.t1;
2> GO
Msg 33557097, Level 16, State 1, Server db-inst, Line 1
permission denied for table t1
```

Conceda SELECT na tabela t1 ao usuário do banco de dados [corp\accounts-group]. Para executar essa etapa, conecte-se por meio do endpoint do TDS usando o login que é membro do sysadmin.

```
1> GRANT SELECT ON dbo.t1 TO [corp\accounts-group];
2> GO
```

Conecte-se como user1 do AD para validar o acesso.

```
1> SELECT * FROM dbo.t1;
2> GO
a
-----------

(0 rows affected)
```

Tratar o comportamento da declaração DDL com base no esquema padrão ou explícito

Ao usar uma sessão autenticada pelo AD, o esquema padrão da sessão atual é determinado pelas seguintes condições:

- Se existir um usuário individual do banco de dados, o esquema padrão do usuário será considerado o esquema padrão da sessão atual.
- Se o esquema padrão para um usuário do banco de dados de grupo existir, esse esquema será considerado o esquema padrão da sessão atual com o menor ID da entidade principal.

Noções básicas sobre o comportamento da declaração CREATE DDL

Se não houver um esquema explícito especificado na declaração CREATE DDL, a criação do objeto ocorrerá no esquema padrão da sessão atual. Se não for possível determinar se o esquema é padrão ou explícito, a declaração DDL gerará o seguinte erro:

```
"Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command."
```

Example: o esquema padrão não existe para o usuário do grupo do Windows.

O usuário do grupo do Windows [corp\accounts-group] tem um esquema padrão NULL e o user1 do AD está tentando executar o DDL sem especificar o esquema explicitamente. Como o login e usuário individuais do Windows não existem para user1, ele receberá os privilégios de nível de banco de dados somente do usuário do grupo do Windows [corp\accounts-group].

```
1> create TABLE t2(a int);
2> GO

Msg 33557097, Level 16, State 1, Server db-inst, Line 1
Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command.
```

> **Note**
>
> O login e usuário individuais do Windows não existem para o user1 do AD.

Example: existe um esquema padrão para usuários do grupo do Windows.

Crie um usuário de grupo do Windows para login [corp\accounts-group] com esquema padrão usando sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group] WITH
 DEFAULT_SCHEMA = sch_acc;
2> GO
1> CREATE SCHEMA sch_acc AUTHORIZATION [gad\accounts-group];
2> GO
1> SELECT name, principal_id, default_schema_name FROM sys.database_principals WHERE
 name = 'corp\accounts-group';
2> GO

name               principal_id default_schema_name
------------------ ------------ ------------------
corp\accounts-group 24162        sch_acc
```

```
(1 rows affected)
```

Tente criar um objeto sem especificar explicitamente o esquema usando o user1 do AD. A tabela t2 será criada no esquema padrão do usuário do grupo do Windows [corp\accounts-group]. O proprietário desse objeto será o mesmo proprietário do esquema sch_acc.

```
1> CREATE TABLE t_group(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't_group';
2> GO

name    schema_name
------- -----------
t_group sch_acc

(1 rows affected)
```

> **Note**
>
> O login e usuário individuais do Windows não existem para o user1 do AD.

Example: o usuário individual do banco de dados também existe para um usuário do AD.

Se também existir um usuário de banco de dados individual para um usuário do AD, os objetos sempre serão criados no esquema associado ao usuário individual do banco de dados. Se o esquema não existir para o usuário do banco de dados, o esquema dbo será usado. Crie um login e um usuário de banco de dados individuais do Windows para o user1 do AD. Conectar-se por meio do endpoint TDS usando um login de administrador do sistema

```
1> CREATE LOGIN [corp\user1] FROM WINDOWS;
2> GO
1> CREATE USER [corp\user1] FOR LOGIN [corp\user1] WITH DEFAULT_SCHEMA = sch1;
```

```
2> GO
1> CREATE SCHEMA sch1 AUTHORIZATION [corp\user1];
2> GO
1> SELECT name, default_schema_name FROM sys.database_principals WHERE name = 'corp
\user1';
2> GO

name      default_schema_name
--------- -------------------
corp\user1 sch1

(1 rows affected)
```

Conecte-se utilizando o user1 do AD e tente criar um objeto sem especificar explicitamente o esquema. A tabela t2 será criada no esquema sch1. Observe também que o proprietário desse objeto será o mesmo do esquema sch1.

```
1> CREATE TABLE t2(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't2';
2> GO

name schema_name
---- -----------
t2   sch1

(1 rows affected)
```

Limitações

- O utilitário Dump/Restore não comporta o despejo de mapeamentos da extensão pg_ad_mapping. Você precisará recriar esses mapeamentos após a restauração.
- Não é possível usar implantação azul-verde com instâncias do Babelfish e do Aurora PostgreSQL com `pg_ad_mapping`.
- A criação implícita de esquemas não é aceita. Também não é possível usar declarações DDL que exijam a criação implícita de um esquema.

- As DDLs em nível de servidor ALTER AUTHORIZATION ON DATABASE, CREATE DATABASE, CREATE LOGIN, ALTER LOGIN, ALTER SERVER ROLE e ALTER DATABASE não são aceitas em uma sessão autenticada do Grupo AD quando existe apenas o login do Windows do grupo, mas não o login individual do Windows. Para contornar essa limitação, é recomendável realizar essas operações em uma sessão autenticada por senha ou criar um login individual do Windows.
- A criação implícita de usuários não é aceita. Comportamento ideal do T-SQL [ainda não aceito no Babelfish]: em alguns casos, como DDL e declarações de controle de acesso, como GRANT/REVOKE, em que o nome do usuário do AD é especificado no comando, mas não existe no banco de dados, o usuário do banco de dados nomeado como usuário do AD é criado implicitamente.
- Para DDLs em procedimentos ou funções PL/pgSQL que são criados por meio do endpoint PSQL e executados com o endpoint TDS na sessão autenticada do Grupo AD:
  - As declarações ALTER/DROP serão aceitas.
  - CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE FUNCTION/PROC, CREATE TYPE, CREATE SEQUENCE, CREATE TRIGGER, SELECT INTO, CREATE FULLTEXT INDEX e CREATE UNIQUE INDEX vão gerar um erro se o esquema não for fornecido explicitamente e o esquema padrão for nulo para a sessão atual.
  - CREATE DATABASE, CREATE EXTENSION e todas as outras declarações CREATE para objetos específicos de PG (não em T-SQL), como CREATE subscription, CREATE tablespace, CREATE policy e CREATE conversion, não serão aceitas.
- As DDLs do endpoint PostgreSQL não são aceitas na sessão autenticada do Grupo AD. Como solução alternativa, você sempre pode se conectar usando o usuário principal ou qualquer outro usuário usando o mecanismo de autenticação baseado em senha.
- Objetos do sistema, como SUSER_SID (), IS_SRVROLEMEMBER (), IS_MEMBER () e sys.dm_exec_sessions, têm as limitações a seguir.
  - SUSER_SID () não exibirá o SID quando o usuário do AD ou o grupo de segurança do AD forem fornecidos.
  - IS_SRVROLEMEMBER () não considerará a associação ao perfil se o usuário atual do AD estiver herdando a associação ao perfil de servidor de qualquer associação ao perfil de servidor de login do grupo do Windows.
  - IS_MEMBER() exibirá falso para qualquer consulta relacionada ao grupo do Windows.
  - sys.dm_exec_sessions não mostrará as colunas login_name e nt_user_name dos valores esperados.

Conectar-se ao Babelfish via endpoint do PostgreSQL na porta do PostgreSQL

É possível utilizar logins de grupo criados por meio da porta do TDS para se conectar por meio da porta do PostgreSQL. Para se conectar por meio da porta do PostgreSQL, é necessário especificar o nome do usuário do AD no formato `<ad_username@FQDN>` das aplicações cliente do PostgreSQL. Não é possível usar o formato `<DNS domain name\ad_username>`.

No entanto, o PostgreSQL usa comparações com distinção de maiúsculas e minúsculas por padrão para nomes de usuário. Para que o Aurora PostgreSQL interprete os nomes de usuário do Kerberos sem fazer distinção de maiúsculas e minúsculas, é necessário definir o parâmetro krb_caseins_users como verdadeiro no grupo de parâmetros de cluster personalizado do Babelfish. Esse parâmetro é definido como falso por padrão. Para obter mais informações, consulte Configurar seu cluster de banco de dados do Aurora PostgreSQL para nomes de usuário que não fazem distinção de maiúsculas e minúsculas.

Diferenças de comportamento entre endpoints do T-SQL e do PostgreSQL quando um usuário do AD faz parte de vários grupos

Suponha que o usuário user1 do AD faça parte de dois grupos de segurança do AD [corp\accounts-group] e [corp\sales-group] e o administrador do banco de dados tenha definido o mapeamento do usuário da maneira a seguir.

```
postgres=> select * from pgadmap_read_mapping();

ad_sid       | pg_role                          | weight | ad_grp
-------------+----------------------------------+--------+--------------
S-1-5-67-980 | accounts-group@CORP.EXAMPLE.COM | 7      | accounts-group
S-1-2-34-560 | sales-group@CORP.EXAMPLE.COM    | 10     | sales-group
(2 rows)
```

Se o usuário estiver se conectando por meio do endpoint do T-SQL, durante a autorização ele herdará os privilégios de todos os logins do T-SQL associados. Neste exemplo, user1 herdará a união de privilégios dos logins do grupo do T-SQL e os pesos serão ignorados. Isso está alinhado com o comportamento padrão do T-SQL.

No entanto, se o mesmo usuário se conectar por meio do endpoint do PostgreSQL, ele poderá herdar privilégios apenas de um login do T-SQL correspondente com o maior peso. Se os logins de

dois grupos do T-SQL tiverem o mesmo peso, o usuário do AD herdará os privilégios do login do T-SQL correspondentes ao mapeamento que foi adicionado mais recentemente. Com relação ao PostgreSQL, a recomendação é especificar pesos que reflitam as permissões e privilégios relativos dos perfis individuais do banco de dados para evitar ambiguidade. No exemplo abaixo, user1 se conectou por meio do endpoint do PSQL e herdou somente privilégios de grupos de vendas.

```
babelfish_db=> select session_user, current_user;

   session_user               |   current_user
------------------------------+---------------------------
 sales-group@CORP.EXAMPLE.COM | sales-group@CORP.EXAMPLE.COM
(1 row)


babelfish_db=> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();

     principal          | gss_authenticated
------------------------+-------------------
 user1@CORP.EXAMPLE.COM | t
(1 row)
```

Conectar-se a um cluster de banco de dados do Babelfish

Para conectar-se ao Babelfish, conecte-se ao endpoint do cluster do Aurora PostgreSQL que executa o Babelfish. Seu cliente pode utilizar um dos seguintes drivers de cliente em conformidade com o TDS versão 7.1 a 7.4:

- Open Database Connectivity (ODBC)
- OLE DB Driver/MSOLEDBSQL
- Conectividade de banco de dados Java (JDBC) versão 8.2.2 (mssql-jdbc-8.2.2) e posteriores
- Microsoft SqlClient Data Provider for SQL Server
- .NET Data Provider for SQL Server
- SQL Server Native Client 11.0 (obsoleto)
- OLE DB Provider/SQLOLEDB (obsoleto)

Com o Babelfish, você executa o seguinte:

- Ferramentas, aplicações e sintaxe do SQL Server na porta do TDS que, por padrão, é a porta 1433.
- Ferramentas, aplicações e sintaxe do PostgreSQL na porta do PostgreSQL que, por padrão, é a porta 5432.

Para saber mais sobre como se conectar ao Aurora PostgreSQL em geral, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL.

> **Note**
>
> Não há suporte para ferramentas de desenvolvedores de terceiros que usam o provedor OLEDB do SQL Server para acessar metadados. Recomendamos utilizar conexões de cliente SQL Server JDBC, ODBC ou SQL Native para essas ferramentas.

A partir da versão 5.1.0 do Babelfish, a criptografia de conexão de ponta a ponta é imposta por padrão. Para garantir a conectividade contínua:

- Configure a criptografia SSL/TLS para suas conexões. Para obter mais informações, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

- Importe os certificados necessários em seus computadores cliente. Para obter mais informações, consulte Uso do SSL com uma instância de banco de dados do Microsoft SQL Server.

Se quiser continuar usando as configurações de criptografia de uma versão precedente do Babelfish (anterior à versão 5.1.0), você pode definir o parâmetro `rds.force_ssl` como `0` no grupo de parâmetros do cluster de banco de dados.

Tópicos

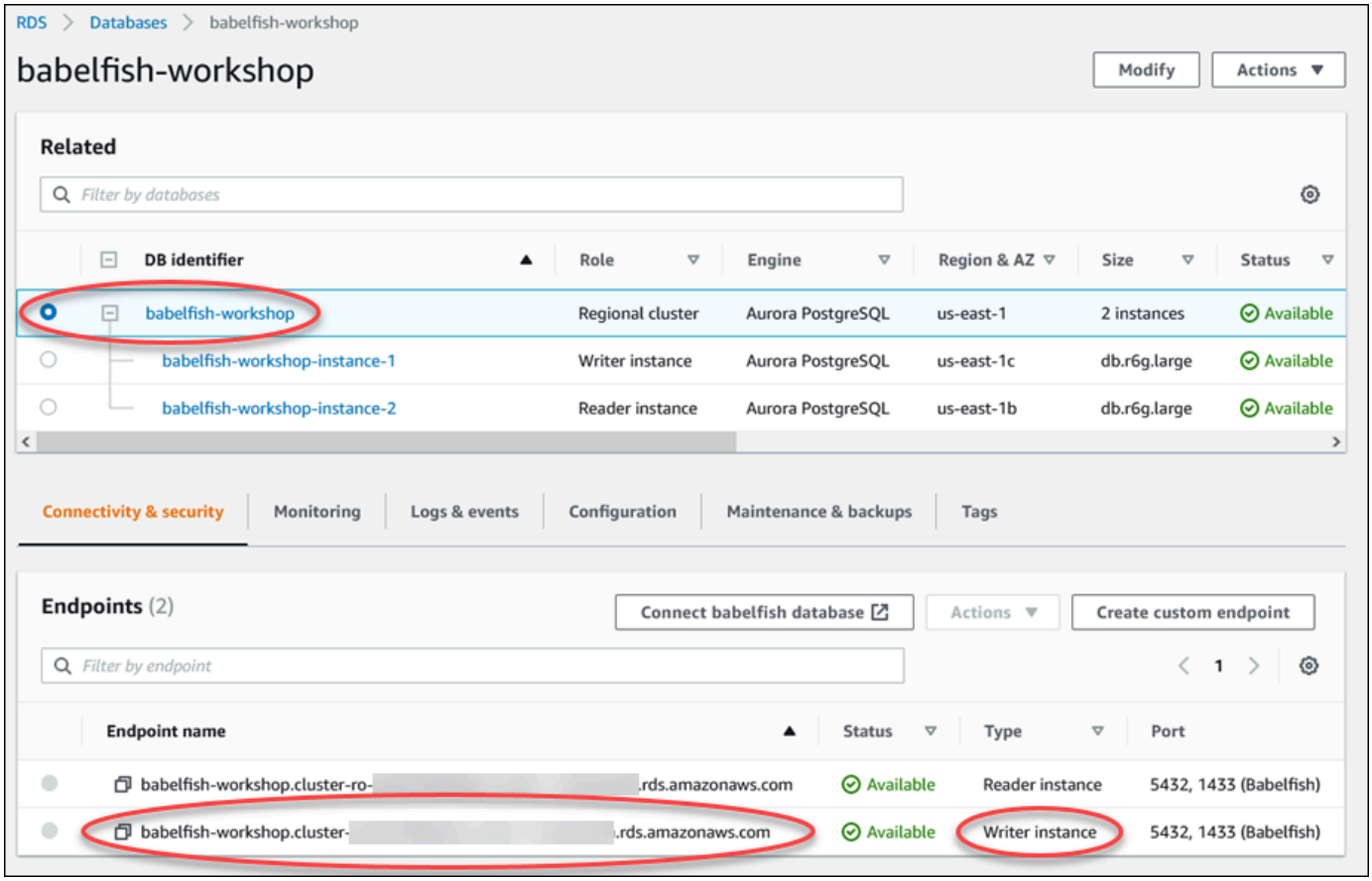
- Localizar o endpoint e o número da porta do gravador
- Criar conexões de cliente C# ou JDBC com o Babelfish
- Utilizar um cliente SQL Server para se conectar ao seu cluster de banco de dados
- Utilizar um cliente PostgreSQL para se conectar ao seu cluster de banco de dados

Localizar o endpoint e o número da porta do gravador

Para conectar-se ao cluster de banco de dados do Babelfish, use o endpoint associado à instância do gravador (primária) do cluster de banco de dados. A instância deve ter o status de Available (Disponível). Pode levar até 20 minutos para que as instâncias estejam disponíveis após a criação do cluster de banco de dados do Babelfish para Aurora PostgreSQL.

Para localizar o endpoint do banco de dados

1. Abra o console do Babelfish.
2. Escolha Databases (Bancos de dados) no painel de navegação.
3. Escolha o cluster de banco de dados do Babelfish para Aurora PostgreSQL dentre os listados para ver os detalhes.
4. Na guia Connectivity & security (Conectividade e segurança), observe os valores de Endpoints od cluster disponível. Use o endpoint de cluster para a instância de gravador nas suas strings de conexão para quaisquer aplicações que realizam operações de gravação ou leitura de banco de dados.



Para obter mais informações sobre detalhes do cluster de banco de dados do Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Criar conexões de cliente C# ou JDBC com o Babelfish

Depois, você pode encontrar alguns exemplos de uso de classes C# e JDBC para conectar-se a um Babelfish para Aurora PostgreSQL.

Example: utilizar código C# para se conectar a um cluster de banco de dados

```
string dataSource = 'babelfishServer_11_24';

//Create connection
connectionString = @"Data Source=" + dataSource
    +";Initial Catalog=your-DB-name"
    +";User ID=user-id;Password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionString = connectionString + ";ServerCertificate=/path/to/certificate.pem";


SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
```

Example: utilizar classes e interfaces genéricas da API JDBC para se conectar a um cluster de banco de dados

```
String dbServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";
String connectionUrl = "jdbc:sqlserver://" + dbServer + ":1433;" +
    "databaseName=your-DB-name;user=user-id;password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionUrl = connectionUrl + ";serverCertificate=/path/to/certificate.pem";

// Load the SQL Server JDBC driver and establish the connection.
System.out.print("Connecting Babelfish Server ... ");
Connection cnn = DriverManager.getConnection(connectionUrl);
```

Example: utilizar classes e interfaces JDBC específicas do SQL Server para se conectar a um cluster de banco de dados

```
// Create datasource.
SQLServerDataSource ds = new SQLServerDataSource();
ds.setUser("user-id");
```

```
ds.setPassword("password");
String babelfishServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";

ds.setServerName(babelfishServer);
ds.setPortNumber(1433);
ds.setDatabaseName("your-DB-name");

// [optional] To validate server certificate during TLS/SSL connection
ds.setServerCertificate("/path/to/certificate.pem");

Connection con = ds.getConnection();
```

Important

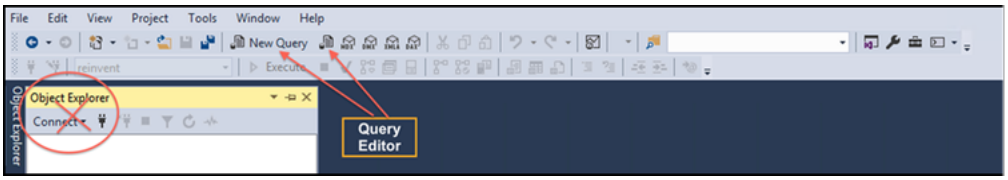
O certificado deve corresponder à autoridade certificadora mostrada na configuração do seu cluster de banco de dados no Console de gerenciamento da AWS.

Utilizar um cliente SQL Server para se conectar ao seu cluster de banco de dados

Você pode utilizar um cliente SQL Server para se conectar ao Babelfish na porta do TDS. A partir das versões Babelfish 2.1.0 e posteriores, você pode usar o SSMS Object Explorer ou o Editor de consultas SSMS para se conectar ao cluster do Babelfish.

Limitações

- No Babelfish 2.1.0 e em versões anteriores, o uso de `PARSE` para conferir a sintaxe SQL não funciona como deveria. Em vez de verificar a sintaxe sem executar a consulta, o comando `PARSE` executa a consulta, mas não exibe nenhum resultado. O uso da combinação de chaves <Ctrl><F5> do SMSS para conferir a sintaxe tem o mesmo comportamento anômalo, ou seja, o Babelfish executa inesperadamente a consulta sem fornecer nenhum resultado.
- O Babelfish não é compatível com o MARS (vários conjuntos de resultados ativos). Certifique-se de que qualquer aplicação cliente que você usa para se conectar ao Babelfish não esteja configurada para usar o MARS.
- Para Babelfish 1.3.0 e versões anteriores, somente o Editor de consultas é compatível com o SSMS. Para usar o SSMS com o Babelfish, abra a caixa de diálogo de conexão do Editor de consulta no SSMS e não o Object Explorer. Se a caixa de diálogo do Object Explorer for aberta, cancele-a e reabra o Editor de consultas. Na imagem a seguir, você pode encontrar as opções de menu para escolha ao conectar-se ao Babelfish 1.3.0 ou versões anteriores.



Para obter mais informações sobre interoperabilidade e diferenças comportamentais entre o SQL Server e o Babelfish, consulte Diferenças entre o Babelfish para Aurora PostgreSQL e o SQL Server.

Utilizar o sqlcmd para se conectar ao cluster de banco de dados

É possível se conectar e interagir com um cluster de bancos de dados do Aurora PostgreSQL que seja compatível com o Babelfish somente utilizando a versão 19.1 e o cliente de linha de comandos `sqlcmd` do SQL Server anterior. A versão 19.2 do SSMS não é aceita para conexão com um cluster do Babelfish. Use o seguinte comando para se conectar.

```
sqlcmd -S endpoint,port -U login-id -P password -d your-DB-name
```

As opções são as seguintes:

- `-S` é o endpoint e a porta do TDS (opcional) do cluster de banco de dados.
- `-U` é o nome de login do usuário.
- `-P` é a senha associada ao usuário.
- `-d` é o nome do seu banco de dados Babelfish.

Após a conexão, será possível utilizar muitos dos mesmos comandos que você utiliza com o SQL Server. Para obter alguns exemplos, consulte Obter informações do catálogo de sistemas do Babelfish.

Utilizar o SSMS para se conectar ao cluster de banco de dados

Você pode se conectar a um cluster de banco de dados do Aurora PostgreSQL que executa o Babelfish usando o Microsoft SQL Server Management Studio (SSMS). O SSMS inclui uma variedade de ferramentas, por exemplo, o assistente de importação e exportação do SQL Server abordado em Migrar um banco de dados do SQL Server para o Babelfish para Aurora PostgreSQL. Para obter mais informações sobre o SSMS, consulte Download SQL Server Management Studio (SSMS) (Baixar o SQL Server Management Studio (SSMS)) na documentação da Microsoft. Para configurar o SSL/TLS, consulte Uso do SSL com uma instância de banco de dados do Microsoft SQL Server.

Para se conectar ao seu banco de dados do Babelfish com o SSMS

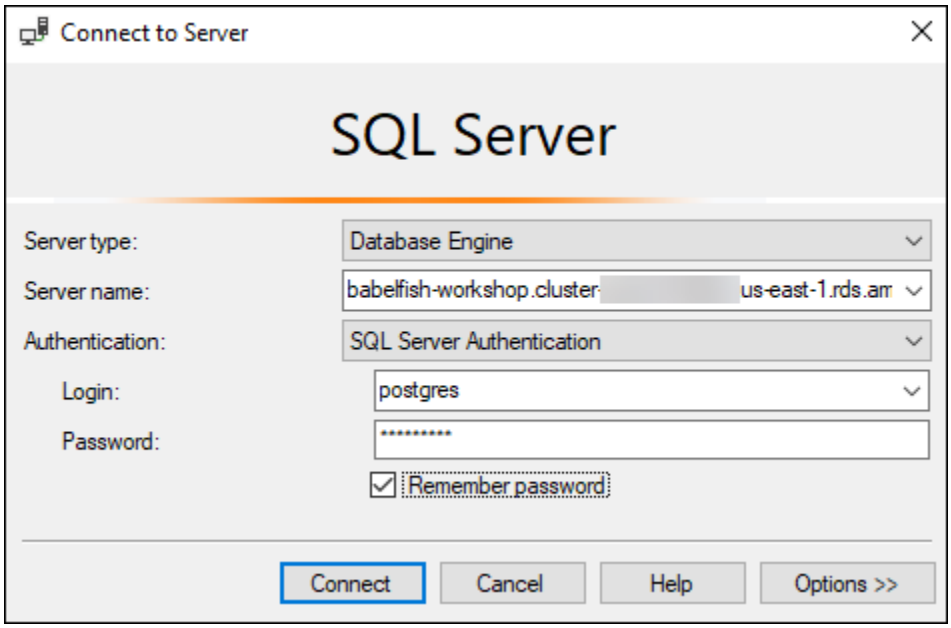
1. Inicie o SSMS.
2. Abra a caixa de diálogo Connect to Server (Conectar-se ao servidor). Para continuar com a conexão, siga um destes procedimentos:
   - Escolha New Query (Nova consulta).
   - Se o Query Editor estiver aberto, escolha Query (Consulta), Connection (Conexão), Connect (Conectar).
3. Forneça as seguintes informações para seu banco de dados:
   a. Para Server type (Tipo de servidor), escolha Database Engine (Mecanismo do banco de dados).
   b. Para Server name (Nome do servidor), insira o nome DNS. Por exemplo, o nome do servidor deve ser semelhante ao seguinte.

```
cluster-name.cluster-555555555555.aws-region.rds.amazonaws.com,1433
```

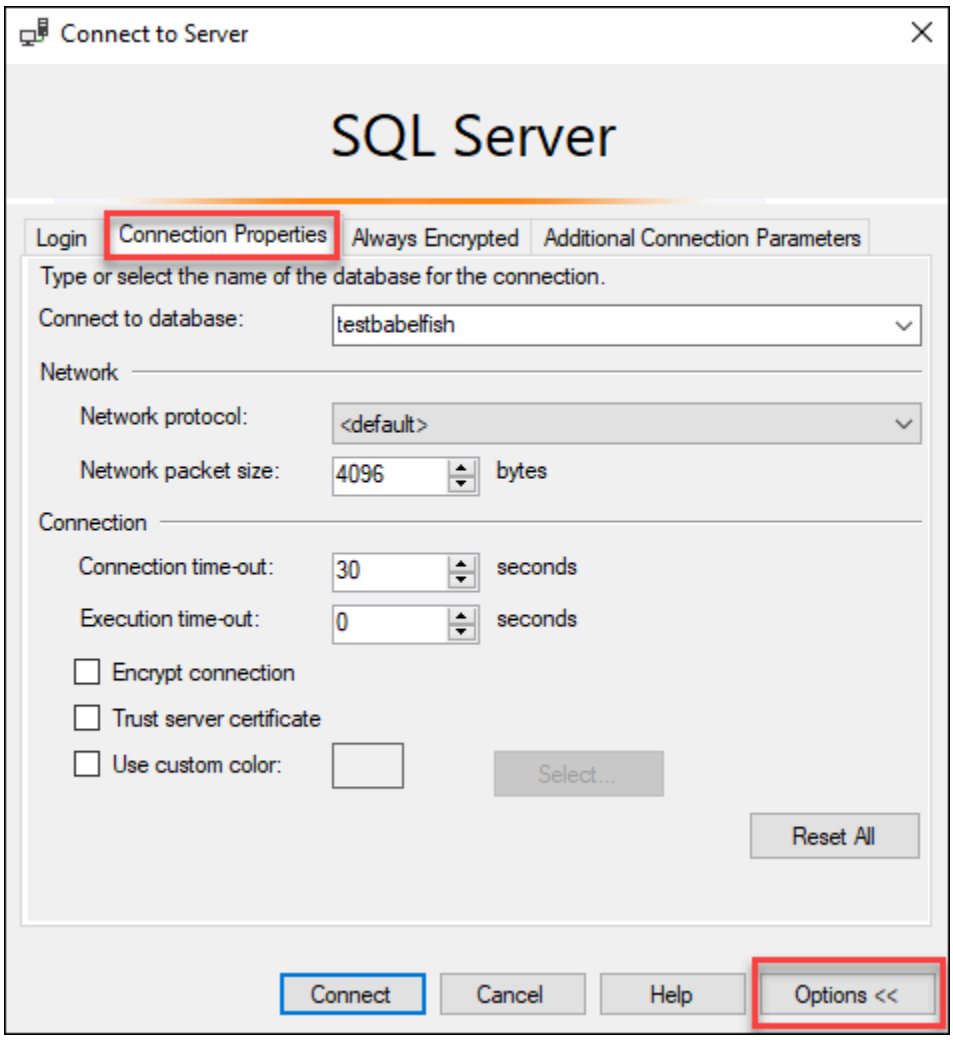
c. Para Authentication (Autenticação), escolha SQL Server Authentication (Autenticação do SQL Server).

d. Para Login, insira o nome de usuário escolhido quando você criou o banco de dados.

e. Para Password (Senha), digite a senha escolhida quando você criou o banco de dados.



4. (Opcional) Escolha Options (Opções) e depois a guia Connection Properties (Propriedades da conexão).



5. (Opcional) Para Connect to database (Conectar-se ao banco de dados), especifique o nome do banco de dados do SQL Server migrado ao qual você deseja se conectar e depois escolha Connect (Conectar).

   Se aparecer uma mensagem indicando que o SSMS não pode aplicar strings de conexão, escolha OK.

   Se você estiver tendo problemas para se conectar ao Babelfish, consulte Falha de conexão.

   Para obter mais informações sobre problemas de conexão do SQL Server, consulte Resolução de problemas para se conectar com a instância de banco de dados do SQL Server no Guia do usuário do Amazon RDS.

Utilizar um cliente PostgreSQL para se conectar ao seu cluster de banco de dados

Você pode utilizar um cliente PostgreSQL para se conectar ao Babelfish na porta do PostgreSQL. A partir da versão 5.1.0, o servidor do Babelfish impõe a criptografia de conexão de ponta a ponta por padrão. Atualize sua aplicação para que ela funcione com o novo certificado SSL/TLS. Para ter mais informações sobre certificados SSL/TLS, consulte the section called “Como proteger dados do Aurora PostgreSQL com SSL/TLS”.

Utilizar o psql para se conectar ao cluster de banco de dados

Você pode baixar o cliente PostgreSQL no site do PostgreSQL. Siga as instruções específicas da versão de seu sistema operacional para instalar o psql.

Você pode consultar um cluster de bancos de dados Aurora PostgreSQL compatível com o Babelfish com o cliente de linha de comando `psql`. Ao conectar-se, utilize a porta do PostgreSQL (por padrão, a porta 5432). Normalmente, você não precisa especificar o número da porta, a menos que o tenha alterado do padrão. Utilize o comando a seguir para se conectar ao Babelfish a partir do cliente `psql`:

```
psql -h bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com
-p 5432 -U postgres -d babelfish_db
```

Os parâmetros são os seguintes:

- `-h`: o nome do host do cluster de banco de dados (endpoint do cluster) que você deseja acessar.
- `-p`: o número da porta do PostgreSQL utilizado para conexão com sua instância de banco de dados.
- `-d`: o banco de dados que você deseja acessar. O padrão é `babelfish_db`.
- `-U`: a conta de usuário de banco de dados que você deseja acessar. (O exemplo mostra o nome de usuário principal padrão.)

Ao executar um comando SQL no cliente psql, você encerra o comando com um ponto-e-vírgula. Por exemplo, o comando SQL a seguir consulta a visualização do sistema pg_tables para retornar informações sobre cada tabela do banco de dados.

```
SELECT * FROM pg_tables;
```

O cliente psql também tem um conjunto de metacomandos integrados. Um metacomando é um atalho que ajusta a formatação ou fornece um atalho que retorna metadados em um formato fácil de

utilizar. Por exemplo, o metacomando a seguir retorna informações semelhantes ao comando SQL anterior:

```
\d
```

Metacomandos não precisam ser encerrados com um ponto e vírgula (;).

Para sair do cliente psql, insira \q.

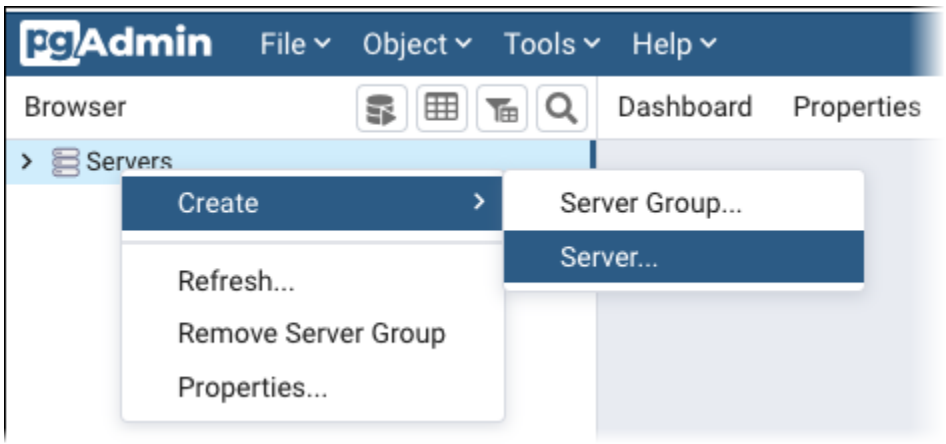
Para ter mais informações sobre como utilizar o cliente psql para consultar um cluster do Aurora PostgreSQL, consulte a documentação do PostgreSQL.

Utilizar o pgAdmin para se conectar ao cluster de banco de dados

Você pode utilizar o cliente pgAdmin para acessar seus dados no dialeto do PostgreSQL nativo.

Para se conectar ao cluster com o cliente pgAdmin

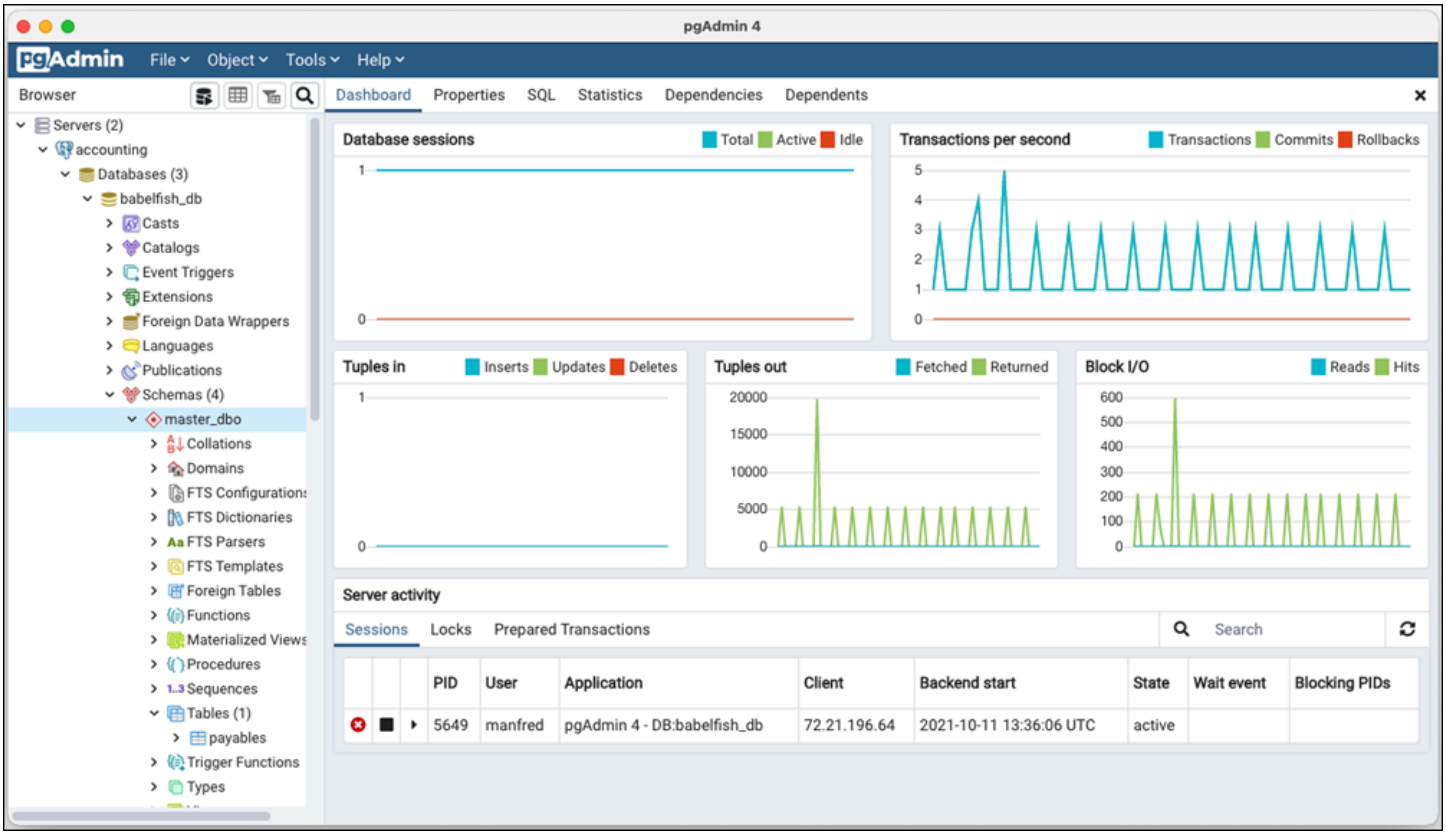
1. Baixe e instale o cliente pgAdmin acessando o site do pgAdmin.
2. Abra o cliente e autentique-se no pgAdmin.
3. Abra o menu contexto (clique com o botão direito do mouse) para Servers (Servidores) e depois escolha Create (Criar), Server (Servidor).



4. Insira informações na caixa de diálogo Create - Server (Criar - Servidor).

   Na guia Connection (Conexão), adicione o endereço do cluster do Aurora PostgreSQL para Host e o número da porta do PostgreSQL (por padrão, 5432) para Port (Porta). Forneça detalhes de autenticação e escolha Save (Salvar).

Após a conexão, será possível utilizar a funcionalidade do pgAdmin para monitorar e gerenciar o cluster do Aurora PostgreSQL na porta do PostgreSQL.



Para saber mais, consulte a página da web pgAdmin.

Trabalhar com o Babelfish

A seguir, você pode encontrar informações de uso do Babelfish, por exemplo, algumas das diferenças entre trabalhar com o Babelfish e o SQL Server, e entre bancos de dados do Babelfish e do PostgreSQL.

Tópicos

- Obter informações do catálogo de sistemas do Babelfish
- Gerenciar permissões e controle de acesso no Babelfish para Aurora PostgreSQL
- Diferenças entre o Babelfish para Aurora PostgreSQL e o SQL Server
- Usar recursos do Babelfish com implementação limitada
- Melhorar a performance de consultas do Babelfish
- Usar extensões do Aurora PostgreSQL com o Babelfish
- O Babelfish é compatível com servidores vinculados
- Usar pesquisa de texto completo no Babelfish
- O Babelfish comporta tipos de dados geoespaciais
- Noções básicas do particionamento no Babelfish

Obter informações do catálogo de sistemas do Babelfish

Para obter informações sobre os objetos de banco de dados armazenados no cluster do Babelfish, consulte muitas das mesmas visualizações do sistema utilizadas no SQL Server. Cada nova versão do Babelfish adiciona compatibilidade com mais visualizações do sistema. Para obter uma lista das visualizações atualmente disponíveis, consulte a tabela SQL Server system catalog views.

Essas visualizações do sistema fornecem informações do catálogo de sistemas (`sys.schemas`). No caso do Babelfish, essas visualizações contêm esquemas de sistema SQL Server e PostgreSQL. Para consultar o Babelfish para obter informações de catálogo de sistemas, você pode usar a porta do TDS ou a porta do PostgreSQL, conforme mostrado nos exemplos a seguir.

- Consulte a porta do T-SQL usando **`sqlcmd`** ou outro cliente do SQL Server.

```
1> SELECT * FROM sys.schemas
2> GO
```

Essa consulta retorna esquemas de sistema do SQL Server e do Aurora PostgreSQL, conforme mostrado a seguir.

```
name
------------------------------------------------------------
demographic_dbo
public
sys
master_dbo
tempdb_dbo
...
```

- Consulte a porta do PostgreSQL usando **`psql`** ou **`pgAdmin`**. Este exemplo usa o metacomando de esquemas da lista `psql` (`\dn`):

```
babelfish_db=> \dn
```

A consulta retorna o mesmo conjunto de resultados que o retornado por `sqlcmd` na porta do T-SQL.

```
        List of schemas
             Name
-----------------------------

 demographic_dbo

 public
 sys
 master_dbo
 tempdb_dbo
...
```

Catálogos de sistemas do SQL Server disponíveis no Babelfish

Na tabela a seguir, você pode encontrar as visualizações do SQL Server implementadas atualmente no Babelfish. Para ter mais informações sobre os catálogos de sistemas no SQL Server, consulte System Catalog Views (Transact-SQL) (Visualizações do catálogo de sistemas (Transact-SQL) na documentação da Microsoft.

Visualizar nome	Descrição ou limitação do Babelfish (se houver)
`sys.all_columns`	Todas as colunas em todas as tabelas e visualizações
`sys.all_objects`	Todos os objetos em todos os esquemas
`sys.all_sql_modules`	A união de `sys.sql_modules` e `sys.system_sql_modules`
`sys.all_views`	Todas as visualizações em todos os esquemas
`sys.columns`	Todas as colunas em tabelas e visualizações definidas pelo usuário
`sys.configurations`	Compatibilidade do Babelfish limitada a uma única configuração somente leitura.
`sys.data_spaces`	Contém uma linha para cada espaço de dados. Isso pode ser um grupo de arquivos, um esquema de partição ou um grupo de arquivos de dados FILESTREAM.
`sys.database_files`	Uma visualização por banco de dados que contém uma linha para cada arquivo de um banco de dados, conforme armazenado no próprio banco de dados.
`sys.database_mirroring`	Para ter mais informações, consulte sys.database_mirroring na documentação do Microsoft Transact-SQL.
`sys.database_principals`	Para ter mais informações, consulte sys.database_principals na documentação do Microsoft Transact-SQL.
`sys.database_role_members`	Para ter mais informações, consulte sys.database_role_members na documentação do Microsoft Transact-SQL.

Visualizar nome	Descrição ou limitação do Babelfish (se houver)
`sys.databases`	Todos os bancos de dados em todos os esquemas
`sys.dm_exec_connections`	Para ter mais informações, consulte sys.dm_exec_connections na documentação do Microsoft Transact-SQL.
`sys.dm_exec_sessions`	Para ter mais informações, consulte sys.dm_exec_sessions na documentação do Microsoft Transact-SQL.
`sys.dm_hadr_database_replica_states`	Para ter mais informações, consulte sys.dm_hadr_database_replica_states na documentação do Microsoft Transact-SQL.
`sys.dm_os_host_info`	Para ter mais informações, consulte sys.dm_os_host_info na documentação do Microsoft Transact-SQL.
`sys.endpoints`	Para ter mais informações, consulte sys.endpoints na documentação do Microsoft Transact-SQL.
`sys.indexes`	Para ter mais informações, consulte sys.indexes na documentação do Microsoft Transact-SQL.
`sys.languages`	Para ter mais informações, consulte sys.languages na documentação do Microsoft Transact-SQL.
`sys.schemas`	Todos os esquemas
`sys.server_principals`	Todos os logins e funções

Visualizar nome	Descrição ou limitação do Babelfish (se houver)
`sys.sql_modules`	Para ter mais informações, consulte sys.sql_modules na documentação do Microsoft Transact-SQL.
`sys.sysconfigures`	Compatibilidade do Babelfish limitada a uma única configuração somente leitura.
`sys.syscurconfigs`	Compatibilidade do Babelfish limitada a uma única configuração somente leitura.
`sys.sysprocesses`	Para ter mais informações, consulte sys.sysprocesses na documentação do Microsoft Transact-SQL.
`sys.system_sql_modules`	Para ter mais informações, consulte sys.system_sql_modules na documentação do Microsoft Transact-SQL.
`sys.table_types`	Para ter mais informações, consulte sys.table_types na documentação do Microsoft Transact-SQL.
`sys.tables`	Todas as tabelas em um esquema
`sys.xml_schema_collections`	Para ter mais informações, consulte sys.xml_schema_collections na documentação do Microsoft Transact-SQL.

O PostgreSQL implementa catálogos do sistema que são semelhantes às visualizações do catálogo de objetos do SQL Server. Para obter uma lista completa de catálogos do sistema, consulte o tópico sobre Catálogos do sistema, na documentação do PostgreSQL.

Exportações de DDL compatíveis com o Babelfish

Nas versões 2.4.0 e 3.1.0 do Babelfish, ele é compatível com exportações de DDL de várias ferramentas. Por exemplo, você pode usar essa funcionalidade do SQL Server Management Studio (SSMS) para gerar os scripts de definição de dados para vários objetos em um banco de dados

Babelfish para Aurora PostgreSQL. Você pode então usar os comandos DDL gerados nesse script para criar os mesmos objetos em outro banco de dados Babelfish para Aurora PostgreSQL ou SQL Server.

O Babelfish é compatível com exportações de DDL para os objetos a seguir nas versões especificadas.

Lista de objetos	2.4.0	3.1.0
Tabelas de usuários	Sim	Sim
Chaves primárias	Sim	Sim
Chaves externas	Sim	Sim
Restrições exclusivas	Sim	Sim
Índices	Sim	Sim
Restrições de verificação	Sim	Sim
Visualizações	Sim	Sim
Procedimentos armazenados	Sim	Sim
Funções definidas pelo usuário	Sim	Sim
Funções com valor de tabela	Sim	Sim
Acionadores	Sim	Sim
Tipos de dados definidos pelo usuário	Não	Não
Tipos de tabela definidos pelo usuário	Não	Não
Usuários	Não	Não
Logins	Não	Não
Sequências	Não	Não
Perfis	Não	Não

Limitações com os DDLs exportados

- Use hatches de escape antes de recriar os objetos com os DDLs exportados: o Babelfish não é compatível com todos os comandos no script DDL exportado. Use hatches de escape para evitar erros causados ao recriar os objetos dos comandos DDL no Babelfish. Para ter mais informações sobre hatches de escape, consulte Gerenciar o tratamento de erros do Babelfish com hatches de escape
- Objetos contendo restrições CHECK com cláusulas COLLATE explícitas: os scripts com esses objetos gerados por meio de um banco de dados SQL Server têm agrupamentos diferentes, mas equivalentes, como no banco de dados Babelfish. Por exemplo, alguns agrupamentos, como sql_latin1_general_cp1_cs_as, sql_latin1_general_cp1251_cs_as e latin1_general_cs_as são gerados como latin1_general_cs_as, que é o agrupamento mais próximo do Windows.

Gerenciar permissões e controle de acesso no Babelfish para Aurora PostgreSQL

No Babelfish para Aurora PostgreSQL, é possível gerenciar permissões e controle de acesso para bancos de dados, esquemas e objetos. As tabelas a seguir descrevem os comandos SQL específicos para conceder permissões no Babelfish para atingir vários cenários de controle de acesso. Serão abordados casos de uso compatíveis que podem ser implementados, bem como soluções alternativas para casos incompatíveis no momento. Isso permitirá que você configure as permissões apropriadas para atender aos seus requisitos de segurança e conformidade ao trabalhar com bancos de dados do Babelfish.

Casos de uso compatíveis

A tabela a seguir explica os casos de uso que são compatíveis com o Babelfish. Para cada caso de uso, a tabela mostra a ação necessária para atingi-lo e exemplos de comandos SQL.

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
Permitir que o login faça SELECTs/DMLs/DDLs em	Adicionar o login ao perfil do servidor sysadmin	ALTER SERVER ROLE sysadmin ADD MEMBER `login`	Nenhum	Todas as versões

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
qualquer banco de dados				
Permitir que o login faça SELECTs/DMLs/DDLs em um banco de dados	Tornar o login o proprietário do banco de dados	ALTER AUTHORIZATION ON DATABASE::`database` TO `login`	Um banco de dados pode ter apenas um proprietário.	Versão 3.4 e posterior
Permitir que o usuário do banco de dados faça SELECTs/DMLs em um esquema	Conceder permissão ao usuário do banco de dados no esquema	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON SCHEMA::`schema` TO `user`	Nenhum	Versão 3.6 e posterior, 4.2 e posterior
Permitir que o usuário do banco de dados faça SELECTs/DMLs em um esquema	Tornar o usuário do banco de dados o proprietário do esquema no momento da criação do esquema	CREATE SCHEMA `schema` AUTHORIZATION `user`	No momento, não é possível alterar a propriedade do esquema após a criação.	Versão 1.2 e posterior
Permitir que o usuário do banco de dados faça SELECTs/DMLs em um objeto	Conceder permissão ao usuário do banco de dados no objeto	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT::`object` TO `user`	Nenhum	Todas as versões

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
Permitir que o usuário do banco de dados faça SELECTs/DMLs/DDLs em um banco de dados, incluindo o descarte do banco de dados	Adicionar usuário ao perfil fixo de banco de dados db_owner	ALTER ROLE db_owner ADD MEMBER `user`	Somente usuários podem ser adicionados ao perfil fixo de banco de dados db_owner. Ainda não é possível adicionar perfis ao perfil db_owner.	Versão 4.5 e posterior, 5.1 e posterior
Permitir que o usuário ou membros de um perfil personalizado de banco de dados façam somente SELEÇÕES em um banco de dados	Adicionar usuário ou perfil ao perfil fixo de banco de dados db_datareader	ALTER ROLE db_datareader ADD MEMBER `user` / `role`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior
Permitir que o usuário ou membros de um perfil personalizado de banco de dados façam somente DLMs em um banco de dados	Adicionar usuário ou perfil ao perfil fixo de banco de dados db_datawriter	ALTER ROLE db_datawriter ADD MEMBER `user` / `role`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
Permitir que o usuário ou membros de um perfil personalizado de banco de dados façam somente DDLs em um banco de dados	Adicionar usuário ou perfil ao perfil fixo de banco de dados db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior
Permitir que o usuário ou membros de um perfil de banco de dados personalizado apenas CRIEM/ ALTEREM/ EXCLUAM perfis personalizados, CONCEDAM/ REVOGUEM permissões em objetos em um banco de dados e/ou CRIEM ESQUEMA em um banco de dados	Adicionar usuário ou perfil ao perfil fixo de banco de dados db_securityadmin	ALTER ROLE db_securityadmin ADD MEMBER `user` / `role`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
Permita que o usuário ou membros de um perfil de banco de dados personalizado apenas CRIEM/ ALTEREM/EXCLUAM qualquer usuário, concedam e revoguem o acesso ao banco de dados e associem contas de usuário a logins e/ou CRIEM ESQUEMA em um banco de dados	Adicionar usuário ou perfil ao perfil fixo de banco de dados db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior
Permitir login somente para CRIAR/DESCARTAR/ALTERAR qualquer banco de dados	Adicionar login ao perfil fixo de servidor dbcreator	ALTER SERVER ROLE dbcreator ADD MEMBER `login`	Somente os bancos de dados aos quais o login dbcreator tem acesso podem ser alterados.	Versão 4.5 e posterior, 5.1 e posterior

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish
Permitir login somente para CRIAR/DESCARTAR/ALTERAR qualquer login	Adicionar login ao perfil fixo de servidor securityadmin	ALTER SERVER ROLE securityadmin ADD MEMBER `login`	Nenhum	Versão 4.5 e posterior, 5.1 e posterior

Casos de uso incompatíveis e soluções alternativas

A tabela a seguir explica os casos de uso que são incompatíveis com o Babelfish, mas que podem ser conseguidos usando uma solução alternativa.

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish para soluções alternativas
Permitir que o usuário faça SELEÇÕES/DMLs em um objeto/esquema com a opção de CONCEDER essas permissões a outros usuários	CONCEDER permissões diretamente a todos os outros usuários	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT/SCHEMA::`object/sc` TO `user`	GRANT ... WITH GRANT OPTION não é permitido no momento.	Versão 3.6 e posterior, 4.2 e posterior
Permitir que o usuário do banco de dados faça SELECTs/D	Adicionar membros do perfil ao perfil	ALTER ROLE db_owner ADD MEMBER `user`	A adição de perfis ao perfil db_owner não	Versão 4.5 e posterior, 5.1 e posterior

Caso de uso	Ação	Comandos SQL	Comentários	Compatibilidade com a versão do Babelfish para soluções alternativas
MLs/DDLs em um banco de dados, incluindo o descarte do banco de dados	fixo de banco de dados db_owner		é permitida no momento.	

Casos de uso sem suporte

A tabela a seguir explica os casos de uso compatíveis com o Babelfish.

Caso de uso	Comentários
Negar que usuários ou membros de um perfil personalizado de banco de dados façam SELEÇÕES em um banco de dados	O perfil fixo de banco de dados db_denydatareader ainda não é permitido.
Negar que usuários ou membros de um perfil personalizado de banco de dados façam DMLs em um banco de dados	O perfil fixo de banco de dados db_denydatawriter não é permitido no momento.
Permitir login somente para ENCERRAMENTO de qualquer conexão de banco de dados	O perfil fixo de servidor processadmin não é permitido no momento.
Permitir login somente para adicionar ou remover servidores vinculados	O perfil fixo de servidor setupadmin não é permitido no momento.

Como lidar com as diferenças de propriedade de objeto após a atualização

As versões 4.6 e posteriores e 5.2 e posteriores do Babelfish incluem uma alteração no tratamento de propriedade de objeto por meio do endpoint TDS. Agora, ao criar objetos por meio do endpoint TDS, eles pertencem ao proprietário do esquema em vez do usuário atual. Essa alteração de

propriedade pode afetar o comportamento das permissões de novos objetos em comparação aos objetos existentes ao atualizar de versões anteriores à 4.6 ou 5.2.

Para resolver essas diferenças de propriedade, o Babelfish oferece a função `sys.generate_alter_ownership_statements()`. Essa função gera instruções SQL que alinham a propriedade do objeto à propriedade do esquema.

Lembre-se das seguintes limitações ao lidar com a propriedade de objeto:

- Os usuários com permissões CREATE concedidas por meio do endpoint do PostgreSQL não podem criar objetos por meio do endpoint TDS nesses esquemas.
- A modificação de permissões em objetos T-SQL por meio do endpoint do PostgreSQL não é recomendada e pode fazer com que o T-SQL tenha um comportamento incorreto.
- As permissões de acesso podem diferir entre objetos antigos e novos devido à incompatibilidade entre propriedades. Por exemplo, considere um esquema pertencente a `sch_own` que inclua objetos pertencentes a `dbo`. Nesse caso, objetos de propriedade de `dbo` que foram criados antes da atualização podem ter permissões de acesso diferentes em comparação aos objetos de propriedade de `sch_own` que foram criados após a atualização. Isso pode afetar operações como SELECT e INSERT.

Se o cluster de banco de dados incluir objetos criados nas versões do Babelfish anteriores à 4.6 ou 5.2, considere a possibilidade de alinhar a propriedade desses objetos.

Como resolver diferenças de propriedade de objeto

1. Conecte-se ao banco de dados `babelfish_db` em seu cluster de banco de dados usando o endpoint do PostgreSQL.
2. Execute o seguinte comando:

   ```
   SELECT * from sys.generate_alter_ownership_statements();
   ```

   Isso gera uma lista de instruções SQL destinadas a padronizar a propriedade entre os objetos.
3. Primeiro, execute as instruções geradas em um ambiente de teste para validar o respectivo efeito antes de aplicá-las no ambiente de produção.

Recomendamos que você execute essas instruções para obter um modelo consistente de propriedade de objeto em seu banco de dados.

Diferenças entre o Babelfish para Aurora PostgreSQL e o SQL Server

O Babelfish é um recurso do Aurora PostgreSQL em evolução, com novas funcionalidades adicionadas em cada versão desde a oferta inicial no Aurora PostgreSQL 13.4. É projetado para fornecer semântica T-SQL sobre o PostgreSQL por meio do dialeto T-SQL usando a porta do TDS. Cada nova versão do Babelfish adiciona recursos e funções que melhor se alinham à funcionalidade e ao comportamento da T-SQL, conforme mostrado na tabela Funcionalidades compatíveis no Babelfish por versão. Para obter melhores resultados ao trabalhar com o Babelfish, recomendamos que você entenda as diferenças que existem atualmente entre o T-SQL compatível com o SQL Server e o Babelfish para a versão mais recente. Para saber mais, consulte Diferenças do T-SQL no Babelfish.

Além das diferenças entre o T-SQL compatível com o Babelfish e o SQL Server, talvez você também precise considerar problemas de interoperabilidade entre o Babelfish e o PostgreSQL no contexto do cluster de banco de dados do Aurora PostgreSQL. Conforme mencionado anteriormente, o Babelfish oferece suporte a semântica T-SQL sobre o PostgreSQL por meio do dialeto T-SQL usando a porta do TDS. Ao mesmo tempo, o banco de dados Babelfish também pode ser acessado por meio da porta padrão do PostgreSQL com instruções SQL do PostgreSQL. Se você está pensando em usar as funcionalidades do PostgreSQL e do Babelfish em uma implantação de produção, precisa estar ciente dos possíveis problemas de interoperabilidade entre nomes de esquemas, identificadores, permissões, semântica transacional, vários conjuntos de resultados, agrupamentos padrão e assim por diante. Em termos simples, quando instruções do PostgreSQL ou acesso ao PostgreSQL ocorrem no contexto do Babelfish, pode ocorrer interferência entre o PostgreSQL e o Babelfish e pode afetar a sintaxe, a semântica e a compatibilidade quando novas versões do Babelfish são lançadas. Para obter informações e orientações completas sobre todas as considerações, consulte a Orientação sobre a interoperabilidade do Babelfish na documentação do Babelfish para PostgreSQL.

> **Note**
>
> Antes de usar a funcionalidade nativa do PostgreSQL e a funcionalidade do Babelfish no mesmo contexto de aplicação, é altamente recomendável que você considere os problemas discutidos na Orientação sobre a interoperabilidade do Babelfish na documentação do Babelfish para PostgreSQL. Esses problemas de interoperabilidade (Aurora PostgreSQL e Babelfish) são relevantes somente se você planeja usar a instância de banco de dados do PostgreSQL no mesmo contexto de aplicação do Babelfish.

Tópicos

- Despejo e restauração do Babelfish
- Diferenças do T-SQL no Babelfish
- Níveis de isolamento de transações no Babelfish

Despejo e restauração do Babelfish

A partir das versões 4.0.0 e 3.4.0, os usuários do Babelfish agora podem usar os utilitários de despejo e restauração para fazer backup e restaurar bancos de dados. Consulte mais informações em Babelfish dump and restore. Esse recurso foi desenvolvido com base nos utilitários de despejo e restauração do PostgreSQL. Consulte mais informações em pg_dump e pg_restore. Para usar efetivamente esse recurso no Babelfish, você precisa usar ferramentas baseadas em PostgreSQL que são adaptadas especificamente para o Babelfish. O recurso de backup e restauração do Babelfish difere significativamente daquele do SQL Server. Consulte mais informações sobre essas diferenças em Dump and restore functionality differences : Babelfish and SQL Server. O Babelfish para Aurora PostgreSQL fornece recursos adicionais para fazer backup e restauração de clusters de baco de dados do Amazon Aurora PostgreSQL. Para obter mais informações, consulte Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora.

Diferenças do T-SQL no Babelfish

A seguir, você pode encontrar uma tabela de funcionalidades do T-SQL compatíveis com a versão atual do Babelfish, com algumas notas sobre as diferenças no comportamento do SQL Server.

Para obter mais informações sobre a compatibilidade com várias versões, consulte Funcionalidades compatíveis no Babelfish por versão. Para obter informações sobre recursos que atualmente não são compatíveis, consulte Funcionalidades não compatíveis com o Babelfish.

O Babelfish está disponível com a edição compatível com Aurora PostgreSQL. Para obter mais informações sobre as versões do Babelfish, consulte as Notas de lançamento do Aurora PostgreSQL.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
\ (caractere de continuação de linha)	O caractere de continuação de linha (uma barra invertida antes de uma nova linha) para caracteres e strings hexadecimais não é compatível no momento. Para strings de caracteres, a nova linha da barra invertida é interpretada como caracteres na string.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
	Para strings hexadecimais, a nova linha da barra invertida gera um erro de sintaxe.
@@version	O formato do valor retornado por `@@version` é um pouco diferente do valor retornado pelo SQL Server. Seu código talvez não funcione corretamente se depender da formatação de `@@version` .
Funções agregadas	As funções agregadas são parcialmente compatíveis (AVG, COUNT, COUNT_BIG, GROUPING, MAX, MIN, STRING_AGG e SUM são compatíveis). Para ver uma lista de funções agregadas não compatíveis, consulte Funções que não há suporte.
ALTER TABLE	Suporta adicionar ou soltar uma única coluna ou restrição somente.
ALTER TABLE..ALTER COLUMN	NULL e NOT NULL não podem ser especificados no momento. Para alterar a nulidade de uma coluna, use a instrução ALTER TABLE..{SET|DROP} NOT NULL do PostgreSQL.

AT TIME ZONE	Durante a transição do horário de verão (DST) para o horário padrão, o período sobreposto é exibido usando a diferença em relação ao horário padrão. Para esclarecer, considere o seguinte exemplo:
	`SELECT CONVERT(DATETIME2(0), '2022-10-30T02:00:00', 126) AT TIME ZONE 'Central European Standard Time';`
	`GO;`
	`Result: 2022-10-30 02:00:00 +01:00`

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
Nomes de colunas em branco sem alias de coluna	Os utilitários `sqlcmd` e `psql` lidam com colunas com nomes em branco de maneira diferente: • O `sqlcmd` do SQL Server retorna um nome de coluna em branco. • O `psql` do PostgreSQL retorna um nome de coluna gerado.
Função CHECKSUM	O Babelfish e o SQL Server usam diferentes algoritmos de hash para a função CHECKSUM. Como resultado, os valores de hash gerados pela função CHECKSUM no Babelfish podem ser diferentes daqueles gerados pela função CHECKSUM no SQL Server.
Padrão de coluna	Ao criar um padrão de coluna, o nome da restrição é ignorado. Para descartar um padrão de coluna, utilize a seguinte sintaxe: `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...`
Constraint_name	No SQL Server, os nomes das restrições devem ser exclusivos dentro do esquema ao qual a tabela pertence. No entanto, no Babelfish, isso se aplica somente às restrições PRIMARY KEY e UNIQUE. Outros tipos de restrição não estão sujeitos a essa restrição.
Restrições	O PostgreSQL não oferece suporte à ativação e à desativação de restrições individuais. A instrução é ignorada, e um aviso é gerado.
Restrições com IGNORE_DUP_KEY	Restrições são criadas sem essa propriedade.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
CREATE, ALTER, DROP SERVER ROLE	ALTER SERVER ROLE tem suporte apenas para `sysadmin`. Todas as outras sintaxes são incompatíveis. O usuário T-SQL no Babelfish tem uma experiência semelhante à do SQL Server para os conceitos de login (entidade principal de servidor), banco de dados e usuário de banco de dados (entidade principal de banco de dados).
Cláusulas CREATE, ALTER LOGIN têm suporte com sintaxe limitada	Há suporte para as cláusulas CREATE LOGIN... PASSWORD, ...DEFAULT_DATABASE e ...DEFAUL T_LANGUAGE. Há suporte para a cláusula ALTER LOGIN... PASSWORD, mas não para a cláusula ALTER LOGIN... OLD_PASSWORD. Apenas um login que seja um membro sysadmin pode modificar uma senha.
Agrupamento CREATE DATABASE com distinção entre maiúsculas e minúsculas	Não há suporte para agrupamentos com distinção entre maiúsculas e minúsculas para a instrução CREATE DATABASE.
Palavras-chave e cláusulas CREATE DATABASE	Não há suporte para opções, exceto COLLATE e CONTAINME NT=NONE. A cláusula COLLATE tem suporte e sempre é definida como o valor de `babelfishpg_tsql.server_col lation_name` .
Cláusulas de suporte CREATE SCHEMA...	Você pode utilizar o comando CREATE SCHEMA para criar um esquema vazio. Utilize comandos adicionais para criar objetos de esquema.
Os valores de ID de banco de dados são diferentes no Babelfish	Os bancos de dados primário e tempdb não serão os IDs de banco de dados 1 e 2.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
A função de tipo de data FORMAT é compatível com as limitações a seguir	Não há suporte para meridiano de caractere único. O formato “yyy” no SQL Server retorna 4 dígitos para anos acima de 1.000, mas apenas 3 dígitos para outros. Não há suporte para os formatos “g” e “R” A tradução local “vi-VN” é um pouco diferente.
Identificadores com mais de 63 caracteres	O PostgreSQL oferece suporte a um máximo de 63 caractere s para identificadores. O Babelfish converte identificadores com mais de 63 caracteres em um nome que inclui um hash do nome original. Por exemplo, uma tabela criada como “AB(ABC123456789012345678901234567890123456789012345678901234567890” pode ser convertida para “ABC123456789012345678901234567890123456789012345678901234567890”.
Suporte para colunas IDENTITY	Colunas `IDENTITY` têm suporte para os tipos de dados `tinyint`, `smallint`, `int`, `bigint`. `numeric` e `decimal`. O SQL Server oferece suporte à precisão de 38 casas para tipos de dados `numeric` e `decimal` em colunas IDENTITY. O PostgreSQL oferece suporte à precisão de 19 casas para tipos de dados `numeric` e `decimal` em colunas IDENTITY.
Índices com IGNORE_DU P_KEY	A sintaxe que cria um índice que inclui IGNORE_DUP_KEY cria um índice como se essa propriedade fosse omitida.
Índices com mais de 32 colunas	Um índice não pode incluir mais de 32 colunas. Colunas de índice incluídas contam para o máximo no PostgreSQL, mas não no SQL Server.
Índices (clusterizados)	Índices clusterizados são criados como se NONCLUSTERED tivesse sido especificada.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
Cláusulas de índice	As cláusulas a seguir são ignoradas: FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX , STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUE NTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP e DATA_COMPRESSION
Suporte para JSON	A ordem dos pares nome-valor não é garantida. Mas o tipo de matriz permanece inalterado.
Objetos LOGIN	Todas as opções para objetos LOGIN são incompatíveis, exceto: PASSWORD, DEFAULT_DATABASE, DEFAULT_L ANGUAGE, ENABLE e DISABLE.
Função NEWSEQUENTIALID	Implementada como NEWID; o comportamento sequencial não é garantido. Ao chamar `NEWSEQUENTIALID` , o PostgreSQL gera um novo valor de GUID.
A cláusula OUTPUT tem suporte com as seguintes limitações	OUTPUT e OUTPUT INTO não têm suporte na mesma consulta DML. Não há suporte para referências à tabela não alvo de operações UPDATE ou DELETE em uma cláusula OUTPUT. OUTPUT... DELETED *, INSERTED * não têm suporte na mesma consulta.
Limite de parâmetros de função ou procedimento	O Babelfish oferece suporte a um máximo de 100 parâmetros para um procedimento ou uma função.
ROWGUIDCOL	Essa cláusula é ignorada no momento. Consultas que fazem referência a `$GUIDGOL` causam um erro de sintaxe.
Suporte para objetos SEQUENCE	Objetos SEQUENCE têm suporte para tipos de dados tinyint, smallint, int, bigint, numeric e decimal. O Aurora PostgreSQL oferece suporte à precisão de 19 casas para tipos de dados numéricos e decimais em um objeto SEQUENCE.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
Funções no nível do servidor	A função no nível do servidor `sysadmin` é compatível. Outras funções no nível do servidor (exceto `sysadmin`) não são compatíveis.
Funções em nível de banco de dados diferentes de `db_owner`	As funções de `db_owner` por banco de dados e as funções por banco de dados definidas pelo usuário são compatíveis. Outras funções no nível do banco de dados (exceto db_owner) não são compatíveis.
Palavra-chave SQL `SPARSE`	A palavra-chave SPARSE é aceita e ignorada.
Cláusula de palavra-chave SQL `ON filegroup`	Essa cláusula é ignorada no momento.
Palavras-chave SQL `CLUSTERED` e `NONCLUSTERED` para índices e restrições	O Babelfish aceita e ignora as palavras-chave `CLUSTERED` e `NONCLUSTERED` .
`sysdatabases.cmptlevel`	`sysdatabases.cmptlevel` sempre é definido como 120.
tempdb não é reinicializado na reinicialização	Objetos permanentes (como tabelas e procedimentos) criados em tempdb não são removidos quando o banco de dados é reiniciado.
TEXTIMAGE_ON filegroup	Babelfish ignora a cláusula `TEXTIMAGE_ON` *`filegroup`* .
Precisão do tempo	O Babelfish oferece suporte com precisão de 6 dígitos para segundos fracionários. Nenhum efeito adverso é antecipado com esse comportamento.
Níveis de isolamento de transações	READUNCOMMITTED é tratada da mesma forma que READCOMMITED.
Colunas calculadas virtuais (não persistentes)	Colunas calculadas virtuais são criadas como persistentes.

Funcionalidade ou sintaxe	Descrição do comportamento ou da diferença
Sem cláusula SCHEMABIN DING	Essa cláusula não tem suporte em funções, procedimentos, acionadores ou visualizações. O objeto é criado, mas como se WITH SCHEMABINDING tivesse sido especificada.

Níveis de isolamento de transações no Babelfish

O Babelfish aceita os níveis de isolamento de transações `READ UNCOMMITTED`, `READ COMMITTED` e `SNAPSHOT`. A partir da versão 3.4 do Babelfish, os níveis de isolamento adicionais `REPEATABLE READ` e `SERIALIZABLE` são aceitos. Todos os níveis de isolamento no Babelfish são aceitos com o comportamento dos níveis de isolamento correspondentes no PostgreSQL. O SQL Server e o Babelfish usam mecanismos subjacentes diferentes para implementar níveis de isolamento de transações (bloqueio de acesso simultâneo, bloqueios mantidos por transações, tratamento de erros etc.). E há algumas diferenças sutis na forma como o acesso simultâneo pode funcionar para diferentes workloads. Para ter mais informações sobre esse comportamento do PostgreSQL, consulte Transaction Isolation.

Tópicos

- Visão geral dos níveis de isolamento de transações
- Configurar os níveis de isolamento de transações
- Habilitar ou desabilitar os níveis de isolamento de transações
- Comparar os níveis de isolamento do Babelfish e do SQL Server

Visão geral dos níveis de isolamento de transações

Os níveis originais de isolamento de transações do SQL Server são baseados no bloqueio pessimista, em que existe apenas uma cópia dos dados e as consultas devem bloquear recursos, como linhas, antes de acessá-los. Posteriormente, foi introduzida uma variação do nível de isolamento `READ COMMITTED`. Isso permite o uso de versões em linha para oferecer melhor simultaneidade entre leitores e gravadores usando acesso sem bloqueio. Além disso, um novo nível de isolamento chamado `SNAPSHOT` está disponível. Ele também usa versões de linha para oferecer melhor simultaneidade do que o nível de isolamento `REPEATABLE READ`, evitando bloqueios compartilhados nos dados de leitura que são mantidos até o final da transação.

Ao contrário do SQL Server, todos os níveis de isolamento de transações no Babelfish são baseados no bloqueio positivo (MVCC). Cada transação vê um snapshot dos dados no início da instrução (`READ COMMITTED`) ou no início da transação (`REPEATABLE READ`, `SERIALIZABLE`), independentemente do estado atual dos dados subjacentes. Portanto, o comportamento de execução de transações simultâneas no Babelfish pode ser diferente do comportamento no SQL Server.

Por exemplo, pense em uma transação com nível de isolamento `SERIALIZABLE` que inicialmente seja bloqueada no SQL Server, mas seja bem-sucedida posteriormente. Ela pode acabar falhando no Babelfish devido a um conflito de serialização com uma transação simultânea que lê ou atualiza as mesmas linhas. Também pode haver casos em que a execução de várias transações simultâneas produza um resultado final diferente no Babelfish em comparação ao SQL Server. As aplicações que usam níveis de isolamento devem ser testadas de maneira completa para cenários de concorrência.

Níveis de isolamento no SQL Server	Nível de isolamento do Babelfish	Nível de isolamento do PostgreSQL	Comentários
`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED` é o mesmo que `READ COMMITTED` no Babelfish ou no PostgreSQL
`READ COMMITTED`	`READ COMMITTED`	`READ COMMITTED`	O `READ COMMITTED` do SQL Server se baseia em bloqueio pessimista, enquanto o `READ COMMITTED` do Babelfish é baseado em snapshots (MVCC).
`READ COMMITTED SNAPSHOT`	`READ COMMITTED`	`READ COMMITTED`	Ambos se baseiam em snapshots (MVCC), mas não são exatamente iguais.
`SNAPSHOT`	`SNAPSHOT`	`REPEATABLE READ`	Exatamente o mesmo.
`REPEATABLE READ`	`REPEATABLE READ`	`REPEATABLE READ`	O `REPEATABLE READ` do SQL Server se baseia em bloqueio pessimista, enquanto o

Níveis de isolamento no SQL Server	Nível de isolamento do Babelfish	Nível de isolamento do PostgreSQL	Comentários
			`REPEATABLE READ` do Babelfish é baseado em snapshots (MVCC).
`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE`	O `SERIALIZABLE` do SQL Server é um isolamento pessimista, enquanto o `SERIALIZABLE` do Babelfish é baseado em snapshots (MVCC).

> **Note**
>
> As dicas da tabela não são aceitas atualmente e o comportamento é controlado com o uso do hatch de escape predefinido do Babelfish `escape_hatch_table_hints`.

Configurar os níveis de isolamento de transações

Use o seguinte comando para definir o nível de isolamento de transações:

Example

```
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
 SNAPSHOT | SERIALIZABLE }
```

Habilitar ou desabilitar os níveis de isolamento de transações

Os níveis de isolamento de transações `REPEATABLE READ` e `SERIALIZABLE` estão desabilitados por padrão no Babelfish, sendo necessário habilitá-los explicitamente

definindo o hatch de escape `babelfishpg_tsql.isolation_level_serializable` ou `babelfishpg_tsql.isolation_level_repeatable_read` como `pg_isolation` usando `sp_babelfish_configure`. Para ter mais informações, consulte Gerenciar o tratamento de erros do Babelfish com hatches de escape.

Veja exemplos para habilitar ou desabilitar o uso de `REPEATABLE READ` e `SERIALIZABLE` na sessão atual definindo seus respectivos hatches de escape. Opcionalmente, inclua um parâmetro `server` para definir o hatch de escape para a sessão atual, bem como para todas as novas sessões subsequentes.

Para habilitar o uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` somente na sessão atual:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
```

Para habilitar o uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` na sessão atual e em todas as novas sessões subsequentes:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation',
 'server'
```

Para desabilitar o uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` na sessão atual e nas novas sessões subsequentes:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'off', 'server'
```

Para habilitar o uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` somente na sessão atual:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

Para habilitar o uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` na sessão atual e em todas as novas sessões subsequentes:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation', 'server'
```

Para desabilitar o uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` na sessão atual e nas novas sessões subsequentes:

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'off', 'server'
```

Comparar os níveis de isolamento do Babelfish e do SQL Server

Veja alguns exemplos das nuances na maneira como o SQL Server e o Babelfish implementam os níveis de isolamento ANSI.

> **Note**
>
> - Os níveis de isolamento `REPEATABLE READ` e `SNAPSHOT` são os mesmos no Babelfish.
> - Os níveis de isolamento `READ UNCOMMITTED` e `READ COMMITTED` são os mesmos no Babelfish.

O exemplo a seguir mostra como criar a tabela base para todos os exemplos mencionados abaixo:

```
CREATE TABLE employee (
    id sys.INT NOT NULL PRIMARY KEY,
    name sys.VARCHAR(255)NOT NULL,
    age sys.INT NOT NULL
);
INSERT INTO employee (id, name, age) VALUES (1, 'A', 10);
INSERT INTO employee (id, name, age) VALUES (2, 'B', 20);
INSERT INTO employee (id, name, age) VALUES (3, 'C', 30);
```

Tópicos

- Comparação entre os níveis de isolamento READ UNCOMMITTED do Babelfish e READ UNCOMMITTED do SQL Server
- Comparação entre os níveis de isolamento READ COMMITTED do Babelfish e READ COMMITTED do SQL Server
- Comparação entre os níveis de isolamento READ COMMITTED do Babelfish e READ COMMITTED SNAPSHOT do SQL Server
- Comparação entre os níveis de isolamento REPEATABLE READ do Babelfish e REPEATABLE READ do SQL Server
- Comparação entre os níveis de isolamento SERIALIZABLE do Babelfish e SERIALIZABLE do SQL Server

Comparação entre os níveis de isolamento **`READ UNCOMMITTED`** do Babelfish e **`READ UNCOMMITTED`** do SQL Server

A tabela a seguir fornece detalhes sobre as leituras sujas quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `READ UNCOMMITTED` no SQL Server em comparação com a implementação do Babelfish.

Transação 1	Transação 2	**`READ UNCOMMITTED`** do SQL Server	**`READ UNCOMMITTED`** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Nenhum	Nenhum
Ocioso na transação	`UPDATE employee SET age=0;`	Atualização bem-sucedida.	Atualização bem-sucedida.
Ocioso na transação	`INSERT INTO employee VALUES (4, 'D', 40);`	Inserção bem-sucedida.	Inserção bem-sucedida.

Transação 1	Transação 2	**READ UNCOMMITTED** do SQL Server	**READ UNCOMMITTED** do Babelfish
`SELECT * FROM employee;`	Ocioso na transação	A transação 1 pode ver as alterações não confirmadas da transação 2.	O mesmo que `READ COMMITTED` no Babelfish. As alterações não confirmadas da transação 2 não são visíveis na transação 1.
Ocioso na transação	`COMMIT`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Vê as alterações confirmadas pela transação 2.	Vê as alterações confirmadas pela transação 2.

Comparação entre os níveis de isolamento **`READ COMMITTED`** do Babelfish e **`READ COMMITTED`** do SQL Server

A tabela a seguir fornece detalhes sobre o comportamento do bloqueio das leituras e das gravações quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `READ COMMITTED` no SQL Server em comparação com a implementação do Babelfish.

Transação 1	Transação 2	**READ COMMITTED** do SQL Server	**READ COMMITTED** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nenhum	Nenhum

Transação 1	Transação 2	**READ COMMITTED** do SQL Server	**READ COMMITTED** do Babelfish
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`UPDATE employee SET age=100 WHERE id = 1;`	Atualização bem-sucedida.	Atualização bem-sucedida.
`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	Ocioso na transação	Etapa bloqueada até que a transação 2 seja confirmada.	As alterações da transação 2 ainda não estão visíveis. Updates row with id=3.
Ocioso na transação	`COMMIT`	A transação 2 foi confirmada com êxito. A transação 1 agora está desbloqueada e vê a atualização da transação 2.	A transação 2 foi confirmada com êxito.
`SELECT * FROM employee;`	Ocioso na transação	A transação 1 atualiza a linha com id = 1.	A transação 1 atualiza a linha com id = 3.

Comparação entre os níveis de isolamento **`READ COMMITTED`** do Babelfish e **`READ COMMITTED SNAPSHOT`** do SQL Server

A tabela a seguir fornece detalhes sobre o comportamento do bloqueio das linhas recém-inseridas quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `READ COMMITTED SNAPSHOT` no SQL Server em comparação com a implementação do `READ COMMITTED` no Babelfish.

Transação 1	Transação 2	**READ COMMITTED SNAPSHOT** do SQL Server	**READ COMMITTED** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nenhum	Nenhum
`INSERT INTO employee VALUES (4, 'D', 40);`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`UPDATE employee SET age = 99;`	Etapa bloqueada até que a Transação 1 seja confirmada. A linha inserida está bloqueada pela transação 1.	Três linhas atualizadas. A linha recém-inserida ainda não está visível.
`COMMIT`	Ocioso na transação	Confirmação bem-sucedida. A transação 2 agora está desbloqueada.	Confirmação bem-sucedida.
Ocioso na transação	`SELECT * FROM employee;`	Todas as quatro linhas têm idade = 99.	A linha com id = 4 tem o valor de idade 40, pois não estava visível para a transação 2 durante a consulta de atualização. Outras linhas são atualizadas para idade = 99.

Comparação entre os níveis de isolamento **`REPEATABLE READ`** do Babelfish e **`REPEATABLE READ`** do SQL Server

A tabela a seguir fornece detalhes sobre o comportamento do bloqueio das leituras e das gravações quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `REPEATABLE READ` no SQL Server em comparação com a implementação do `REPEATABLE READ` no Babelfish.

Transação 1	Transação 2	**`REPEATABLE READ`** do SQL Server	**`REPEATABLE READ`** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`SELECT * FROM employee WHERE id != 1;`	Nenhum	Nenhum
Ocioso na transação	`SELECT * FROM employee;`	A transação 2 permanece bloqueada até que a transação 1 seja confirmada.	A transação 2 prossegue normalmente.
`COMMIT`	Ocioso na transação	Nenhum	Nenhum

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
Ocioso na transação	`SELECT * FROM employee;`	A atualização da transação 1 está visível.	A atualização da transação 1 não está visível.
`COMMIT`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`SELECT * FROM employee;`	Vê a atualização da transação 1.	Vê a atualização da transação 1.

A tabela a seguir fornece detalhes sobre o comportamento do bloqueio de gravação-gravação quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `REPEATABLE READ` no SQL Server em comparação com a implementação do `REPEATABLE READ` no Babelfish.

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nenhum	Nenhum
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`UPDATE employee SET name='A_TXN2' WHERE id=1;`	Transação 2 bloqueada.	Transação 2 bloqueada.

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`COMMIT`	Ocioso na transação	A confirmação foi bem-sucedida e a transação 2 foi desbloqueada.	A confirmação foi bem-sucedida e a transação 2 falha com erro, não foi possível serializar o acesso devido à atualização simultânea.
Ocioso na transação	`COMMIT`	Confirmação bem-sucedida.	A transação 2 já foi cancelada.
Ocioso na transação	`SELECT * FROM employee;`	Row with id=1 has name='A_TX2'.	Row with id=1 has name='A_TX1'.

A tabela a seguir fornece detalhes sobre as leituras fantasmas quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `REPEATABLE READ` no SQL Server em comparação com a implementação do `REPEATABLE READ` no Babelfish.

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
Ocioso na transação	`INSERT INTO employee VALUES (4, 'NewRowName', 20);`	A transação 2 prossegue sem nenhum bloqueio.	A transação 2 prossegue sem nenhum bloqueio.
Ocioso na transação	`SELECT * FROM employee;`	A linha recém-inserida está visível.	A linha recém-inserida está visível.
Ocioso na transação	`COMMIT`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	A nova linha inserida pela transação 2 está visível.	A nova linha inserida pela transação 2 não está visível.
`COMMIT`	Ocioso na transação	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	A linha recém-inserida está visível.	A linha recém-inserida está visível.

A tabela a seguir fornece detalhes de quando as transações simultâneas são executadas e os diferentes resultados finais ao usar o nível de isolamento `REPEATABLE READ` no SQL Server em comparação com a implementação do `REPEATABLE READ` do Babelfish.

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nenhum	Nenhum

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`UPDATE employee SET age = 100 WHERE age IN (SELECT MIN(age) FROM employee);`	Ocioso na transação	A transação 1 atualiza a linha com id 1.	A transação 1 atualiza a linha com id 1.
Ocioso na transação	`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	A transação 2 está bloqueada porque a declaração SELECT tenta ler as linhas bloqueadas pela consulta UPDATE na transação 1.	A transação 2 prossegue sem nenhum bloqueio, pois a leitura nunca é bloqueada, a declaração SELECT é executada e, finalmente, a linha com id = 3 é atualizad a, pois as alterações da transação 1 ainda não estão visíveis.
Ocioso na transação	`SELECT * FROM employee;`	Essa etapa é executada após a confirmação da transação 1. A linha com id = 1 é atualizad a pela transação 2 na etapa anterior e está visível aqui.	A linha com o id = 3 é atualizada pela Transação 2.
`COMMIT`	Ocioso na transação	A transação 2 agora está desbloqueada.	Confirmação bem-sucedida.
Ocioso na transação	`COMMIT`	Nenhum	Nenhum

Transação 1	Transação 2	**REPEATABLE READ** do SQL Server	**REPEATABLE READ** do Babelfish
`SELECT * FROM employee;`	Ocioso na transação	As duas transações executam a atualização na linha com id = 1.	Linhas diferentes são atualizadas pelas transações 1 e 2.

Comparação entre os níveis de isolamento **SERIALIZABLE** do Babelfish e **SERIALIZABLE** do SQL Server

A tabela a seguir fornece detalhes sobre os bloqueios de alcance quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `SERIALIZABLE` no SQL Server em comparação com a implementação do `SERIALIZABLE` no Babelfish.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`INSERT INTO employee VALUES (4, 'D', 35);`	A transação 2 permanece bloqueada até que a transação 1 seja confirmada.	A transação 2 prossegue sem nenhum bloqueio.
Ocioso na transação	`SELECT * FROM employee;`	Nenhum	Nenhum
`COMMIT`	Ocioso na transação	A transação 1 foi confirmada com êxito.	A transação 1 foi confirmada com êxito.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
		A transação 2 agora está desbloqueada.	
Ocioso na transação	`COMMIT`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	A linha recém-inserida está visível.	A linha recém-inserida está visível.

A tabela a seguir fornece detalhes de quando as transações simultâneas são executadas e os diferentes resultados finais ao usar o nível de isolamento `SERIALIZABLE` no SQL Server em comparação com a implementação do `SERIALIZABLE` do Babelfish.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nenhum	Nenhum
Ocioso na transação	`INSERT INTO employee VALUES (4, 'D', 40);`	Nenhum	Nenhum
`UPDATE employee SET age =99 WHERE id = 4;`	Ocioso na transação	A transação 1 permanece bloqueada até que a transação 2 seja confirmada.	A transação 1 prossegue sem nenhum bloqueio.
Ocioso na transação	`COMMIT`	A transação 2 foi confirmada com êxito. A transação 1 agora está desbloqueada.	A transação 2 foi confirmada com êxito.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`COMMIT`	Ocioso na transação	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	A linha recém-inserida é visível com valor de idade = 99.	A linha recém-inserida é visível com valor de idade = 40.

A tabela a seguir fornece detalhes quando você aplica `INSERT` em uma tabela com restrição exclusiva. Ela mostra os resultados observados ao usar o nível de isolamento `SERIALIZABLE` no SQL Server em comparação com a implementação do `SERIALIZABLE` no Babelfish.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nenhum	Nenhum
Ocioso na transação	`INSERT INTO employee VALUES (4, 'D', 40);`	Nenhum	Nenhum
`INSERT INTO employee VALUES ((SELECT MAX(id)+1 FROM employee), 'E', 50);`	Ocioso na transação	A transação 1 permanece bloqueada até que a transação 2 seja confirmada.	A transação 1 permanece bloqueada até que a transação 2 seja confirmada.
Ocioso na transação	`COMMIT`	A transação 2 foi confirmada com êxito.	A transação 2 foi confirmada com êxito. A transação 1

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
		A transação 1 agora está desbloqueada.	cancelada com erro de valor de chave duplicado viola a restrição exclusiva.
`COMMIT`	Ocioso na transação	A transação 1 foi confirmada com êxito.	As confirmações da transação 1 falham, pois não foi possível serializar o acesso devido às dependências de leitura ou de gravação entre as transações.
`SELECT * FROM employee;`	Ocioso na transação	row (5, 'E', 50) is inserted.	Existem apenas quatro linhas.

No Babelfish, transações simultâneas executadas com o nível de isolamento serializável falharão com erro de anomalia de serialização se a execução dessas transações for inconsistente com todas as possíveis execuções seriais (uma por vez) dessas transações.

As tabelas a seguir fornecem detalhes sobre a anomalia de serialização quando transações simultâneas são realizadas. Ela mostra os resultados observados ao usar o nível de isolamento `SERIALIZABLE` no SQL Server em comparação com a implementação do `SERIALIZABLE` no Babelfish.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum
`UPDATE employee SET age=5 WHERE age=10;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`SELECT * FROM employee;`	A transação 2 permanece bloqueada até que a transação 1 seja confirmada.	A transação 2 prossegue sem nenhum bloqueio.
Ocioso na transação	`UPDATE employee SET age=35 WHERE age=30;`	Nenhum	Nenhum
`COMMIT`	Ocioso na transação	A transação 1 foi confirmada com êxito.	A transação 1 é confirmada primeiro e pode ser confirmada com êxito.
Ocioso na transação	`COMMIT`	A transação 2 foi confirmada com êxito.	A confirmação da transação 2 falha com erro de serialização, toda a transação foi revertida. Repita a transação 2.

Transação 1	Transação 2	**SERIALIZABLE** do SQL Server	**SERIALIZABLE** do Babelfish
`SELECT * FROM employee;`	Ocioso na transação	As alterações das duas transações estão visíveis.	A transação 2 foi revertida. Somente as alterações da transação 1 são observadas.

No Babelfish, a anomalia de serialização só será possível se todas as transações simultâneas estiverem sendo executadas no nível de isolamento `SERIALIZABLE`. Na tabela a seguir, vamos pensar no exemplo acima, mas definir a transação 2 como nível de isolamento `REPEATABLE READ`.

Transação 1	Transação 2	Níveis de isolamento do SQL Server	Níveis de isolamento do Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nenhum	Nenhum
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nenhum	Nenhum
`SELECT * FROM employee;`	Ocioso na transação	Nenhum	Nenhum
`UPDATE employee SET age=5 WHERE age=10;`	Ocioso na transação	Nenhum	Nenhum
Ocioso na transação	`SELECT * FROM employee;`	A transação 2 permanece bloqueada até que a transação 1 seja confirmada.	A transação 2 prossegue sem nenhum bloqueio.

Transação 1	Transação 2	Níveis de isolamento do SQL Server	Níveis de isolamento do Babelfish
Ocioso na transação	`UPDATE employee SET age=35 WHERE age=30;`	Nenhum	Nenhum
`COMMIT`	Ocioso na transação	A transação 1 foi confirmada com êxito.	A transação 1 foi confirmada com êxito.
Ocioso na transação	`COMMIT`	A transação 2 foi confirmada com êxito.	A transação 2 foi confirmada com êxito.
`SELECT * FROM employee;`	Ocioso na transação	As alterações das duas transações estão visíveis.	As alterações das duas transações estão visíveis.

Usar recursos do Babelfish com implementação limitada

Toda nova versão do Babelfish adiciona compatibilidade com recursos que se alinham melhor à funcionalidade e ao comportamento do T-SQL. Ainda assim, existem alguns recursos e diferenças não compatíveis com a implementação atual. A seguir, você pode encontrar informações sobre diferenças funcionais entre o Babelfish e o T-SQL, com algumas soluções alternativas ou notas de uso.

A partir da versão 1.2.0 do Babelfish, os recursos a seguir no momento têm implementações limitadas:

- Catálogos do SQL Server (visualizações do sistema): os catálogos `sys.sysconfigures`, `sys.syscurconfigs` e `sys.configurations` são compatíveis com uma única configuração somente leitura. O `sp_configure` não é compatível no momento. Para obter mais informações sobre as outras visualizações do SQL Server implementadas pelo Babelfish, consulte Obter informações do catálogo de sistemas do Babelfish.
- Permissões GRANT: GRANT…TO PUBLIC é compatível, mas GRANT…TO PUBLIC WITH GRANT OPTION não é compatível no momento.
- Cadeia de propriedade e limitação do mecanismo de permissão do SQL Server: no Babelfish, a cadeia de propriedade do SQL Server funciona para visualizações, mas não para procedimentos

armazenados. Isso significa que, assim como os procedimentos de chamada, os procedimentos devem ter acesso explícito a outros objetos pertencentes ao mesmo proprietário. No SQL Server, conceder permissões EXECUTE ao chamador no procedimento é suficiente para chamar outros objetos pertencentes ao mesmo proprietário. No Babelfish, o chamador também deve receber permissões nos objetos acessados pelo procedimento.

- Resolução de referências de objeto não qualificados (sem nome do esquema): quando um objeto SQL (procedimento, visualização, função ou acionador) faz referência a um objeto sem o qualificar um nome de esquema, o SQL Server resolve o nome do esquema do objeto usando o nome do esquema do objeto SQL no qual a referência ocorre. No momento, o Babelfish resolve isso de forma diferente, usando o esquema padrão do usuário do banco de dados que executa o procedimento.
- Alterações, sessões e conexões de esquema padrão: se os usuários alterarem o esquema padrão com `ALTER USER...WITH DEFAULT SCHEMA`, a alteração entrará em vigor imediatamente nessa sessão. No entanto, quanto a outras sessões pertencentes ao mesmo usuário que estão conectadas no momento, o tempo difere da seguinte forma:
  - Para SQL Server: a alteração entra em vigor imediatamente em todas as outras conexões para esse usuário.
  - Para Babelfish: a alteração entra em vigor para esse usuário apenas para novas conexões.
- Implementação de tipos de dados ROWVERSION e TIMESTAMP e configuração de hatch de escape: os tipos de dados ROWVERSION e TIMESTAMP agora são compatíveis com o Babelfish. Para usar ROWVERSION ou TIMESTAMP no Babelfish, você deve alterar a configuração do hatch de escape `babelfishpg_tsql.escape_hatch_rowversion` de seu padrão (estrito) para `ignore`. Na maioria das vezes, a implementação do Babelfish dos tipos de dados ROWVERSION e TIMESTAMP é semanticamente idêntica à do SQL Server, com as seguintes exceções:
  - A função @@DBTS integrada se comporta de forma semelhante ao SQL Server, mas com pequenas diferenças. Em vez de retornar o último valor usado para `SELECT @@DBTS`, o Babelfish gera um novo carimbo de data/hora, devido ao mecanismo de banco de dados PostgreSQL subjacente e sua implementação de controle de simultaneidade de várias versões (MVCC).
  - No SQL Server, cada linha inserida ou atualizada recebe um valor ROWVERSION/TIMESTAMP exclusivo. No Babelfish, cada linha inserida atualizada pela mesma instrução recebe o mesmo valor ROWVERSION/TIMESTAMP.

Por exemplo, quando uma instrução UPDATE ou INSERT-SELECT afeta várias linhas, no SQL Server todas as linhas afetadas têm valores diferentes na coluna ROWVERSION/TIMESTAMP. No Babelfish (PostgreSQL), as linhas têm o mesmo valor.

- No SQL Server, ao criar uma tabela com SELECT-INTO, você pode converter um valor explícito (como NULL) em uma coluna ROWVERSION/TIMESTAMP a ser criada. Ao executar essa mesma ação no Babelfish, ele atribui um valor ROWVERSION/TIMESTAMP real a cada linha na nova tabela para você.

Essas pequenas diferenças nos tipos de dados ROWVERSION/TIMESTAMP não devem ter um impacto adverso nas aplicações executadas no Babelfish.

- Cláusula TOP N PERCENT: o Babelfish aceita a cláusula TOP N PERCENT com algumas limitações. As operações SELECT são aceitas, enquanto as operações UPDATE, DELETE e INSERT com TOP N PERCENT, não. A opção WITH TIES e as subconsultas dentro da cláusula TOP também não são aceitas. Quando o valor da expressão excede 100, o comportamento é diferente:
  - Para SQL Server: gera um erro.
  - Para Babelfish: trata o valor como válido e exibe os resultados.
- Criação, propriedade e permissões de esquema: as permissões para criar e acessar objetos em um esquema criado por um usuário não DBO (usando `CREATE SCHEMA` *`schema name`* `AUTHORIZATION` *`user name`*) diferem para usuários não DBO do SQL Server e do Babelfish, conforme mostrado na seguinte tabela:

O usuário do banco de dados (não DBO) que é dono do esquema pode fazer o seguinte:	SQL Server	Babelfish
Criar objetos no esquema sem concessões adicionais pelo DBO?	Não	Sim
Acessar objetos criados por DBO no esquema sem concessões adicionais?	Sim	Não

- Sintaxe CREATE OR ALTER VIEW / ALTER VIEW: essas sintaxes podem ser usadas no Babelfish com as seguintes limitações:

- Essas instruções não podem ser usadas em visualizações que tenham um acionador INSTEAD-OF anexado.
- Essas instruções não podem ser usadas em uma visualização que tenha outra visualização baseada nela.

Melhorar a performance de consultas do Babelfish

Você pode alcançar um processamento de consultas mais rápido no Babelfish usando dicas de consulta e o otimizador do PostgreSQL.

Tópicos

- Usar o plano de explicação para melhorar a performance das consultas do Babelfish
- Usar dicas de consulta T-SQL para melhorar a performance de consultas do Babelfish

Você também pode melhorar a performance de consultas usando o procedimento `sp_babelfish_volatility`. Para ter mais informações, consulte sp_babelfish_volatility.

Você também pode melhorar a performance da consulta usando transformação de subconsultas e cache de subconsultas. Para ter mais informações, consulte Otimizar subconsultas correlacionadas no Aurora PostgreSQL.

Usar o plano de explicação para melhorar a performance das consultas do Babelfish

A partir da versão 2.1.0, o Babelfish inclui duas funções que usam de forma transparente o otimizador do PostgreSQL para gerar planos de consulta estimados e reais para consultas T-SQL na porta do TDS. Essas funções são semelhantes ao uso de SET STATISTICS PROFILE ou SET SHOWPLAN_ALL com bancos de dados do SQL Server para identificar e melhorar consultas de execução lenta.

> **Note**
>
> A obtenção de planos de consulta de funções, fluxos de controle e cursores não é compatível no momento.

Na tabela, você pode encontrar uma comparação das funções de explicação do plano de consulta no SQL Server, no Babelfish e no PostgreSQL.

SQL Server	Babelfish	PostgreSQL
SHOWPLAN_ALL	BABELFISH_SHOWPLAN_ALL	EXPLAIN
STATISTICS PROFILE	BABELFISH_STATISTICS PROFILE	EXPLAIN ANALYZE
Usa o otimizador do SQL Server	Usa o otimizador do PostgreSQL	Usa o otimizador do PostgreSQL
Formato de entrada e saída do SQL Server	Formato de entrada do SQL Server e saída do PostgreSQL	Formato de entrada e saída do PostgreSQL
Definido para a sessão	Definido para a sessão	Aplicar a uma declaração específica
É compatível com o seguinte: • SELECT • INSERT • UPDATE • DELETE • CURSOR • CREATE • EXECUTE • EXEC e funções, incluindo fluxo de controle (CASE, WHILE-BREAK-CONTINUE, WAITFOR, BEGIN-END, IF-ELSE, etc.)	É compatível com o seguinte: • SELECT • INSERT • UPDATE • DELETE • CREATE • EXECUTE • EXEC • RAISEERROR • THROW • PRINT • USE	É compatível com o seguinte: • SELECT • INSERT • UPDATE • DELETE • CURSOR • CREATE • EXECUTE

Use as funções do Babelfish da seguinte forma:

- SET BABELFISH_SHOWPLAN_ALL [ON|OFF]: defina como ON (Ativo) para gerar um plano de execução de consulta estimado. Essa função implementa o comportamento do comando `EXPLAIN` do PostgreSQL. Use esse comando para obter o plano de explicação para determinada consulta.
- SET BABELFISH_STATISTICS PROFILE [ON|OFF]: defina como ON (Ativo) para planos de execução de consulta reais. Essa função implementa o comportamento do comando `EXPLAIN ANALYZE` do PostgreSQL.

Para obter mais informações sobre `EXPLAIN` e `EXPLAIN ANALYZE` do PostgreSQL, consulte EXPLAIN na documentação do PostgreSQL.

> **Note**
>
> A partir da versão 2.2.0, você pode definir o parâmetro `escape_hatch_showplan_all` como ignore para evitar o uso do prefixo BABELFISH_ na sintaxe dos comandos SET `SHOWPLAN_ALL` e `STATISTICS PROFILE` do SQL Server.

Por exemplo, a sequência de comandos a seguir ativa o planejamento de consulta e, depois, retorna um plano de execução de consulta estimado para a instrução SELECT sem executar a consulta. Esse exemplo usa o exemplo de banco de dados do SQL Server `northwind` usando a ferramenta de linha de comando `sqlcmd` para consultar a porta do TDS:

```
1> SET BABELFISH_SHOWPLAN_ALL ON
2> GO
1> SELECT t.territoryid, e.employeeid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE e.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO

QUERY PLAN


------------------------------------------------------------------------------


Query Text: SELECT t.territoryid, e.employeeid FROM
dbo.employeeterritories e, dbo.territories t
WHERE e.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=6231.74..6399.22 rows=66992 width=10)
  Sort Key: t.territoryid NULLS FIRST
```

```
  ->  Nested Loop  (cost=0.00..861.76 rows=66992 width=10)
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1264 width=4)
              Filter: ((territoryid)::"varchar" IS NOT NULL)
        ->  Materialize  (cost=0.00..1.79 rows=53 width=6)
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Quando você terminar de revisar e ajustar sua consulta, desative a função como mostrado a seguir:

```
1> SET BABELFISH_SHOWPLAN_ALL OFF
```

Com BABELFISH_STATISTICS PROFILE definido como ON (Ativo), cada consulta executada retorna seu conjunto de resultados regular seguido de um conjunto de resultados adicional que mostra planos de execução de consulta reais. O Babelfish gera o plano de consulta que fornece o conjunto de resultados mais rápido quando invoca a instrução SELECT.

```
1> SET BABELFISH_STATISTICS PROFILE ON
1>
2> GO
1> SELECT e.employeeid, t.territoryid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE t.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO
```

O conjunto de resultados e o plano de consulta são retornados (esse exemplo mostra apenas o plano de consulta).

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT e.employeeid, t.territoryid FROM
dbo.employeeterritories e, dbo.territories t
WHERE t.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=42.44..43.28 rows=337 width=10)
  Sort Key: t.territoryid NULLS FIRST


  ->  Hash Join  (cost=2.19..28.29 rows=337 width=10)
       Hash Cond: ((e.territoryid)::"varchar" = (t.territoryid)::"varchar")
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1270 width=36)
        ->  Hash  (cost=1.53..1.53 rows=53 width=6)
```

```
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Para saber mais sobre como analisar suas consultas e os resultados retornados pelo otimizador do PostgreSQL, consulte explain.depesz.com. Para obter mais informações sobre EXPLAIN e EXPLAIN ANALYZE do PostgreSQL, consulte EXPLAIN na documentação do PostgreSQL.

Parâmetros que controlam as opções de explicação do Babelfish

Você pode usar os parâmetros mostrados na tabela a seguir para controlar o tipo de informação que é exibida pelo plano de consulta.

Parâmetro	Descrição
babelfishpg_tsql.explain_buffers	Um valor booliano que ativa (e desativa) as informações de uso do buffer para o otimizador. (Padrão: desativado) (Permitido: desativado, ativado)
babelfishpg_tsql.explain_costs	Um valor booliano que ativa (e desativa) as informações estimadas de inicialização e custo total para o otimizador. (Padrão: ativado) (Permitido: desativado, ativado)
babelfishpg_tsql.explain_format	Especifica o formato de saída do plano `EXPLAIN`. (Padrão: texto) (Permitido: texto, xml, json, yaml)
babelfishpg_tsql.explain_settings	Um valor booliano que ativa (ou desativa) a inclusão de informações sobre parâmetros de configuração na saída do plano EXPLAIN. (Padrão: desativado) (Permitido: desativado, ativado)
babelfishpg_tsql.explain_summary	Um valor booliano que ativa (ou desativa) informações resumidas, como o tempo total após o plano de consulta. (Padrão: ativado) (Permitido: desativado, ativado)

Parâmetro	Descrição
babelfishpg_tsql.explain_timing	Um valor booliano que ativa (ou desativa) o tempo real de inicialização e o tempo gasto em cada nó na saída. (Padrão: ativado) (Permitido: desativado, ativado)
babelfishpg_tsql.explain_verbose	Um valor booliano que ativa (ou desliga) a versão mais detalhada de um plano de explicação. (Padrão: desativado) (Permitido: desativado, ativado)
babelfishpg_tsql.explain_wal	Um valor booliano que ativa (ou desativa) a geração de informações de registro WAL como parte de um plano de explicação. (Padrão: desativado) (Permitido: desativado, ativado)

Você pode conferir os valores de quaisquer parâmetros relacionados ao Babelfish em seu sistema usando o cliente do PostgreSQL ou do SQL Server. Execute o seguinte comando para obter os valores atuais dos parâmetros:

```
1> execute sp_babelfish_configure '%explain%';
2> GO
```

Na saída a seguir, você pode ver que todas as configurações nesse cluster de banco de dados do Babelfish específico estão com seus valores padrão. Nem todas as saídas são mostradas neste exemplo.

```
             name                    setting                     short_desc
--------------------------------- --------
 ---------------------------------------------------------
babelfishpg_tsql.explain_buffers   off      Include information on buffer usage
babelfishpg_tsql.explain_costs     on       Include information on estimated startup
 and total cost
babelfishpg_tsql.explain_format    text     Specify the output format, which can be
 TEXT, XML, JSON, or YAML
babelfishpg_tsql.explain_settings  off      Include information on configuration
 parameters
```

```
babelfishpg_tsql.explain_summary   on       Include summary information (e.g.,totaled
 timing information) after the query plan
babelfishpg_tsql.explain_timing    on       Include actual startup time and time spent
 in each node in the output
babelfishpg_tsql.explain_verbose   off      Display additional information regarding
 the plan
babelfishpg_tsql.explain_wal       off      Include information on WAL record
 generation

(8 rows affected)
```

Você pode alterar a configuração desses parâmetros usando `sp_babelfish_configure`, conforme mostrado no exemplo a seguir.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on';
2> GO
```

Se quiser tornar a configuração permanente em todo o cluster, inclua a palavra-chave server, conforme mostrado a seguir:

```
1> execute sp_babelfish_configure 'explain_verbose', 'on', 'server';
2> GO
```

Usar dicas de consulta T-SQL para melhorar a performance de consultas do Babelfish

A partir da versão 2.3.0, o Babelfish comporta o uso de dicas de consulta com `pg_hint_plan`. No Aurora PostgreSQL, `pg_hint_plan` é instalado por padrão. Para obter mais informações sobre a extensão `pg_hint_plan` para PostgreSQL, consulte https://github.com/ossc-db/pg_hint_plan. Para obter detalhes sobre a versão dessa extensão compatível com o Aurora PostgreSQL, consulte Versões de extensão para o Amazon Aurora PostgreSQL em Notas de versão do Aurora PostgreSQL.

O otimizador de consultas é projetado para encontrar o plano de execução ideal para uma instrução SQL. Ao selecionar um plano, o otimizador de consultas considera tanto o modelo de custo do mecanismo quanto as estatísticas de colunas e tabelas. No entanto, o plano sugerido pode não atender às necessidades de seus conjuntos de dados. Assim, as dicas de consulta abordam os problemas de performance para melhorar os planos de execução. Uma `query hint` é uma sintaxe adicionada ao padrão SQL que instrui o mecanismo do banco de dados sobre como executar a consulta. Por exemplo, uma dica pode instruir o mecanismo a seguir uma varredura sequencial e anular qualquer plano que o otimizador de consultas tenha selecionado.

Ativar as dicas de consulta T-SQL no Babelfish

No momento, o Babelfish ignora todas as dicas T-SQL por padrão. Para aplicar as dicas T-SQL, execute o comando `sp_babelfish_configure` com o valor de enable_pg_hint como ON.

```
EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on' [, 'server']
```

É possível tornar as configurações permanentes para todo o cluster ao incluir a palavra-chave server. Para definir a configuração apenas para a sessão atual, não use server.

Depois de ativar `enable_pg_hint`, o Babelfish aplica as dicas T-SQL a seguir.

- Dicas de INDEX
- Dicas de JOIN
- Dica de FORCE ORDER
- Dica de MAXDOP

Por exemplo, a sequência de comandos a seguir ativa `pg_hint_plan`.

```
1> CREATE TABLE t1 (a1 INT PRIMARY KEY, b1 INT);
2> CREATE TABLE t2 (a2 INT PRIMARY KEY, b2 INT);
3> GO
1> EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on';
2> GO
1> SET BABELFISH_SHOWPLAN_ALL ON;
2> GO
1> SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2; --NO HINTS (HASH JOIN)
2> GO
```

Nenhuma dica é aplicada à instrução SELECT. O plano de consulta sem nenhuma dica é retornado.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2
Hash Join (cost=60.85..99.39 rows=2260 width=16)
 Hash Cond: (t1.a1 = t2.a2)
 -> Seq Scan on t1 (cost=0.00..32.60 rows=2260 width=8)
```

```
 -> Hash (cost=32.60..32.60 rows=2260 width=8)
 -> Seq Scan on t2 (cost=0.00..32.60 rows=2260 width=8)
```

```
1> SELECT * FROM t1 INNER MERGE JOIN t2 ON t1.a1 = t2.a2;
2> GO
```

A dica de consulta é aplicada à instrução SELECT. A saída a seguir mostra que o plano de consulta com junção de mesclagem é retornado.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT/*+ MergeJoin(t1 t2) Leading(t1 t2)*/ * FROM t1 INNER JOIN t2 ON
 t1.a1 = t2.a2
Merge Join (cost=0.31..190.01 rows=2260 width=16)
 Merge Cond: (t1.a1 = t2.a2)
 -> Index Scan using t1_pkey on t1 (cost=0.15..78.06 rows=2260 width=8)
 -> Index Scan using t2_pkey on t2 (cost=0.15..78.06 rows=2260 width=8)
```

```
1> SET BABELFISH_SHOWPLAN_ALL OFF;
2> GO
```

Limitações

Ao usar as dicas de consulta, considere as seguintes limitações:

- Se um plano de consulta for armazenado em cache antes da ativação de `enable_pg_hint`, as dicas não serão aplicadas na mesma sessão. Elas serão aplicadas na nova sessão.
- Se nomes de esquemas forem fornecidos explicitamente, não será possível aplicar as dicas. Você pode usar aliases de tabela como uma solução alternativa.
- Uma dica de consulta não pode ser aplicada a visualizações e subconsultas.
- As dicas não funcionam para instruções UPDATE/DELETE com JOINs.
- Uma dica de índice para um índice ou tabela inexistente é ignorada.
- A dica FORCE ORDER não funciona para HASH JOINs e não ANSI JOINs.

Usar extensões do Aurora PostgreSQL com o Babelfish

O Aurora PostgreSQL fornece extensões para trabalhar com outros serviços da AWS. Essas são extensões opcionais que oferecem suporte a vários casos de uso, como usar o Simple Storage Service (Amazon S3) com seu cluster de banco de dados para importar ou exportar dados.

- Para importar dados de um bucket do Amazon S3 para seu cluster de banco de dados do Babelfish, configure a extensão `aws_s3` do Aurora PostgreSQL. Essa extensão também permite exportar dados do seu cluster de bancos de dados Aurora PostgreSQL para um bucket do Simple Storage Service (Amazon S3).
- O AWS Lambda é um serviço de computação que permite executar código sem o provisionamento ou gerenciamento de servidores. Por exemplo, você pode usar funções Lambda para realizar certas ações, como processar notificações de eventos de uma instância de banco de dados. Para saber mais sobre o Lambda, consulte O que é o AWS Lambda? no Guia do desenvolvedor do AWS Lambda. Para invocar funções do Lambda a partir do cluster de banco de dados do Babelfish, configure a extensão `aws_lambda` do Aurora PostgreSQL.

Para configurar essas extensões para o cluster do Babelfish, primeiro você precisa conceder permissão ao usuário interno do Babelfish para carregar as extensões. Depois de conceder permissão, você pode carregar extensões do Aurora PostgreSQL.

Habilitar extensões do Aurora PostgreSQL em seu cluster de banco de dados do Babelfish

Antes que você possa carregar o `aws_s3` ou as extensões do `aws_lambda`, você concede os privilégios necessários ao cluster de banco de dados do Babelfish.

O procedimento a seguir usa a ferramenta da linha de comando `psql` do PostgreSQL para se conectar ao cluster de banco de dados. Para obter mais informações, consulte Utilizar o psql para se conectar ao cluster de banco de dados. Você também pode usar o pgAdmin. Para obter detalhes, consulte Utilizar o pgAdmin para se conectar ao cluster de banco de dados.

Este procedimento carrega ambos `aws_s3` e `aws_lambda`, um após o outro. Você não precisa carregar os dois se quiser usar apenas uma dessas extensões. A extensão `aws_commons` é necessária para cada um, e é carregada por padrão, conforme mostrado na saída.

Para configurar seu cluster de banco de dados do Babelfish com privilégios para as extensões do Aurora PostgreSQL

1. Conecte-se ao seu cluster de banco de dados do Babelfish. Use o nome para o usuário "primário" (-U) que você especificou quando criou o cluster de banco de dados do Babelfish. O padrão (`postgres`) é mostrado nos exemplos.

   Para Linux, macOS ou Unix:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com \
   -U postgres \
   -d babelfish_db \
   -p 5432
   ```

   Para Windows:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com ^
   -U postgres ^
   -d babelfish_db ^
   -p 5432
   ```

   O comando responde com um prompt para inserir a senha para o nome de usuário (-U).

   ```
   Password:
   ```

   Insira a senha do nome de usuário (-U) para o cluster de banco de dados. Quando a conexão é bem-sucedida, a saída é semelhante à seguinte:

   ```
   psql (13.4)
   SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
    compression: off)
   Type "help" for help.

   postgres=>
   ```

2. Conceda privilégios ao usuário interno do Babelfish para criar e carregar extensões.

   ```
   babelfish_db=> GRANT rds_superuser TO master_dbo;
   GRANT ROLE
   ```

3. Crie e carregue a extensão `aws_s3`. A extensão `aws_commons` é necessária e é instalada automaticamente quando o `aws_s3`está instalado.

```
babelfish_db=> create extension aws_s3 cascade;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

4. Crie e carregue a extensão `aws_lambda`.

```
babelfish_db=> create extension aws_lambda cascade;
CREATE EXTENSION
babelfish_db=>
```

Usar o Babelfish com o Simple Storage Service (Amazon S3)

Se ainda não tiver um bucket do Simple Storage Service (Amazon S3) para usar com o cluster de banco de dados do Babelfish, crie um. Para qualquer bucket do Simple Storage Service (Amazon S3) que você deseja usar, você fornece acesso.

Antes de tentar importar ou exportar dados usando um bucket do Simple Storage Service (Amazon S3), conclua as etapas únicas a seguir.

Para configurar o acesso da instância de banco de dados do Babelfish ao bucket do Simple Storage Service (Amazon S3)

1. Crie um bucket do Simple Storage Service (Amazon S3) para a instância do Babelfish, se necessário. Para isso, siga as instruções em Criar um bucket no Guia do usuário do Amazon Simple Storage Service.
2. Carregue arquivos no seu bucket do Amazon S3. Para isso, siga as etapas em Adicionar um objeto a um bucket no Guia do usuário do Amazon Simple Storage Service.
3. Configure permissões conforme necessário:

   - Para importar dados do Amazon S3, o cluster de banco de dados do Babelfish precisa de permissão para acessar o bucket. Recomendamos usar uma função AWS Identity and Access Management (IAM) e anexar uma política do IAM a essa função para o cluster. Para isso, siga as etapas em Usar uma função do IAM para acessar um bucket do Amazon S3.
   - Para exportar dados de seu cluster de banco de dados do Babelfish, seu cluster deve ter acesso ao bucket do Amazon S3. Assim como na importação, recomendamos o uso de uma

função e uma política do IAM. Para isso, siga as etapas em Configurar o acesso a um bucket do Amazon S3.

Agora você pode usar o Amazon S3 com a extensão `aws_s3` com seu cluster de banco de dados do Babelfish.

Para importar dados do Simple Storage Service (Amazon S3) para o Babelfish e exportar dados do Babelfish para o Simple Storage Service (Amazon S3)

1. Use a extensão `aws_s3` com seu cluster de banco de dados do Babelfish.

   Ao fazer isso, certifique-se de fazer referência às tabelas conforme elas existem no contexto do PostgreSQL. Ou seja, se você quiser importar para uma tabela Babelfish chamada `[database].[schema].[tableA]`, consulte essa tabela como `database_schema_tableA` na função `aws_s3`:

   - Para um exemplo de uso de uma função `aws_s3` para importar dados, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL.
   - Para exemplos de uso de funções `aws_s3` para exportar dados, consulte Exportar dados de consulta usando a função aws_s3.query_export_to_s3.
2. Certifique-se de fazer referência a tabelas do Babelfish usando a nomeação do PostgreSQL ao usar a extensão `aws_s3` e o Simple Storage Service (Amazon S3), conforme mostrado na tabela a seguir.

Mesa Babelfish	Mesa Aurora PostgreSQL
`database``.`*`schema`*`.`*`table`*	*`database_schema_table`*

Para saber mais sobre como usar o Simple Storage Service (Amazon S3) com o Aurora PostgreSQL, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL e Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3.

Usar o Babelfish com o AWS Lambda

Depois que a extensão `aws_lambda` é carregada em seu cluster de banco de dados do Babelfish, mas antes que você possa chamar funções Lambda, você dá acesso ao Lambda ao cluster de banco de dados seguindo este procedimento.

Para configurar o acesso para o cluster de banco de dados do Babelfish para trabalhar com o Lambda

Este procedimento usa a AWS CLI para criar a política e o perfil do IAM e associá-los ao cluster de banco de dados do Babelfish.

1. Crie uma política do IAM que permita acesso ao Lambda a partir do seu cluster de banco de dados do Babelfish.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
 "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

2. Crie uma função do IAM que a política possa assumir em tempo de execução.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Anexe a política ao perfil.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Anexar o perfil ao seu cluster de banco de dados do Babelfish

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Depois de concluir essas tarefas, você pode invocar suas funções Lambda. Para obter mais informações e exemplos de configuração do AWS Lambda para o cluster de bancos de dados Aurora PostgreSQL com o AWS Lambda, consulte Etapa 2: configurar o IAM para o cluster de bancos de dados do Aurora PostgreSQL e o AWS Lambda.

Para invocar uma função Lambda a partir do seu cluster de banco de dados do Babelfish

AWS LambdaO suporta funções escritas em Java, Node.js, Python, Ruby e outras linguagens. Se a função retornar texto quando invocada, você poderá chamá-la do cluster de banco de dados do Babelfish. O exemplo a seguir é uma função placeholder do python que retorna uma saudação.

```
lambda_function.py
import json
def lambda_handler(event, context):
    #TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
```

Atualmente, o Babelfish não é compatível com o JSON. Se sua função retornar JSON, você usa um wrapper para lidar com o JSON. Por exemplo, digamos que a `lambda_function.py` mostrada antes é armazenada no Lambda como `my-function`.

1. Conecte-se ao seu cluster de banco de dados do Babelfish usando o cliente `psql` (ou o cliente pgAdmin). Para obter mais informações, consulte Utilizar o psql para se conectar ao cluster de banco de dados.
2. Crie o wrapper. Este exemplo usa a linguagem processual do PostgreSQL para SQL , PL/pgSQL. Para saber mais, consulte Linguagem processual PL/pgSQL–SQL.

   ```
   create or replace function master_dbo.lambda_wrapper()
   returns text
   language plpgsql
   ```

```
as
$$
declare
   r_status_code integer;
   r_payload text;
begin
   SELECT payload INTO r_payload
     FROM aws_lambda.invoke(  aws_commons.create_lambda_function_arn('my-function',
 'us-east-1')
                            ,'{"body": "Hello from Postgres!"}'::json );
   return r_payload ;
end;
$$;
```

A função agora pode ser executada a partir da porta Babelfish TDS (1433) ou da porta PostgreSQL (5433).

a. Para invocar (chamar) essa função a partir da sua porta PostgreSQL:

```
SELECT * from aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function', 'us-east-1'), '{"body": "Hello from Postgres!"}'::json );
```

A saída é semelhante à seguinte:

```
status_code |                        payload                         |
 executed_version | log_result
-------------+--------------------------------------------------------
+-----------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
         |
(1 row)
```

b. Para invocar (chamar) essa função a partir da porta TDS, conecte-se à porta usando o cliente da linha de comando `sqlcmd` do SQL Server. Para obter detalhes, consulte Utilizar um cliente SQL Server para se conectar ao seu cluster de banco de dados. Quando conectado, execute o seguinte:

```
1> select lambda_wrapper();
2> go
```

Esse comando retorna uma saída semelhante à seguinte:

```
{"statusCode": 200, "body": "\"Hello from Lambda!\""}
```

Para saber mais sobre como usar o Lambda com o Aurora PostgreSQL, consulte Invocar uma função do AWS Lambda de um cluster de banco de dados do Aurora PostgreSQL. Para obter mais informações sobre como usar funções Lambda, consulte Conceitos básicos do Lambda no Guia do desenvolvedor do AWS Lambda.

Usando pg_stat_statements no Babelfish

O Babelfish para Aurora PostgreSQL é compatível com a extensão `pg_stat_statements` a partir da versão 3.3.0. Para saber mais, consulte pg_stat_statements.

Para obter detalhes sobre a versão dessa extensão compatível com o Aurora PostgreSQL, consulte Versões de extensão.

Criar a extensão pg_stat_statements

Para ativar `pg_stat_statements`, você deve ativar o cálculo do identificador de consulta. Isso será feito automaticamente se `compute_query_id` estiver definido como on ou `auto` no grupo de parâmetros. O valor padrão do parâmetro `compute_query_id` é `auto`. Também é necessário criar essa extensão para ativar esse atributo. Use o comando a seguir para instalar a extensão do endpoint T-SQL:

```
1>EXEC sp_execute_postgresql 'CREATE EXTENSION pg_stat_statements WITH SCHEMA sys';
```

Você pode acessar as estatísticas da consulta usando a seguinte consulta:

```
postgres=>select * from pg_stat_statements;
```

> **Note**
>
> Durante a instalação, se você não fornecer o nome do esquema para a extensão, por padrão, ela o criará no esquema público. Para acessá-lo, você deve usar colchetes com o qualificador de esquema, conforme mostrado abaixo:

```
postgres=>select * from [public].pg_stat_statements;
```

Você também pode criar a extensão a partir do endpoint PSQL.

Autorizar a extensão

Por padrão, você pode ver as estatísticas das consultas realizadas em seu banco de dados T-SQL sem a necessidade de qualquer autorização.

Para acessar as estatísticas de consultas criadas por outras pessoas, você precisa ter a função `pg_read_all_stats` PostgreSQL. Siga as etapas mencionadas abaixo para construir o comando GRANT pg_read_all_stats.

1. No T-SQL, use a consulta a seguir que retorna o nome da função interna do PG.

   ```
   SELECT rolname FROM pg_roles WHERE oid = USER_ID();
   ```

2. Conecte-se ao banco de dados do Babelfish para Aurora PostgreSQL com o privilégio rds_superuser e use o seguinte comando:

   ```
   GRANT pg_read_all_stats TO <rolname_from_above_query>
   ```

Exemplo

No endpoint do T-SQL:

```
1>SELECT rolname FROM pg_roles WHERE oid = USER_ID();
2>go
```

```
rolname
-------
```

```
master_dbo
(1 rows affected)
```

No endpoint do PSQL:

```
babelfish_db=# grant pg_read_all_stats to master_dbo;
```

```
GRANT ROLE
```

Você pode acessar as estatísticas da consulta usando a exibição pg_stat_statements:

```
1>create table t1(cola int);
2>go
1>insert into t1 values (1),(2),(3);
2>go
```

```
(3 rows affected)
```

```
1>select userid, dbid, queryid, query from pg_stat_statements;
2>go
```

```
userid dbid queryid             query
------ ---- -------             -----
37503 34582 6487973085327558478 select * from t1
37503 34582 6284378402749466286 SET QUOTED_IDENTIFIER OFF
37503 34582 2864302298511657420 insert into t1 values ($1),($2),($3)
10    34582 NULL                <insufficient privilege>
37503 34582 5615368793313871642 SET TEXTSIZE 4096
37503 34582 639400815330803392  create table t1(cola int)
(6 rows affected)
```

Redefinir as estatísticas de consulta

Você pode usar `pg_stat_statements_reset()` para redefinir as estatísticas coletadas até agora por pg_stat_statements. Para saber mais, consulte pg_stat_statements. Atualmente, isso é

compatível somente por meio do endpoint PSQL. Conecte-se ao Babelfish para Aurora PostgreSQL com o privilégio `rds_superuser` e use o seguinte comando:

```
SELECT pg_stat_statements_reset();
```

Limitações

- Atualmente, `pg_stat_statements()` não é compatível com o endpoint T-SQL. A exibição `pg_stat_statements` é a forma recomendada de coletar as estatísticas.
- Algumas das consultas podem ser reescritas pelo analisador T-SQL implementado pelo mecanismo Aurora PostgreSQL. A exibição `pg_stat_statements` mostrará a consulta reescrita e não a consulta original.

  Exemplo

  ```
  select next value for [dbo].[newCounter];
  ```

  A consulta acima foi reescrita da seguinte forma na exibição pg_stat_statements.

  ```
  select nextval($1);
  ```
- Com base no fluxo de execução das instruções, algumas das consultas podem não ser rastreadas por pg_stat_statements e não estarão visíveis na exibição. Isso inclui as seguintes declarações: `use dbname`, `goto`, `print`, `raise error`, `set`, `throw`, `declare cursor`.
- Para as instruções CREATE LOGIN e ALTER LOGIN, query e queryid não serão mostrados. Isso mostrará privilégios insuficientes.
- A exibição `pg_stat_statements` sempre contém as duas entradas abaixo, pois elas são executadas internamente pelo cliente `sqlcmd`.
  - SET QUOTED_IDENTIFIER OFF
  - DEFINIR TAMANHO DO TEXTO 4096

Usar o pgvector no Babelfish

O pgvector, uma extensão de código aberto, permite pesquisar dados semelhantes diretamente no banco de dados Postgres. O Babelfish comporta essa extensão a partir das versões 15.6 e 16.2. Para ter mais informações, consulte a documentação de código aberto do pgvector.

Pré-requisitos

Para habilitar a funcionalidade pgvector, instale a extensão no esquema sys usando um dos seguintes métodos:

- Execute o seguinte comando no cliente sqlcmd:

```
exec sys.sp_execute_postgresql 'CREATE EXTENSION vector WITH SCHEMA sys';
```

- Conecte-se ao `babelfish_db` e execute o seguinte comando no cliente psql:

```
CREATE EXTENSION vector WITH SCHEMA sys;
```

> **Note**
>
> Depois de instalar a extensão pgvector, os dados do tipo vetorial só estarão disponíveis nas novas conexões de banco de dados que você estabelecer. As conexões existentes não reconhecerão o novo tipo de dados.

Funções compatíveis

O Babelfish estende a funcionalidade T-SQL para comportar o seguinte:

- Armazenamento

  O Babelfish agora aceita sintaxe compatível com dados do tipo vetorial, aprimorando a compatibilidade com T-SQL. Para saber mais sobre como armazenar dados com o pgvector, consulte Storing.

- Consultas

  O Babelfish expande o suporte à expressão T-SQL para incluir operadores de similaridade vetorial. No entanto, para todas as outras consultas, a sintaxe T-SQL padrão ainda é necessária.

> **Note**
>
> O T-SQL não comporta o tipo Array e os drivers do banco de dados não têm nenhuma interface para lidar com eles. Como solução alternativa, o Babelfish usa strings de texto (varchar/nvarchar) para armazenar dados vetoriais. Por exemplo, quando você solicita

um valor vetorial [1,2,3], o Babelfish exibe uma string “[1,2,3]” como resposta. É possível analisar e dividir essa string em nível de aplicação de acordo com suas necessidades.

Para saber mais sobre como consultar dados com o pgvector, consulte Querying.

- Indexação

  O T-SQL `Create Index` agora aceita a sintaxe `USING INDEX_METHOD`. Agora você pode definir o operador de pesquisa por similaridade a ser usado em uma coluna específica ao criar um índice.

  A gramática também é estendida para comportar operações de similaridade vetorial na coluna necessária (confira a gramática column_name_list_with_order_for_vector).

```
CREATE [UNIQUE] [clustered] [COLUMNSTORE] INDEX <index_name> ON <table_name> [USING
 vector_index_method] (<column_name_list_with_order_for_vector>)
Where column_name_list_with_order_for_vector is:
    <column_name> [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS | VECTOR_L2_OPS]
 (COMMA simple_column_name [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS |
 VECTOR_L2_OPS])
```

  Para saber mais sobre a indexação de dados com pgvector, consulte Indexing.

- desempenho
  - Use `SET BABELFISH_STATISTICS PROFILE ON` para depurar os planos de consulta do endpoint T-SQL.
  - Aumente `max_parallel_workers_get_gather` usando a função `set_config` aceita no T-SQL.
  - Use `IVFFlat` para pesquisas aproximadas. Para ter mais informações, consulte IVFFlat.

  Para melhorar a performance com o pgvector, consulte Performance.

Limitações

- O Babelfish não aceita pesquisa de texto completo para pesquisa híbrida. Para ter mais informações, consulte Hybrid Search.
- No momento, o Babelfish não comporta a funcionalidade de reindexação. No entanto, ainda é possível usar o endpoint PostgreSQL para reindexar. Para ter mais informações, consulte Vacuuming.

Usar o machine learning do Amazon Aurora com Babelfish

É possível ampliar os recursos do cluster de banco de dados do Babelfish para Aurora PostgreSQL integrando-o ao machine learning do Amazon Aurora. Essa integração perfeita concede acesso a uma série de serviços avançados, como o Amazon Comprehend, o Amazon SageMaker AI ou o Amazon Bedrock, cada um adaptado para atender às necessidades distintas de machine learning.

Como usuário do Babelfish, é possível usar o conhecimento existente da sintaxe e da semântica do T-SQL ao trabalhar com o machine learning do Aurora. Siga as instruções fornecidas na documentação da AWS do Aurora PostgreSQL. Para obter mais informações, consulte Usar machine learning do Amazon Aurora com o Aurora PostgreSQL.

Pré-requisitos

- Antes de tentar configurar o cluster de banco de dados do Babelfish para Aurora PostgreSQL para usar o machine learning do Aurora, é necessário entender os requisitos e os pré-requisitos relacionados. Para obter mais informações, consulte Requisitos para usar o machine learning do Aurora com o Aurora PostgreSQL.
- Não se esqueça de instalar a extensão `aws_ml` usando o endpoint Postgres ou o procedimento de armazenamento `sp_execute_postgresql`.

```
exec sys.sp_execute_postgresql 'Create Extension aws_ml'
```

> **Note**
>
> No momento, o Babelfish não aceita operações em cascata com `sp_execute_postgresql` no Babelfish. Como `aws_ml` depende de `aws_commons`, você precisará instalá-lo separadamente usando o endpoint Postgres.

```
create extension aws_common;
```

Tratar a sintaxe e a semântica do T-SQL com funções **`aws_ml`**

Os exemplos a seguir explicam como a sintaxe e a semântica do T-SQL são aplicadas aos serviços de ML da Amazon:

Example: aws_bedrock.invoke_model: uma consulta simples utilizando as funções do Amazon Bedrock.

```
aws_bedrock.invoke_model(
   model_id      varchar,
   content_type  text,
   accept_type   text,
   model_input   text)
Returns Varchar(MAX)
```

O exemplo a seguir mostra como invocar um modelo do Anthropic Claude 2 para o Bedrock usando invoke_model.

```
SELECT aws_bedrock.invoke_model (
    'anthropic.claude-v2', -- model_id
    'application/json', -- content_type
    'application/json', -- accept_type
    '{"prompt": "\n\nHuman:
    You are a helpful assistant that answers questions directly
    and only using the information provided in the context below.
    \nDescribe the answerin detail.\n\nContext: %s \n\nQuestion:
    %s \n\nAssistant:","max_tokens_to_sample":4096,"temperature"
    :0.5,"top_k":250,"top_p":0.5,"stop_sequences":[]}' -- model_input
);
```

Example: aws_comprehend.detect_sentiment: uma consulta simples utilizando as funções do Amazon Comprehend.

```
aws_comprehend.detect_sentiment(
   input_text varchar,
   language_code varchar,
   max_rows_per_batch int)
Returns table (sentiment varchar, confidence real)
```

O código de exemplo a seguir mostra como invocar o serviço do Amazon Comprehend.

```
select sentiment from aws_comprehend.detect_sentiment('This is great', 'en');
```

Example: aws_sagemaker.invoke_endpoint: uma consulta simples usando as funções do Amazon SageMaker.

```
aws_sagemaker.invoke_endpoint(
  endpoint_name varchar,
  max_rows_per_batch int,
  VARIADIC model_input "any") -- Babelfish inherits PG's variadic parameter type
Rerurns Varchar(MAX)
```

Como model_input está marcado como VARIADIC e com o tipo “any”, os usuários podem transmitir uma lista de qualquer tamanho e de qualquer tipo de dados para a função, que atuará como entrada para o modelo. O código de exemplo a seguir mostra como invocar o serviço do Amazon SageMaker.

```
SELECT CAST (aws_sagemaker.invoke_endpoint(
    'sagemaker_model_endpoint_name',
    NULL,
    arg1, arg2 -- model inputs are separate arguments )
AS INT) -- cast the output to INT
```

Para ter mais informações detalhadas sobre como usar o machine learning do Aurora com o Aurora PostgreSQL, consulte Usar machine learning do Amazon Aurora com o Aurora PostgreSQL.

Limitações

- Embora o Babelfish não permita a criação de matrizes, ele ainda pode tratar dados que representam matrizes. Quando você usa funções como `aws_bedrock.invoke_model_get_embeddings` que exibem matrizes, os resultados são entregues como uma string com os elementos da matriz.

O Babelfish é compatível com servidores vinculados

O Babelfish para Aurora PostgreSQL é compatível com servidores vinculados usando a extensão do PostgreSQL `tds_fdw` na versão 3.1.0. Para trabalhar com servidores vinculados, é necessário

instalar a extensão `tds_fdw`. Para ter mais informações sobre a extensão `tds_fdw`, consulte Trabalhar com os invólucros de dados externos compatíveis do Amazon Aurora PostgreSQL.

Instalar a extensão **`tds_fdw`**

É possível instalar a extensão `tds_fdw` usando os métodos a seguir.

Usar CREATE EXTENSION no endpoint do PostgreSQL

1. Conecte-se à sua instância de banco de dados PostgreSQL no banco de dados Babelfish na porta PostgreSQL. Use uma conta que tenha o perfil rds_superuser.

```
psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
username=test --dbname=babelfish_db --password
```

2. Instale a extensão `tds_fdw`. É um processo de instalação único. Não é necessária a reinstalação quando o cluster de banco de dados é reiniciado.

```
babelfish_db=> CREATE EXTENSION tds_fdw;
CREATE EXTENSION
```

Chamar o procedimento **`sp_execute_postgresql`** armazenado pelo endpoint do TDS

O Babelfish é compatível com a instalação da extensão `tds_fdw` chamando o procedimento `sp_execute_postgresql` na versão 3.3.0. É possível executar as instruções do PostgreSQL no endpoint do T-SQL sem sair da porta do T-SQL. Para obter mais informações, consulte . Trabalhar com o Babelfish para Aurora PostgreSQL

1. Conecte-se à sua instância de banco de dados do PostgreSQL no banco de dados do Babelfish na porta do PostgreSQL.

```
sqlcmd -S your-DB-instance.aws-region.rds.amazonaws.com -U test -P password
```

2. Instale a extensão `tds_fdw`.

```
1>EXEC sp_execute_postgresql N'CREATE EXTENSION tds_fdw';
2>go
```

Funções compatíveis

O Babelfish é compatível com a adição de endpoints RDS remotos para SQL Server ou Babelfish para Aurora PostgreSQL como servidor vinculado. Você também pode adicionar outras instâncias remotas do SQL Server como servidores vinculados. Depois, use `OPENQUERY()` para recuperar dados desses servidores vinculados. A partir da versão 3.2.0 do Babelfish, nomes de quatro partes também são compatíveis.

Os procedimentos armazenados e visualizações de catálogo a seguir são compatíveis com o uso dos servidores vinculados.

Procedimentos armazenados

- sp_addlinkedserver: o Babelfish não é compatível com o parâmetro `@provstr`.
- sp_addlinkedsrvlogin
  - Você deve fornecer um nome de usuário e senha remotos explícitos para se conectar à fonte de dados remota. Não é possível se conectar com as credenciais próprias do usuário. O Babelfish é compatível apenas com `@useself = false`.
  - O Babelfish não é compatível com o parâmetro `@locallogin`, pois não há suporte à configuração do acesso remoto ao servidor específico para login local.
- sp_linkedservers
- sp_helplinkedsrvlogin
- sp_dropserver
- sp_droplinkedsrvlogin: o Babelfish não é compatível com o parâmetro `@locallogin`, pois não há suporte à configuração do acesso remoto ao servidor específico para login local.
- sp_serveroption: o Babelfish suporta as seguintes opções de servidor:
  - tempo limite de consulta (da versão 3.2.0 do Babelfish)
  - tempo limite de conexão (da versão 3.3.0 do Babelfish)
- sp_testlinkedserver (da versão 3.3.0 do Babelfish)
- sp_enum_oledb_providers (da versão 3.3.0 do Babelfish)

Visualizações do catálogo

- sys.servers
- sys.linked_logins

Usar criptografia em trânsito para a conexão

A conexão do servidor de origem do Babelfish para Aurora PostgreSQL com o servidor de destino usa criptografia em trânsito (TLS/SSL), dependendo da configuração do banco de dados do servidor remoto. Se o servidor remoto não estiver configurado para criptografia, o servidor do Babelfish que faz a solicitação ao banco de dados remoto retornará ao estado de não criptografado.

Como impor a criptografia de conexão

- Se o servidor vinculado de destino for uma instância do RDS para SQL Server, defina `rds.force_ssl = on` para a instância de destino do SQL Server. Para ter mais informações sobre a configuração de SSL/TLS para o RDS para SQL Server, consulte Usar SSL com uma instância de banco de dados do Microsoft SQL Server.
- Se o servidor vinculado de destino for um cluster do Babelfish para Aurora PostgreSQL, defina `babelfishpg_tds.tds_ssl_encrypt = on` e `ssl = on` para o servidor de destino. Para ter mais informações sobre SSL/TLS, consulte Configurações SSL e conexões do cliente do Babelfish.

Adicionar o Babelfish como um servidor vinculado do SQL Server

O Babelfish para Aurora PostgreSQL pode ser adicionado como um servidor vinculado de um SQL Server. Em um banco de dados do SQL Server, você pode adicionar o Babelfish como um servidor vinculado usando o provedor de banco de dados do Microsoft OLE para ODBC: MSDASQL.

Há duas maneiras de configurar o Babelfish como um servidor vinculado do SQL Server usando o provedor MSDASQL:

- Fornecendo a string de conexão ODBC como string do provedor.
- Forneça o DSN do sistema da fonte de dados ODBC ao adicionar o servidor vinculado.

Limitações

- OPENQUERY() funciona somente para SELECT e não para DML.
- Os nomes de objetos em quatro partes funcionam somente para leitura e não para modificar a tabela remota. Um UPDATE pode referenciar uma tabela remota na cláusula FROM sem a modificar.
- A execução de procedimentos armazenados em servidores vinculados ao Babelfish não é compatível.

- O upgrade da versão principal do Babelfish pode não funcionar se houver objetos dependentes de `OPENQUERY()` ou objetos referenciados por meio de nomes de quatro partes. Você deve garantir que todos os objetos que fazem referência a `OPENQUERY()` ou a nomes de quatro partes sejam descartados antes de um upgrade de versão principal.
- Os seguintes tipos de dados não funcionam conforme o esperado no servidor remoto do Babelfish: `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, `binary(max)` e `time`. Recomendamos usar a função CAST para convertê-los nos tipos de dados compatíveis.

Exemplo

No exemplo a seguir, uma instância do Babelfish para Aurora PostgreSQL está se conectando a uma instância do RDS para SQL Server na nuvem.

```
EXEC master.dbo.sp_addlinkedserver @server=N'rds_sqlserver', @srvproduct=N'',
 @provider=N'SQLNCLI', @datasrc=N'myserver.CB2XKFSFFMY7.US-WEST-2.RDS.AMAZONAWS.COM';
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'rds_sqlserver',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassw
```

Quando o servidor vinculado estiver instalado, você poderá usar T-SQL OPENQUERY() ou a nomenclatura padrão de quatro partes para fazer referência a uma tabela, exibição ou outros objetos compatíveis no servidor remoto:

```
SELECT * FROM OPENQUERY(rds_sqlserver, 'SELECT * FROM TestDB.dbo.t1');
SELECT * FROM rds_sqlserver.TestDB.dbo.t1;
```

Para descartar o servidor vinculado e todos os logins associados:

```
EXEC master.dbo.sp_dropserver @server=N'rds_sqlserver', @droplogins=N'droplogins';
```

Solução de problemas

Você pode usar o mesmo grupo de segurança para os servidores de origem e remotos para permitir que eles se comuniquem entre si. O grupo de segurança deve permitir somente tráfego de entrada

na porta TDS (1433 por padrão) e o IP de origem no grupo de segurança pode ser definido como o próprio ID do grupo de segurança. Para obter mais informações sobre como definir as regras para se conectar a uma instância de outra instância com o mesmo grupo de segurança, consulte Regras para se conectar a instâncias de uma instância com o mesmo grupo de segurança.

Se o acesso não estiver configurado corretamente, uma mensagem de erro semelhante ao exemplo a seguir será exibida quando você tentar consultar o servidor remoto.

```
TDS client library error: DB #: 20009, DB Msg: Unable to connect: server is unavailable
 or does not exist (mssql2019.aws-region.rds.amazonaws.com), OS #: 110, OS Msg:
 Connection timed out, Level: 9
```

Usar pesquisa de texto completo no Babelfish

A partir da versão 4.0.0, o Babelfish fornece suporte limitado para Pesquisa de texto completo (FTS). FTS é um recurso avançado em bancos de dados relacionais que permite a pesquisa e a indexação eficientes de dados com muito texto. Ela permite que os usuários realizem pesquisas de texto complexas e recuperem resultados relevantes rapidamente. A FTS é particularmente valiosa em aplicações que lidam com grandes volumes de dados textuais, como sistemas de gerenciamento de conteúdo, plataformas de comércio eletrônico e arquivos de documentos.

Noções básicas sobre pesquisa de texto completo do Babelfish

O Babelfish comporta os seguintes recursos de pesquisa de texto completo:

- Cláusula CONTAINS:
  - Suporte básico para a cláusula CONTAINS.

```
CONTAINS (
     {
        column_name
     }
     , '<contains_search_condition>'
)
```

> **Note**
>
> Atualmente, somente o idioma inglês é compatível.

  - Manipulação e tradução abrangentes de strings de pesquisa de `simple_term`.
- Cláusula `FULLTEXT INDEX`:
  - Aceita apenas a declaração `CREATE FULLTEXT INDEX ON table_name(column_name [...n]) KEY INDEX index_name`.
  - Aceita a declaração `DROP FULLTEXT INDEX` completa.

> **Note**
>
> Para reindexar o Índice de texto completo, você precisa descartá-lo e criar outro na mesma coluna.

- Caracteres especiais na condição de pesquisa:
  - O Babelfish garante que ocorrências únicas de caracteres especiais em strings de pesquisa sejam tratadas de forma eficaz.

> **Note**
>
> Embora o Babelfish agora identifique caracteres especiais na string de pesquisa, é essencial reconhecer que os resultados podem variar em comparação com os gerados com o T-SQL.

- Alias da tabela em column_name:
  - Com o suporte a alias de tabela, os usuários podem criar consultas SQL mais concisas e legíveis para a pesquisa de texto completo.

Limitações da pesquisa de texto completo no Babelfish

- Atualmente, as seguintes opções não são compatíveis com o Babelfish para a cláusula `CONTAINS`.
  - Caracteres especiais e idiomas diferentes do inglês não são compatíveis. Você receberá a mensagem de erro genérica para caracteres e idiomas não compatíveis

```
Full-text search conditions with special characters or languages other than English
 are not currently supported in Babelfish
```

  - Várias colunas, como `column_list`

  - Atributo PROPERTY
  - `prefix_term`, `generation_term`, `generic_proximity_term`, `custom_proximity_term`, e `weighted_term`
  - Como não há suporte para operadores boolianos, você receberá a seguinte mensagem de erro quando usá-los:

    ```
    boolean operators not supported
    ```

  - Não há suporte para nomes de identificadores com pontos.
- Atualmente, as seguintes opções não são compatíveis com o Babelfish para a cláusula `CREATE FULLTEXT INDEX`.
  - [TYPE COLUMN type_column_name]
  - [LANGUAGE language_term]
  - [STATISTICAL_SEMANTICS]
  - opções de grupos de arquivos do catálogo
  - com opções
- Não há compatibilidade com a criação de um catálogo de texto completo. A criação de um índice de texto completo não exige um catálogo de texto completo.
- `CREATE FULLTEXT INDEX` não aceita nomes de identificadores com pontos.
- No momento, o Babelfish não comporta caracteres especiais consecutivos nas strings de pesquisa. A seguinte mensagem de erro será exibida quando eles forem usados:

  ```
  Consecutive special characters in the full-text search condition are not currently
   supported in Babelfish
  ```

O Babelfish comporta tipos de dados geoespaciais

A partir das versões 3.5.0 e 4.1.0, o Babelfish inclui suporte para estes dois tipos de dados espaciais:

- Tipo de dados de geometria: esse tipo de dados é destinado ao armazenamento de dados planares ou euclidianos (Terra plana).
- Tipo de dados geográficos: esse tipo de dados é destinado ao armazenamento de dados elipsoidais (Terra redonda), como coordenadas de latitude e longitude do GPS.

Esses tipos de dados permitem o armazenamento e a manipulação de dados espaciais, mas com limitações.

Noções básicas sobre os tipos de dados geoespaciais no Babelfish

- Os tipos de dados geoespaciais são aceitos em vários objetos de banco de dados, como visualizações, procedimentos e tabelas.
- Comporta o tipo de dados de pontos para armazenar dados de localização como pontos definidos por latitude, longitude e um identificador de sistema de referência espacial (SRID) válido. Um ponto pode conter valores Z (elevação), M (medida) e pode estar vazio.
- Comporta o tipo de dados de cadeia de linhas (da versão 5.4.0) definido por uma sequência de pontos e os segmentos de linha que os conectam e por um identificador de sistema de referência espacial (SRID) válido. Uma string de linha pode conter pontos com valores Z (elevação), M (medida) e pode estar vazia.
- Aplicações que se conectam ao Babelfish por meio de drivers, como JDBC, ODBC, DOTNET e PYTHON, podem utilizar esse recurso geoespacial.

Funções de tipo de dados de geometria aceitas no Babelfish

- STGeomFromText (***geometry_tagged_text***, SRID): cria uma instância de geometria usando a representação de texto conhecido (WKT).
- STPointFromText (***point_tagged_text***, SRID): cria uma instância de ponto usando a representação WKT.
- Ponto (X, Y, SRID): cria uma instância de ponto usando valores flutuantes das coordenadas x e y.
- <geometry_instance>.STAsText ( ): extrai a representação WKT da instância de geometria.
- <geometry_instance>.STAsBinary ( ): extrai a representação WKB da instância de geometria.
- <geometry_instance>.STArea ( ): calcula a área total da superfície da instância de geometria.
- <geometry_instance>.STSrid ( ): extrai o identificador de referência espacial (SRID) da instância de geometria.
- <geometry_instance>.STDimension ( ): recupera a dimensão espacial da instância de geometria.
- <geometry_instance>.STIsEmpty ( ): confere se a instância da geometria está vazia.
- <geometry_instance>.STIsClosed ( ): confere se a instância da geometria está vazia.
- <geometry_instance>.STIsValid ( ): confere se a instância da geometria está vazia.

- <geometry_instance>.STDistance (other_geometry): calcula a distância entre duas instâncias de geometria.
- <geometry_instance>.STEquals (other_geometry): confere se a instância de geometria representa o mesmo conjunto de pontos que outra instância de geometria.
- <geometry_instance>.STContains (other_geometry): confere se a instância de geometria contém a instância other_geometry.
- <geometry_instance>.STDisjoint (other_geometry): confere se duas instâncias de geometria não têm pontos em comum.
- <geometry_instance>.STIntersects (other_geometry): confere se há intersecção espacial entre duas instâncias de geometria.
- <geometry_instance>.STX: extrai a coordenada X (longitude) para a instância de geometria.
- <geometry_instance>.STY: extrai a coordenada Y (latitude) para a instância de geometria.

A partir das versões 4.7.0 e 5.3.0, o Babelfish inclui suporte para as seguintes funções de dados espaciais:

- <geometry_instance>.M: extrai a coordenada M da instância de geometria.
- <geometry_instance>.Z: extrai a coordenada Z da instância de geometria.
- <geometry_instance>.HasM: confere se a instância de geometria tem pelo menos um valor M.
- <geometry_instance>.HasZ: confere se a instância de geometria tem pelo menos um valor Z.

A partir das versões 5.4.0, o Babelfish inclui suporte para a seguinte função de dados espaciais:

- STLineFromText (***linestring_tagged_text***, SRID): cria uma instância de string de linha usando a representação WKT.

Funções de tipo de dados de geografia aceitas no Babelfish

- STGeomFromText (***geography_tagged_text***, SRID): cria uma instância de geografia usando a representação WKT.
- STPointFromText (***point_tagged_text***, SRID): cria uma instância de ponto usando a representação WKT.
- Ponto (Lat, Long, SRID): cria uma instância de ponto usando valores flutuantes de latitude e de longitude.

- <geography_instance>.STAsText ( ): extrai a representação WKT da instância de geografia.
- <geography_instance>.STAsBinary ( ): extrai a representação WKB da instância de geografia.
- <geography_instance>.STArea ( ): calcula a área total da superfície da instância de geografia.
- <geography_instance>.STSrid ( ): extrai o identificador de referência espacial (SRID) da instância de geografia.
- <geography_instance>.STDimension ( ): recupera a dimensão espacial da instância de geografia.
- <geography_instance>.STIsEmpty ( ): confere se a instância da geografia está vazia.
- <geography_instance>.STIsClosed ( ): confere se a instância da geografia está fechada.
- <geography_instance>.STIsValid( ): confere se a instância da geografia é válida.
- <geography_instance>.STDistance (other_geography): calcula a distância entre duas instâncias de geografia.
- <geography_instance>.STEquals (other_geography): confere se a instância de geografia representa o mesmo conjunto de pontos que outra instância de geografia.
- <geography_instance>.STContains (other_geography): confere se a instância de geografia contém a instância other_geography.
- <geography_instance>.STDisjoint (other_geography): confere se duas instâncias de geografia não têm pontos em comum.
- <geography_instance>.STIntersects (other_geography): confere se há intersecção espacial entre duas instâncias de geografia.
- <geography_instance>.Lat: extrai o valor de latitude para a instância de geografia.
- <geography_instance>.Long: extrai o valor de longitude para a instância de geografia.

A partir das versões 4.7.0 e 5.3.0, o Babelfish inclui suporte para as seguintes funções de dados espaciais:

- <geography_instance>.M: extrai a coordenada M da instância geográfica.
- <geography_instance>.M: extrai a coordenada M da instância de geografia.
- <geography_instance>.HasM: confere se a instância de geografia tem pelo menos um valor M.
- <geography_instance>.HasZ: confere se a instância de geografia tem pelo menos um valor Z.

A partir das versões 5.4.0, o Babelfish inclui suporte para a seguinte função de dados espaciais:

- STLineFromText (***linestring_tagged_text***, SRID): cria uma instância de string de linha usando a representação WKT.

Limitações no Babelfish para tipos de dados geoespaciais

- No momento, os tipos de geometria e geografia diferentes de instâncias de pontos e string de linha não são aceitos:
  - CircularString
  - CompoundCurve
  - Polígono
  - CurvePolygon
  - MultiPoint
  - MultiLineString
  - MultiPolygon
  - GeometryCollection
- No momento, a indexação espacial não comporta tipos de dados geoespaciais.
- No momento, somente as funções listadas comportam esses tipos de dados. Para obter mais informações, consulte Funções de tipo de dados de geometria aceitas no Babelfish e Funções de tipo de dados de geografia aceitas no Babelfish.
- A saída da função STDistance para dados de geografia pode ter pequenas variações de precisão em comparação com o T-SQL. Isso se deve à implementação subjacente do PostGIS. Para ter mais informações, consulte ST_Distance.
- A saída da função STIsValid para dados de geometria e geografia pode ter alguns desvios em comparação ao T-SQL. Por esse motivo, as funções STDistance, STContains, STInstersects, STDisjoint, STDimension, STArea, STEquals também podem se desviar do T-SQL em alguns casos (exibe a saída em vez de gerar erro). Isso se deve à implementação subjacente do PostGIS. Para ter mais informações, consulte ST_IsValid.
- Para ter uma performance ideal, use tipos de dados geoespaciais integrados, sem criar camadas adicionais de abstração no Babelfish.
- No Babelfish, os nomes das funções geoespaciais são usados como palavras-chave e realizarão operações espaciais somente se utilizados da maneira pretendida.

Noções básicas do particionamento no Babelfish

A partir da versão 4.3.0, o Babelfish introduz o particionamento de tabelas e índices com suporte limitado. As seções a seguir fornecem detalhes sobre como criar funções de partição, definir esquemas de partição e implementar tabelas e índices particionados no Babelfish.

Tópicos

- Introdução ao particionamento no Babelfish
- Limitações e soluções alternativas

Introdução ao particionamento no Babelfish

- Funções de partição:
  - `CREATE PARTITION FUNCTION:` define como uma tabela ou um índice é particionado especificando o tipo de coluna de particionamento e o intervalo de valores de cada partição.
  - `DROP PARTITION FUNCTION:` remove uma função de partição existente.
- Esquemas de partição:
  - `CREATE PARTITION SCHEME:` define o mapeamento entre partições e grupos de arquivos.

    > **Note**
    >
    > No Babelfish, os grupos de arquivos são tratados como objetos fictícios e não representam locais de armazenamento físico.

  - `DROP PARTITION SCHEME:` remove um esquema de partição existente.
- Função do sistema:
  - `$PARTITION:` essa função do sistema exibe o número da partição à qual um valor especificado em uma coluna de particionamento pertenceria em uma tabela particionada especificada.
- Tabelas e índices particionados:
  - `CREATE TABLE ... ON partition_scheme_name (partition_column_name):` cria uma tabela particionada com base em um esquema de particionamento e uma coluna de particionamento especificados.
  - `CREATE INDEX ... ON partition_scheme_name (partition_column_name):` cria um índice particionado com base em um esquema de particionamento e uma coluna de particionamento especificados.

- Visualizações do sistema para particionar metadados:

  As seguintes visualizações do sistema foram adicionadas para fornecer metadados relacionados ao particionamento:

  - `sys.destination_data_spaces`
  - `sys.partitions`
  - `sys.partition_functions`
  - `sys.partition_parameters`
  - `sys.partition_range_values`
  - `sys.partition_schemes`

Limitações e soluções alternativas

Os seguintes recursos de particionamento do SQL Server ainda não são aceitos no Babelfish:

- `ALTER PARTITION FUNCTION` e `ALTER PARTITION SCHEME`.

  > **Note**
  >
  > O Babelfish não comporta operações de divisão e de mesclagem. Defina todas as partições nas funções de partição durante a criação, pois não é possível adicionar nem remover partições posteriormente.

- Colunas computadas como colunas de particionamento.
- Utilitários `INSERT BULK` e `BCP` para tabelas particionadas.
- Opção de limite `LEFT` para funções de partição.
- Tipo de dados `SQL_VARIANT` para funções de partição.
- `TRUNCATE TABLE ... WITH PARTITION`.
- `ALTER TABLE ... SWITCH PARTITION`.
- Índices particionados não alinhados, como esquema de partição e coluna de partição que diferem da tabela particionada.
- A migração do DMS da origem do Babelfish é possível somente com tarefas de carga total em tabelas particionadas.
- Uso de agrupamento na função de partição.

- Uso de uma coluna de particionamento com um agrupamento diferente do agrupamento padrão do banco de dados.
- O Babelfish não aceita essas opções de sintaxe, mas fornece soluções alternativas:
  - Uso do esquema de partição com restrições ou índices na instrução CREATE TABLE.
  - ALTER TABLE ... ADD CONSTRAINT ... ON partition_scheme_name (partition_column_name).

> **Note**
>
> Você pode configurar o hatch de escape `babelfishpg_tsql.escape_hatch_storage_on_partition` para ser ignorado. Isso permitirá que o analisador ignore a opção de esquema de partição usada com restrições ou índices, e o backend criará restrições ou índices individuais para cada partição.

Solução de problemas do Babelfish

A seguir, é possível encontrar ideias de solução de problemas e soluções alternativas para alguns problemas de cluster de banco de dados do Babelfish.

Tópicos

- Falha de conexão

Falha de conexão

As causas comuns de falhas de conexão a um novo cluster de bancos de dados Aurora com Babelfish são as seguintes:

- O grupo de segurança não permite acesso: se você estiver com problemas para se conectar a um Babelfish, certifique-se de ter adicionado seu endereço IP ao grupo de segurança padrão do Amazon EC2. É possível utilizar https://checkip.amazonaws.com/ para determinar seu endereço IP e depois adicioná-lo à regra de entrada para a porta do TDS e a porta do PostgreSQL. Para obter mais informações, consulte Adicionar regras a um grupo de segurança, no Guia do usuário do Amazon EC2.
- Configurações incorretas do SSL: se o parâmetro `rds.force_ssl` estiver ativado (definido como 1) no Aurora PostgreSQL, os clientes deverão se conectar ao Babelfish por SSL. Se o cliente não estiver configurado corretamente, você verá uma mensagem de erro como a seguir:

  ```
  Cannot connect to your-Babelfish-DB-cluster, 1433
  ---------------------
  ADDITIONAL INFORMATION:
  no pg_hba_conf entry for host "256.256.256.256", user "your-user-name",
  "database babelfish_db", SSL off (Microsoft SQL Server, Error: 33557097)
  ...
  ```

  Esse erro indica um possível problema de configuração do SSL entre o cliente local e o cluster de banco de dados do Babelfish, e que o cluster exige que os clientes usem SSL (o parâmetro `rds.force_ssl` é definido como 1). Para obter mais informações sobre como configurar o SSL, consulte Usar o SSL com uma instância de banco de dados do PostgreSQL no Guia do usuário do Amazon RDS.

  Se estiver usando o SQL Server Management Studio (SSMS) para se conectar ao Babelfish e receber esse erro, você poderá escolher as opções de conexão Encrypt connection (Criptografar

conexão) e Trust server certificate (Confiar no certificado do servidor), no painel Properties (Propriedades) da conexão, e tentar novamente. Essas configurações lidam com o requisito de conexão SSL para SSMS.

Para obter mais informações sobre como solucionar problemas de conexão com o Aurora, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS.

Desativar o Babelfish

Quando você não precisar mais do Babelfish, pode desativar sua funcionalidade.

Esteja ciente de algumas considerações:

- Em alguns casos, é possível desativar o Babelfish antes de concluir uma migração para o Aurora PostgreSQL. Se você fizer isso e sua DDL depender dos tipos de dados do SQL Server ou se você utilizar qualquer funcionalidade T-SQL no seu código, este falhará.
- Se você desativar a extensão Babelfish após o provisionamento de uma instância do Babelfish, não poderá provisionar esse mesmo banco de dados novamente no mesmo cluster.

Para desativar o Babelfish, modifique seu grupo de parâmetros definindo `rds.babelfish_status` como `OFF`. É possível continuar utilizando tipos de dados do SQL Server com o Babelfish desativado, definindo `rds.babelfish_status` como `datatypesonly`.

Se você desativar o Babelfish no grupo de parâmetros, todos os clusters que utilizam esse grupo de parâmetros perderão a funcionalidade do Babelfish.

Para obter mais informações sobre como modificar grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora. Para obter informações sobre os parâmetros específicos do Babelfish, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish.

Gerenciar as atualizações da versão do Babelfish para Aurora PostgreSQL

O Babelfish é uma opção disponível com Aurora PostgreSQL versão 13.4 e versões posteriores. As atualizações do Babelfish ficam disponíveis com determinadas novas versões do mecanismo de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Notas de lançamento do Aurora PostgreSQL.

> **Note**
>
> Clusters de banco de dados do Babelfish em qualquer versão do Aurora PostgreSQL 13 não podem ser atualizados para o Aurora PostgreSQL 14.3, 14.4 e 14.5. Além disso, o Babelfish não é compatível com uma atualização direta de 13.x para 15.x. Você deve primeiro atualizar seu cluster de banco de dados 13.x para a versão 14.6 e posterior e depois atualizar para a versão 15.x.

Para obter uma lista de funcionalidades compatíveis em diferentes versões do Babelfish, consulte Funcionalidades compatíveis no Babelfish por versão.

Para ver uma lista de funcionalidades atualmente incompatíveis, consulte Funcionalidades não compatíveis com o Babelfish.

É possível obter uma lista de versões do Aurora PostgreSQL que comportam o Babelfish consultando a Região da AWS com o comando describe-db-engine-versions da AWS CLI da forma a seguir.

Para Linux, macOS ou Unix:

```
$ aws rds describe-db-engine-versions --region us-east-1 \
    --engine aurora-postgresql \
    --query '*[]|[?SupportsBabelfish==`true`].[EngineVersion]' \
    --output text
```

Para obter mais informações, consulte describe-db-engine-versions na Referência de comandos da AWS CLI.

Nos tópicos a seguir, você pode aprender a identificar a versão do Babelfish em execução no cluster de banco de dados do Aurora PostgreSQL e atualizar para uma nova versão.

Sumário

- Identificar a versão do Babelfish
- Atualizar o cluster do Babelfish para uma nova versão principal
  - Atualizar o Babelfish para uma nova versão secundária
  - Atualizar o Babelfish para uma nova versão principal
    - Antes de atualizar o Babelfish para uma nova versão principal
    - Realizar a atualização da versão principal
    - Depois de atualizar para uma nova versão principal
    - Exemplo: Atualizar o cluster de banco de dados do Babelfish para uma versão principal
- Usar o parâmetro de versão do produto Babelfish
  - Configurar o parâmetro de versão do produto Babelfish
  - Consultas e parâmetros afetados
  - Interface com o parâmetro babelfishpg_tsql.version

Identificar a versão do Babelfish

Você pode consultar o Babelfish para localizar detalhes sobre a versão do Babelfish, a versão do Aurora PostgreSQL e a versão compatível do Microsoft SQL Server. Você pode usar a porta do TDS ou a porta do PostgreSQL.

- To use the TDS port to query for version information
- To use the PostgreSQL port to query for version information

Como usar a porta do TDS para consultar informações sobre a versão

1. Use o `sqlcmd` ou o `ssms` para se conectar ao endpoint do cluster de banco de dados do Babelfish.

   ```
   sqlcmd -S bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
       login-id -P password -d db_name
   ```

2. Para identificar a versão do Babelfish, execute a seguinte consulta:

   ```
   1> SELECT CAST(serverproperty('babelfishversion') AS VARCHAR)
   2> GO
   ```

A consulta retorna resultados semelhantes aos seguintes:

```
serverproperty
------------------------------
3.4.0

(1 rows affected)
```

3. Para identificar a versão do cluster de bancos de dados Aurora PostgreSQL, execute a seguinte consulta:

```
1> SELECT aurora_version() AS aurora_version
2> GO
```

A consulta retorna resultados semelhantes aos seguintes:

```
aurora_version


-----------------------------------------------
15.5.0

(1 rows affected)
```

4. Para identificar a versão compatível do Microsoft SQL Server, execute a seguinte consulta:

```
1> SELECT @@VERSION AS version
2> GO
```

A consulta retorna resultados semelhantes aos seguintes:

```
Babelfish for Aurora PostgreSQL with SQL Server Compatibility - 12.0.2000.8
Dec 7 2023 09:43:06
Copyright (c) Amazon Web Services
PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)

(1 rows affected)
```

Como um exemplo que mostra uma pequena diferença entre o Babelfish e o Microsoft SQL Server, você pode executar a consulta a seguir. No Babelfish, a consulta retorna 1, enquanto no Microsoft SQL Server, a consulta retorna NULL.

```
SELECT CAST(serverproperty('babelfish') AS VARCHAR) AS runs_on_babelfish
```

Você também pode usar a porta do PostgreSQL para obter informações sobre a versão, conforme mostrado no procedimento a seguir.

Como usar a porta do PostgreSQL para consultar informações sobre a versão

1. Use o `psql` ou o `pgAdmin` para se conectar ao endpoint do cluster de banco de dados do Babelfish.

   ```
   psql host=bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com
        port=5432 dbname=babelfish_db user=sa
   ```

2. Ative o recurso estendido (`\x`) de `psql` para obter uma saída mais legível.

   ```
   babelfish_db=> \x
   babelfish_db=> SELECT
   babelfish_db=> aurora_version() AS aurora_version,
   babelfish_db=> version() AS postgresql_version,
   babelfish_db=> sys.version() AS Babelfish_compatibility,
   babelfish_db=> sys.SERVERPROPERTY('BabelfishVersion') AS Babelfish_Version;
   ```

   A consulta retorna uma saída similar à seguinte:

   ```
   -[ RECORD 1 ]-----------
   +-------------------------------------------------------------------------------------------
   aurora_version          | 15.5.0
   postgresql_version      | PostgreSQL 15.5 on x86_64-pc-linux-gnu, compiled by
    x86_64-pc-linux-gnu-gcc (GCC) 9.5.0, 64-bit
   babelfish_compatibility | Babelfish for Aurora Postgres with SQL Server
    Compatibility - 12.0.2000.8                    +
                           | Dec 7 2023 09:43:06
                                              +
                           | Copyright (c) Amazon Web Services
                                               +
                           | PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)
   ```

```
babelfish_version       | 3.4.0
```

Atualizar o cluster do Babelfish para uma nova versão principal

Novas versões do Babelfish ficam disponíveis com alguns novos lançamentos do mecanismo de banco de dados Aurora PostgreSQL após a versão 13.4. Cada nova versão do Babelfish tem um número exclusivo. Assim como no Aurora PostgreSQL, o Babelfish usa o esquema de nomenclatura *major*.*minor*.*patch* para versões. Por exemplo, a primeira versão do Babelfish, a 1.0.0, foi disponibilizada como parte do Aurora PostgreSQL 13.4.0.

O Babelfish não requer um processo de instalação separado. Conforme discutido em Criar um cluster de banco de dados do Babelfish para Aurora PostgreSQL, Turn on Babelfish (Ativar o Babelfish) é uma opção que você escolhe ao criar um cluster de banco de dados do Aurora PostgreSQL.

Da mesma forma, você não pode atualizar o Babelfish independentemente do cluster de banco de dados do Aurora auxiliar. Para atualizar um cluster de banco de dados do Babelfish para Aurora PostgreSQL para uma nova versão do Babelfish, atualize o cluster de banco de dados do Aurora PostgreSQL para uma nova versão compatível com a versão do Babelfish que você deseja usar. O procedimento para atualização depende da versão do Aurora PostgreSQL compatível com sua implantação do Babelfish, conforme mostrado a seguir.

Atualizações de versão principal

Você deve atualizar as versões a seguir do Aurora PostgreSQL para o Aurora PostgreSQL 14.6 e posterior antes de atualizar para a versão 15.2 do Aurora PostgreSQL.

- Aurora PostgreSQL 13.8 e todas as versões posteriores
- Aurora PostgreSQL 13.7.1 e todas as versões secundárias posteriores
- Aurora PostgreSQL 13.6.4 e todas as versões secundárias posteriores

Você pode atualizar o Aurora PostgreSQL 14.6 e versões posteriores para o Aurora PostgreSQL 15.2 e versões posteriores.

A atualização de um cluster de banco de dados do Aurora PostgreSQL para uma nova versão principal envolve várias tarefas preliminares. Para ter mais informações, consulte Realizar uma atualização da versão principal. Para atualizar o cluster de banco de dados do Babelfish para Aurora PostgreSQL, você precisa criar um grupo de parâmetros de cluster de banco de dados personalizado para a nova versão do Aurora PostgreSQL. Esse novo grupo de

parâmetros deve conter as mesmas configurações de parâmetros do Babelfish do cluster que você está atualizando. Para ter mais informações e conferir uma tabela de fontes e destinos para atualização da versão principal, consulte Atualizar o Babelfish para uma nova versão principal.

Patches e atualizações de versões secundárias

Versões secundárias e patches não exigem a criação de um grupo de parâmetros de cluster de banco de dados para a atualização. Versões secundárias e patches podem usar o processo de atualização de versão secundária, que pode ser aplicado manual ou automaticamente. Para ter mais informações e conferir uma tabela de origens e destinos de versão, consulte Atualizar o Babelfish para uma nova versão secundária.

> **Note**
>
> Antes de realizar uma atualização principal ou secundária, aplique todas as tarefas de manutenção pendentes ao seu cluster do Babelfish para Aurora PostgreSQL.

Tópicos

- Atualizar o Babelfish para uma nova versão secundária
- Atualizar o Babelfish para uma nova versão principal

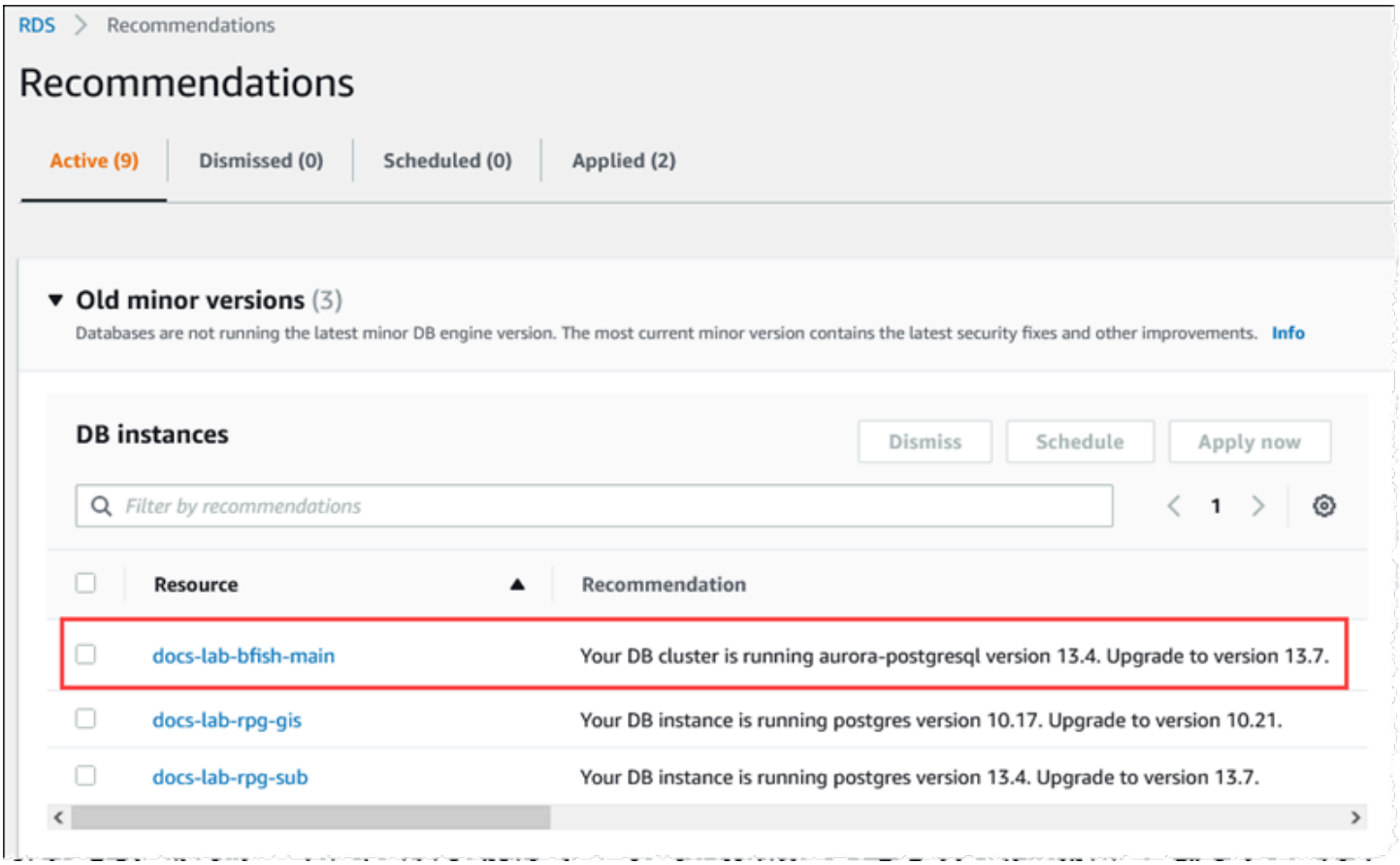
Atualizar o Babelfish para uma nova versão secundária

A atualização de versão secundária visa manter a compatibilidade com versões anteriores. No entanto, em alguns casos, correções essenciais de segurança ou correções de bugs importantes podem não ser totalmente compatíveis com versões anteriores. Uma versão de patch inclui correções importantes para a uma versão secundária após o lançamento. Por exemplo, o rótulo da primeira versão do Aurora PostgreSQL 13.4 foi Aurora PostgreSQL 13.4.0. Vários patches para essa versão secundária já foram lançados até o momento, incluindo Aurora PostgreSQL 13.4.1, 13.4.2 e 13.4.4. Você pode encontrar os patches disponíveis para cada versão do Aurora PostgreSQL na lista de Patch releases (Lançamentos de patches) na parte superior das notas de versão do Aurora PostgreSQL da versão específica. Para obter um exemplo, consulte PostgreSQL 14.3 em Notas de versão do Aurora PostgreSQL.

Se o cluster de banco de dados do Aurora PostgreSQL estiver configurado com a opção Auto minor version upgrade (Atualização automática de versão secundária), o cluster de banco de dados do Babelfish para Aurora PostgreSQL será atualizado automaticamente durante a respectiva janela de

manutenção. Para saber mais sobre a atualização automática de versão secundária (AmVU) e como usá-la, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora. Se o cluster não estiver usando AmVU, você poderá atualizar manualmente o cluster de banco de dados do Babelfish para Aurora PostgreSQL para novas versões secundárias respondendo a tarefas de manutenção ou modificando o cluster para usar a nova versão.

Quando você escolhe uma versão do Aurora PostgreSQL para instalar e quando visualiza um cluster de banco de dados do Aurora PostgreSQL existente no Console de gerenciamento da AWS, a versão exibe apenas os dígitos *major.minor*. Por exemplo, a imagem a seguir do console de um cluster de banco de dados do Babelfish para Aurora PostgreSQL com Aurora PostgreSQL 13.4 recomenda atualizar o cluster para a versão 13.7, uma nova versão secundária do Aurora PostgreSQL.



Para obter os detalhes completos da versão, incluindo o nível de *patch*, você pode consultar o cluster de banco de dados do Aurora PostgreSQL usando a função `aurora_version` do Aurora PostgreSQL. Para ter mais informações, consulte aurora_version no Referência de funções do Aurora PostgreSQL. Você pode encontrar um exemplo de uso da função no procedimento To use the PostgreSQL port to query for version information em Identificar a versão do Babelfish.

A tabela a seguir mostra as versões do Aurora PostgreSQL e do Babelfish e as versões de destino disponíveis que são compatíveis com o processo de atualização da versão secundária.

Versões de origem atuais	Destinos de atualização mais recentes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
17.4 (5.1)	17.5 (5.2)
16.8 (4.5)	16.9 (4.6)
16.6 (4.4.0)	16.9 (4.6), 16.8 (4.5.0)
16.4 (4.3.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0)
16.3 (4.2.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
16.2 (4.1.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
16.1 (4.0.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.12 (3.9.0)	15.13 (3.10)
15.10 (3.8.0)	15.13 (3.10), 15.12 (3.9.0)
15.8 (3.7.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0)
15.7 (3.6.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
15.6 (3.5.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
15.5 (3.4.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
15.4 (3.3.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)

Versões de origem atuais	Destinos de atualização mais recentes
15.12 (3.9.0), 15.3 (3.2.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
15.2 (3.1.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.17 (2.12.0)	14.18 (2.13.0)
14.15 (2.11.0)	14.18 (2.13.0), 14.17 (2.12.0)
14.13 (2.10.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0)
14.12 (2.9.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0)
14.11 (2.8.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0)
14.10 (2.7.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0)
14.9 (2.6.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0)
14.8 (2.5.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0)
14.7 (2.4.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0)

Versões de origem atuais	Destinos de atualização mais recentes
14.6 (2.3.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0)
14.5 (2.2.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0), 14.6 (2.3.0)
14.3 (2.1.0)	14.18 (2.13.0), 14.6 (2.3.0)
13.8 (1.4.0)	13.9 (1.5.0)
13.7 (1.3.0)	13.9 (1.5.0), 13.8 (1.4.0)

Atualizar o Babelfish para uma nova versão principal

Para uma atualização da versão principal, primeiro é necessário atualizar o cluster de banco de dados do Babelfish para Aurora PostgreSQL para uma versão comporte a atualização da versão principal. Para conseguir isso, aplique atualizações de patch ou atualizações de versão secundária no cluster de banco de dados. Para ter mais informações, consulte Atualizar o Babelfish para uma nova versão secundária.

A tabela a seguir mostra a versão do Aurora PostgreSQL e a versão do Babelfish que são compatíveis com o processo de atualização da versão principal.

Versões de origem atuais	Destinos de atualização mais recentes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
16.9 (4.6.0)	17.5 (5.2.0)
16.8 (4.5.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.6 (4.4.0)	17.5 (5.2.0), 17.4 (5.1.0)

Versões de origem atuais	Destinos de atualização mais recentes
16.4 (4.3.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.3 (4.2.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.2 (4.1.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.1 (4.0.0)	17.5 (5.2.0), 17.4 (5.1.0)
15.13 (3.10)	17.5 (5.2.0) 16.9 (4.6.0)
15.12 (3.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0)
15.10 (3.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0)
15.8 (3.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
15.7 (3.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
15.6 (3.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.5 (3.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)

Versões de origem atuais	Destinos de atualização mais recentes
15.4 (3.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.3 (3.2.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.2 (3.1.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
14.18 (2.13.0)	17.5 (5.2.0) 16.9 (4.6.0) 15.13 (3.10.0)
14.17 (2.12.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0) 15.13 (3.10.0), 15.12 (3.9.0)
14.15 (2.11.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0)
14.13 (2.10.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)

Versões de origem atuais	Destinos de atualização mais recentes
14.12 (2.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
14.11 (2.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
14.10 (2.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
14.9 (2.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)

Versões de origem atuais	Destinos de atualização mais recentes
14.8 (2.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.7 (2.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
14.6 (2.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
13.9 (1.5.0)	14.6 (2.3.0)
13.8 (1.4.0)	14.6 (2.3.0)
13.7 (1.3.0)	14.6 (2.3.0)

Antes de atualizar o Babelfish para uma nova versão principal

Uma atualização pode envolver breves interrupções. Por esse motivo, recomendamos que você execute ou programe atualizações durante sua janela de manutenção ou durante outros períodos de baixa utilização.

Antes de realizar uma atualização da versão principal

1. Identifique a versão do Babelfish do cluster de banco de dados do Aurora PostgreSQL usando os comandos descritos em Identificar a versão do Babelfish. As informações da versão do Aurora PostgreSQL e da versão do Babelfish são tratadas pelo PostgreSQL, portanto siga as etapas detalhadas no procedimento To use the PostgreSQL port to query for version information para obter os detalhes.
2. Verifique se sua versão é compatível com a atualização da versão principal. Para ver a lista de versões compatíveis com o recurso de atualização da versão principal, consulte Atualizar o Babelfish para uma nova versão secundária e execute as tarefas de pré-atualização necessárias.

   Por exemplo, se sua versão do Babelfish estiver sendo executada em um cluster de banco de dados do Aurora PostgreSQL 13.5 e você quiser atualizar para o Aurora PostgreSQL 15.2, primeiro aplique todas as versões secundárias e patches para atualizar o cluster para o Aurora PostgreSQL 14.6 ou versão posterior. Quando o cluster estiver na versão 14.6 ou posterior, continue com o processo de atualização da versão principal.
3. Crie um snapshot manual do cluster de banco de dados do Babelfish atual como backup. O backup permite restaurar o cluster para a versão do Aurora PostgreSQL e a versão do Babelfish, bem como restaurar todos os dados para o estado anterior à atualização. Para ter mais informações, consulte Criar um snapshot de cluster de banco de dados. Mantenha seu grupo de parâmetros de cluster de banco de dados personalizado para usá-lo novamente se você decidir restaurar esse cluster ao estado antes da atualização. Para obter mais informações, consulte Restauração de um snapshot de um cluster de banco de dados e Considerações de grupos de parâmetros.
4. Prepare um grupo de parâmetros de cluster de banco de dados personalizado para a versão do banco de dados de destino do Aurora PostgreSQL. Duplique as configurações dos parâmetros do Babelfish do cluster de banco de dados do Babelfish para Aurora PostgreSQL. Para conferir uma lista de todos os parâmetros do Babelfish, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish. Para uma atualização da versão principal, os parâmetros a seguir exigem as mesmas configurações do cluster de banco de dados de origem. Para que a atualização seja bem-sucedida, todas as configurações devem ser iguais.

   - rds.babelfish_status
   - babelfishpg_tds.tds_default_numeric_precision
   - babelfishpg_tds.tds_default_numeric_scale

- babelfishpg_tsql.database_name
- babelfishpg_tsql.default_locale
- babelfishpg_tsql.migration_mode
- babelfishpg_tsql.server_collation_name

> **Warning**
>
> Se as configurações dos parâmetros do Babelfish no grupo de parâmetros do cluster de banco de dados personalizado para a nova versão do Aurora PostgreSQL não corresponderem aos valores dos parâmetros do cluster que você está atualizando, a operação `ModifyDBCluster` falhará. Uma mensagem de erro `InvalidParameterCombination` aparecerá no Console de gerenciamento da AWS ou na saída do comando `modify-db-cluster` da AWS CLI.

5. Use o Console de gerenciamento da AWS ou a AWS CLI para criar o grupo de parâmetros personalizado do cluster de banco de dados. Escolha a família do Aurora PostgreSQL aplicável para a versão do Aurora PostgreSQL que você deseja para a atualização.

> **Tip**
>
> Os grupos de parâmetros são gerenciados no âmbito da Região da AWS. Ao trabalhar com a AWS CLI, você pode configurar com uma região padrão em vez de especificar `--region` no comando. Para saber mais sobre como usar a AWS CLI, consulte Instalação rápida no Guia do usuário da AWS Command Line Interface.

Realizar a atualização da versão principal

1. Atualize o cluster de banco de dados do Aurora PostgreSQL para uma nova versão principal. Para ter mais informações, consulte Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal.
2. Reinicialize a instância gravadora do cluster para que as configurações dos parâmetros entrem em vigor.

Depois de atualizar para uma nova versão principal

Depois de uma atualização da versão principal para uma nova versão do Aurora PostgreSQL, o valor de `IDENTITY` nas tabelas com uma coluna `IDENTITY` pode ser maior (+32) do que o valor antes da atualização. O resultado é que, quando a próxima linha é inserida nessas tabelas, o valor da coluna de identidade gerada salta para o número +32 e inicia a sequência a partir daí. Essa condição não afetará negativamente as funções do cluster de banco de dados do Babelfish. No entanto, se quiser, você poderá redefinir o objeto de sequência com base no valor máximo da coluna. Para fazer isso, conecte-se à porta T-SQL da sua instância gravadora do Babelfish usando `sqlcmd` ou outro cliente do SQL Server. Para ter mais informações, consulte Utilizar um cliente SQL Server para se conectar ao seu cluster de banco de dados.

```
sqlcmd -S bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
     sa -P ******** -d dbname
```

Após a conexão, use o comando SQL a seguir para gerar instruções que você pode usar para definir o seed do objeto de sequência associado. Esse comando SQL funciona tanto para configurações de um único banco de dados quanto para vários bancos do Babelfish. Para ter mais informações sobre esses dois modelos de implantação, consulte Utilizar o Babelfish com um único banco de dados ou vários bancos de dados.

```
DECLARE @schema_prefix NVARCHAR(200) = ''
IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id)
    + '.' + tables.name + ''', ''' + columns.name + '''),(select max(' + columns.name +
 ')
    FROM ' + schema_name(tables.schema_id) + '.' + tables.name + '));
    'FROM sys.tables tables JOIN sys.columns
    columns ON tables.object_id = columns.object_id
    WHERE columns.is_identity = 1
GO
```

A consulta gera uma série de instruções SELECT que você pode executar para redefinir o valor máximo de IDENTITY e fechar qualquer lacuna. Veja a seguir a saída ao usar o exemplo de banco de dados do SQL Server, Northwind, em execução em um cluster do Babelfish.

```
-----------------------------------------------------
```

```
SELECT setval(pg_get_serial_sequence('northwind_dbo.categories', 'categoryid'),(select
 max(categoryid)
    FROM dbo.categories));

SELECT setval(pg_get_serial_sequence('northwind_dbo.orders', 'orderid'),(select
 max(orderid)
    FROM dbo.orders));


SELECT setval(pg_get_serial_sequence('northwind_dbo.products', 'productid'),(select
 max(productid)
    FROM dbo.products));


SELECT setval(pg_get_serial_sequence('northwind_dbo.shippers', 'shipperid'),(select
 max(shipperid)
    FROM dbo.shippers));


SELECT setval(pg_get_serial_sequence('northwind_dbo.suppliers', 'supplierid'),(select
 max(supplierid)
    FROM dbo.suppliers));


(5 rows affected)
```

Execute as instruções uma a uma para redefinir os valores da sequência.

Exemplo: Atualizar o cluster de banco de dados do Babelfish para uma versão principal

Neste exemplo, você pode encontrar a série de comandos da AWS CLI que explica como fazer upgrade de um cluster de banco de dados do Aurora PostgreSQL 13.6.4 que executa o Babelfish versão 1.2.2 para o Aurora PostgreSQL 14.6. Antes de começar, crie um grupo de parâmetros de cluster de banco de dados personalizado para o Aurora PostgreSQL 14. Depois, modifique os valores dos parâmetros para que correspondam aos da origem do Aurora PostgreSQL versão 13. Por fim, execute a atualização modificando o cluster de origem. Para ter mais informações, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish. Nesse tópico, você também pode encontrar informações sobre como usar o Console de gerenciamento da AWS para realizar a atualização.

Use o comando create-db-cluster-parameter-group da CLI para criar o grupo de parâmetros do cluster de banco de dados para a nova versão.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
    --db-parameter-group-family aurora-postgresql14 \
    --description 'New custom parameter group for upgrade to new major version' \
    --region us-west-1
```

Quando você emite esse comando, o grupo de parâmetros do cluster de banco de dados personalizado é criado na Região da AWS. Você verá uma saída semelhante à seguinte.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14",
        "DBParameterGroupFamily": "aurora-postgresql14",
        "Description": "New custom parameter group for upgrade to new major version",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-west-1:111122223333:cluster-
pg:docs-lab-babelfish-apg-14"
    }
}
```

Para ter mais informações, consulte Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Use o comando modify-db-cluster-parameter-group da CLI para modificar as configurações de forma que correspondam ao cluster de origem.

Para Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name docs-lab-
babelfish-apg-14 ^
  --parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_precision,ParameterValue=38,ApplyMethod=pen
reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_scale,ParameterValue=8,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.database_name,ParameterValue=babelfish_db,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.default_locale,ParameterValue=en-
US,ApplyMethod=pending-reboot" ^
```

```
 "ParameterName=babelfishpg_tsql.migration_mode,ParameterValue=single-
db,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.server_collation_name,ParameterValue=sql_latin1_general_cp1_ci
reboot"
```

A resposta é semelhante à seguinte.

```
{
    "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14"
}
```

Use o comando modify-db-cluster da CLI para modificar o cluster a fim de usar a nova versão e o novo grupo de parâmetros de cluster de banco de dados personalizado. Você também especifica o argumento `--allow-major-version-upgrade`, conforme exibido no exemplo a seguir.

```
aws rds modify-db-cluster \
--db-cluster-identifier docs-lab-bfish-apg-14 \
--engine-version 14.6 \
--db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
--allow-major-version-upgrade \
--region us-west-1 \
--apply-immediately
```

Use o comando reboot-db-instance da CLI para reinicializar a instância gravadora do cluster para que as configurações dos parâmetros entrem em vigor.

```
aws rds reboot-db-instance \
--db-instance-identifier docs-lab-bfish-apg-14-instance-1\
--region us-west-1
```

Usar o parâmetro de versão do produto Babelfish

Um novo parâmetro de Grand Unified Configuration (GUC) denominado `babelfishpg_tds.product_version` foi introduzido nas versões 2.4.0 e 3.1.0 do Babelfish. Esse parâmetro permite que você defina o número da versão do produto SQL Server como a saída do Babelfish.

O parâmetro é uma string de ID de versão de quatro partes e cada parte deve ser separada por “.”.

Sintaxe

```
Major.Minor.Build.Revision
```

- Versão principal: um número entre 11 e 16.
- Versão principal: um número entre 0 e 255.
- Versão de compilação: um número entre 0 e 65535.
- Revisão: 0 e qualquer número positivo.

Configurar o parâmetro de versão do produto Babelfish

Você deve usar o grupo de parâmetros do cluster para definir o parâmetro babelfishpg_tds.product_version no console. Para ter mais informações sobre como modificar o parâmetro de cluster de banco de dados, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Quando você define o parâmetro da versão do produto como um valor inválido, a alteração não entra em vigor. Embora o console possa mostrar o novo valor, o parâmetro retém o valor anterior. Confira o arquivo de log do mecanismo para ter detalhes sobre as mensagens de erro.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name mydbparametergroup \
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
--db-cluster-parameter-group-name mydbparametergroup ^
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Consultas e parâmetros afetados

Consulta/parâmetro	Resultado	Tempo efetivo
SELECT @@VERSION	Retorna a versão do SQL Server definida pelo usuário (babelfishpg_tsql.version value = Default)	Imediatamente
SELECT SERVERPROPERTY('ProductVersion')	Retorna a versão do SQL Server definida pelo usuário	Imediatamente
SELECT SERVERPROPERTY('ProductMajorVersion')	Retorna a versão principal do SQL Server definida pelo usuário	Imediatamente
Tokens VERSION na mensagem de resposta PRELOGIN	O servidor retorna mensagens PRELOGIN com a versão do SQL Server definida pelo usuário	Entra em vigor quando um usuário cria uma sessão
SQLServerVersion em LoginAck ao usar JDBC	DatabaseMetaData.getDatabaseProductVersion() retorna a versão do SQL Server definida pelo usuário	Entra em vigor quando um usuário cria uma sessão

Interface com o parâmetro babelfishpg_tsql.version

Você pode definir a saída do @@VERSION usando os parâmetros babelfishpg_tsql.version e babelfishpg_tds.product_version. Os exemplos a seguir mostram como esses dois parâmetros se relacionam.

- Quando o parâmetro babelfishpg_tsql.version é “padrão” e babelfishpg_tds.product_version é 15.0.2000.8.
  - Saída de @@version: 15.0.2000.8.
- Quando o parâmetro babelfishpg_tsql.version é definido como 13.0.2000.8 e babelfishpg_tds.product_version é 15.0.2000.8.

- Saída de @@version: 13.0.2000.8.

Referência do Babelfish para Aurora PostgreSQL

Tópicos

- Funcionalidades compatíveis no Babelfish por versão
- Funcionalidades não compatíveis com o Babelfish
- Trabalhar com o Babelfish para Aurora PostgreSQL
- O Babelfish comporta métodos de tipo de dados XML.

Funcionalidades compatíveis no Babelfish por versão

Nas tabelas a seguir, é possível encontrar a lista de todas as funcionalidades compatíveis com diferentes versões do Babelfish. Para ver uma lista de funcionalidades incompatíveis, consulte Funcionalidades não compatíveis com o Babelfish. Para obter informações sobre várias versões do Babelfish, consulte as Notas de lançamento do Aurora PostgreSQL.

Recursos do Aurora e do PostgreSQL

Na tabela a seguir, é possível encontrar os recursos do Aurora e do PostgreSQL aceitos por diferentes versões do Babelfish.

Recursos do Aurora e do PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Serviços de ML do Aurora	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Autenticação de banco de dados com Kerberos usando o Directory Service	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
Despejo e restauração	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Extensão pg_stat_statement																																	

| | | |

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
ALTER FUNCTION	S	S	S	S	S	N	N	N	N	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
ALTER PROCEDURE	S	S	S	S	S	S	S	N	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
ALTER ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
ALTER USER...WITH LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
ALTER VIEW	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Palavra-chave AS em CREATE FUNCTION	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Cláusula AT TIME ZONE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Instância do Babelfish como um servidor vinculado	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	Não
Operações de comparação !< e !>	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não

| Funcionalidade ou sintaxe do T-SQL | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Criar índices exclusivos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Execução de procedimentos entre bancos de dados	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
Referências entre bancos de dados SELECT, INSERT, UPDATE, DELETE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Somente parâmetros digitados por cursor para parâmetros de entrada (não de saída)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Migração de dados usando o utilitário cliente bcp	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Tipos de dados: TIMESTAMP, ROWVERSION (para ter informaçõ es de uso, consulte Recursos com implementação limitada).	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Palavra-chave DEFAULT em chamadas para procedimentos armazenados e funções	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
DBCC CHECKIDENT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
DROP DATABASE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
DROP IF EXISTS (para objetos SCHEMA, DATABASE e USER)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Índice DROP INDEX ON schema.table	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
DROP INDEX schema.table.index	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
DROP PARTITION FUNCTION	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
DROP PARTITION SCHEME	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DROP ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
ENABLE/DISABLE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
FULL TEXT SEARCH	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Pesquisa de texto completo com a cláusula CONTAINS	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Tipos de dados espaciais de geometria e geografia	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
GRANT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Permissão GRANT [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
GUC babelfish pg_tds.product_version	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	Não
Identificadores com caracteres de ponto inicial	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Gatilhos INSTEAD OF em tabelas	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Acionadores INSTEAD OF em visualizações	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
KILL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
$PARTITION.partition_function_name(partition_column_value)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
PIVOT (use na definição de visualização a partir de 4.4.0 e 3.8.0, expressão de tabela comum a partir de 3.4.0, junção a partir de 3.4.0)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
REVOKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Permissão REVOKE [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Cláusulas SELECT... OFFSET...FETCH	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SELECT FOR JSON AUTO	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
SET BABELFISH _SHOWPLAN_ALL ON (e OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SET BABELFISH _STATISTICS PROFILE ON (OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SET CONTEXT_I NFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	Não
SET LOCK_TIME OUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SET NO_BROWSE TABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
SET rowcount	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	Não
SET SHOWPLAN_ ALL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
SET STATISTICS IO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Sintaxe SET TRANSACTION ISOLATION LEVEL	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
SSMS: caixa de diálogo Conectar-se com a conexão do explorador de objetos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SSMS: migração de dados com o assistente de importação/exportação	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SSMS: suporte parcial para o explorador de objetos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
STDEV	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	N	Não

Funcionalidade ou sintaxe do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
STDEVP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	Não
Gatilhos com várias ações DML podem fazer referência tabelas de transição	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
Dicas T-SQL (métodos de junção, uso de índices, MAXDOP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	Não
Sintaxe de colchetes do T-SQL com o predicado LIKE.	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	Não
UNPIVOT	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
Valores de string não delimitados por aspas em chamadas de procedimento armazenado e valores padrão	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
VAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	S	N	N	N	N	N	Não
VARP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	S	N	N	N	N	N	Não

Funções incorporadas do T-SQL

Na tabela a seguir, é possível encontrar as funções incorporadas do T-SQL compatíveis com diferentes versões do Babelfish.

Funções incorporadas do T-SQL	5.5	5.4	4.4.0	4.4.1	4.4.2	4.5.0	4.5.1	4.6.0	4.6.1	3.3.0	3.3.1	3.3.2	3.4.0	3.4.1	3.4.2	3.4.3	3.4.4	3.4.5	3.4.6	3.4.7	2.2.0	2.2.1	2.2.2	2.3.0	2.3.1	2.3.2	2.3.3	2.3.4	2.3.5	2.3.6	2.3.7	2.3.8	2.3.9	2.3.10	1.0
APP_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
ATN2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
CHARINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
CHOOSE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
COL_LENGTH	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
COL_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
COLUMNS_UPDATED	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
COLUMNPROPERTY (somente CharMaxLen, AllowsNull)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
CONCAT_WS																																		

Funções incorporadas do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	Não
CURSOR_STATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
DATABASE_PRINCIPAL_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	Não
DATEADD	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
DATEDIFF	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
DATEDIFF_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	Não
DATEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
DATENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não
DATEPART	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	Não

| | | |

| | | |

| | | |

Funções incorporadas do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
ROWCOUNT_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	Não
SCHEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SESSION_CONTEXT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	Não
SESSION_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SID_BINARY (exibe NULL sempre)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
SMALLDATETIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	S	S	N	N	N	N	N	Não
SQUARE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
STR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	Não
STRING_AGG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	Não
STRING_SPLIT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim

Funções incorporadas do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SUSER_SID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SUSER_SNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
SWITCHOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
SYSTEM_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	Não
TODATETIMEOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
TO_CHAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	Não
TRIGGER_NESTLEVEL (somente sem argumentos)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
TRY_CONVERT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	Não

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

Procedimentos armazenados do sistema do T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_helprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sp_helpsrvrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
sp_helpuser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sp_linkedservers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	Não
sp_oledb_ro_usrname	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sp_pkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sp_prefix	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Não
sp_prepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sp_procedure_params_100_managed	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

Visualizações do SQL Server aceitas pelo Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.dm_os_host_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sys.dm_os_sys_info	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
sys.endpoints	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sys.extended_properties	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	Não
sys.indexes	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sim
sys.partitions	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
sys.partition_functions	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
sys.partition_parameters	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não
sys.partition_range_values	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Não

| | | |

| | | |

- Configurações que não há suporte
- Comandos que não são compatíveis
- Nomes de coluna que não são compatíveis
- Tipos de dados que não há suporte
- Tipos de objeto que não há suporte
- Funções que não há suporte
- Sintaxes que não há suporte

Funcionalidades que não são compatíveis no momento

Na tabela, você pode encontrar informações sobre determinada funcionalidade que não é compatível no momento.

Funcionalidade ou sintaxe	Descrição
Módulos de montagem e rotinas SQL Common Language Runtime (CLR)	Não há suporte para a funcionalidade relacionada a módulos de assembly e rotinas CLR.
Atributos de cookies	Não há suporte para ROWGUIDCOL, SPARSE, FILESTREAM e MASKED.
Bancos de dados contidos	Não há suporte para bancos de dados contidos com logins autenticados em nível de banco de dados, em vez de em nível de servidor.
DDL entre bancos de dados	Ainda não é possível executar instruções DDL que fazem referência ou operam em objetos em vários bancos de dados.
Cursores (atualizáveis)	Não há suporte para cursores atualizáveis.
Cursores (globais)	Não há suporte para cursores GLOBAL.
Cursor (comportamentos de busca)	Não há suporte para os seguintes comportamentos de busca do cursor: FETCH PRIOR, FIRST, LAST, ABSOLUTE e RELATIVE

Funcionalidade ou sintaxe	Descrição
Parâmetros de saída com tipo de cursor	Não há suporte para variáveis e parâmetros com tipo de cursor para parâmetros de saída (geram erro).
Opções do cursor	SCROLL, KEYSET, DYNAMIC, FAST_FORWARD, SCROLL_LOCKS, OPTIMISTIC, TYPE_WARNING e FOR UPDATE
Criptografia de dados	Não há suporte para a criptografia de dados.
Aplicações no nível de dados (DAC)	Não há suporte para operações de importação ou exportação de aplicações no nível de dados (DAC) com arquivos de pacote DAC (.dacpac) ou backup DAC (.bacpac).
Comandos DBCC	Os Database Console Commands (DBCC – Comandos do console de banco de dados) do Microsoft SQL Server não são compatíveis. DBCC CHECKIDENT é aceito no Babelfish 3.4.0 e versões posteriores.
DROP IF EXISTS	Essa sintaxe não tem suporte para objetos USER e SCHEMA. Ela tem suporte para os objetos TABLE, VIEW, PROCEDURE, FUNCTION e DATABASE.
Criptografia	Funções e instruções incorporadas não oferecem suporte para criptografia.
Conexões ENCRYPT_C LIENT_CERT	Não há suporte para conexões de certificados de cliente.
Instrução EXECUTE AS	Não há suporte para essa instrução.
Cláusula EXECUTE AS SELF	Essa cláusula não tem suporte em funções, procedimentos ou acionadores.
Cláusula EXECUTE AS USER	Essa cláusula não tem suporte em funções, procedimentos ou acionadores.

Funcionalidade ou sintaxe	Descrição
Restrições de chave externa que fazem referência ao nome do banco de dados	Não há suporte para restrições de chave externa que fazem referência ao nome do banco de dados.
FORMAT	Tipos definidos pelo usuário não são compatíveis.
Declarações de função com mais de 100 parâmetros	Não há suporte para declarações de função que contêm mais de 100 parâmetros.
Chamadas de função que incluem DEFAULT como valor de parâmetro	DEFAULT não é um valor de parâmetro com suporte para uma chamada de função. DEFAULT como valor de parâmetro para uma chamada de função é aceito no Babelfish 3.4.0 e em versões posteriores.
Funções definidas externamente	Não há suporte para funções externas, incluindo funções SQL CLR.
Tabelas temporárias globais (tabelas com nomes que começam com ##)	Não há suporte para tabelas temporárias globais.
Funcionalidades de gráfico	Não há suporte para todas as funcionalidades de gráfico do SQL.
Procedimentos armazenados estendidos gerais	Procedimentos armazenados do sistema que fornecem uma interface de uma instância do SQL Server para programas externos para várias atividades de manutenção não são compatíveis. Isso inclui `xp_cmdshell` e outros procedimentos armazenados do sistema. Para ter mais informações, consulte Procedimentos armazenados estendidos gerais.
Identificadores (variáveis ou parâmetros) com vários caracteres @ iniciais	Não há suporte para identificadores que começam com mais de um @ inicial.

Funcionalidade ou sintaxe	Descrição
Identificadores, nomes de tabelas ou colunas que contêm caracteres @ ou]]	Não há suporte para nomes de tabelas ou colunas que contêm um sinal @ ou colchetes.
Índices em linha	Não há suporte para índices em linha.
Invocar um procedimento cujo nome esteja em uma variável	Não há suporte ao uso de uma variável como nome de procedimento.
Visualizações materializadas	Não há suporte para visualizações materializadas.
Cláusula FOR REPLICATION	Essa sintaxe é aceita e ignorada.
Funções de escape ODBC	Não há suporte para funções de escape ODBC.
Chamadas de procedimento que incluem DEFAULT como valor de parâmetro	DEFAULT não é um valor de parâmetro com suporte. DEFAULT como valor de parâmetro para uma chamada de função é aceito no Babelfish 3.4.0 e em versões posteriores.
Declarações de procedimentos com mais de 100 parâmetros	Não há suporte para declarações com mais de 100 parâmetros.
Procedimentos definidos externamente	Não há suporte para procedimentos definidos externamente, incluindo SQL CLR.
Versionamento de procedimentos	Não há suporte para versionamento de procedimentos.
Procedimentos WITH RECOMPILE	Não há suporte para WITH RECOMPILE (quando utilizada em conjunto com as instruções DECLARE e EXECUTE).

Funcionalidade ou sintaxe	Descrição
Referências de objetos remotos	Não é possível executar procedimentos armazenados em servidores vinculados ao Babelfish. Os nomes de objetos em quatro partes funcionam somente para leitura e não para modificar a tabela remota. Um UPDATE pode referenciar uma tabela remota na cláusula FROM sem a modificar. Para obter mais informações, consulte O Babelfish é compatível com servidores vinculados.
Segurança por linha	Não há suporte para segurança em nível de linha com CREATE SECURITY POLICY e funções com valor de tabela em linha.
Funcionalidade do agente de serviço	Não há suporte para a funcionalidade do agente de serviço.
SESSIONPROPERTY	Propriedades sem suporte: ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL e NUMERIC_ROUNDABORT
SET LANGUAGE	Não há suporte para essa sintaxe com nenhum valor que não seja `english` ou `us_english` .
SP_CONFIGURE	Não há suporte para esse procedimento armazenado do sistema.
Palavra-chave SQL `SPARSE`	A palavra-chave SPARSE é aceita e ignorada.
Sintaxe do construtor de valores de tabela (cláusula FROM)	A sintaxe sem suporte é para uma tabela derivada construída com a cláusula FROM.
Tabelas temporais	Tabelas temporais não são compatíveis.
Procedimentos temporários não são descartados automaticamente	Não há suporte para essa funcionalidade.

Funcionalidade ou sintaxe	Descrição
Acionadores definidos externamente	Não há suporte para esses acionadores, incluindo SQL Common Language Runtime (CLR).

Configurações que não há suporte

Não há suporte para as configurações:

- SET ANSI_NULL_DFLT_OFF ON
- SET ANSI_NULL_DFLT_ON OFF
- SET ANSI_PADDING OFF
- SET ANSI_WARNINGS OFF
- SET ARITHABORT OFF
- SET ARITHIGNORE ON
- SET CURSOR_CLOSE_ON_COMMIT ON
- SET NUMERIC_ROUNDABORT ON
- SET PARSEONLY ON (o comando não funciona como esperado)
- SET FMTONLY ON (o comando não funciona como esperado. Ele suprime somente a execução de declarações SELECT, mas não de outras.)

Comandos que não são compatíveis

Não há suporte para certas funcionalidades para os seguintes comandos:

- ADD SIGNATURE
- ALTER DATABASE, ALTER DATABASE SET
- BACKUP/RESTORE DATABASE/LOG
- BACPAC e DACPAC FILES RESTORE
- CREATE, ALTER, DROP AUTHORIZATION. ALTER AUTHORIZATION é compatível com objetos de banco de dados.
- CREATE, ALTER, DROP AVAILABILITY GROUP
- CREATE, ALTER, DROP BROKER PRIORITY
- CREATE, ALTER, DROP COLUMN ENCRYPTION KEY

- CREATE, ALTER, DROP DATABASE ENCRYPTION KEY
- CREATE, ALTER, DROP, BACKUP CERTIFICATE
- CREATE AGGREGATE
- CREATE CONTRACT
- CHECKPOINT

Nomes de coluna que não são compatíveis

Não há suporte para os seguintes nomes de coluna:

- $IDENTITY
- $ROWGUID
- IDENTITYCOL

Tipos de dados que não há suporte

Não há suporte para os seguintes tipos de dados:

- HIERARCHYID

Tipos de objeto que não há suporte

Não há suporte para os seguintes tipos de objeto:

- COLUMN MASTER KEY
- CREATE, ALTER EXTERNAL DATA SOURCE
- CREATE, ALTER, DROP DATABASE AUDIT SPECIFICATION
- CREATE, ALTER, DROP EXTERNAL LIBRARY
- CREATE, ALTER, DROP SERVER AUDIT
- CREATE, ALTER, DROP SERVER AUDIT SPECIFICATION
- CREATE, ALTER, DROP, OPEN/CLOSE SYMMETRIC KEY
- CREATE, DROP DEFAULT
- CREDENTIAL
- CRYPTOGRAPHIC PROVIDER
- DIAGNOSTIC SESSION

- Visualizações indexadas
- SERVICE MASTER KEY
- SYNONYM

Funções que não há suporte

As seguintes funções integradas não são compatíveis:

Funções agregadas

- APPROX_COUNT_DISTINCT
- CHECKSUM_AGG
- GROUPING_ID
- STRING_AGG usando a cláusula WITHIN GROUP

Funções criptográficas

- Função CERTITENCODED
- Função CERTID
- Função CERTPROPERTY

Funções de metadados

- COLUMNPROPERTY
- TYPEPROPERTY
- Função SERVERPROPERTY: as seguintes propriedades não são compatíveis:
  - BuildCLRVersion
  - ComparisonStyle
  - ComputerNamePhysicalNetBIOS
  - HadrManagerStatus
  - InstanceDefaultDataPath
  - InstanceDefaultLogPath
  - IsClustered
  - IsHadrEnabled

- LCID
- NumLicenses
- ProcessID
- ProductBuild
- ProductBuildType
- ProductUpdateReference
- ResourceLastUpdateDateTime
- ResourceVersion
- ServerName
- SqlCharSet
- SqlCharSetName
- SqlSortOrder
- SqlSortOrderName
- FilestreamShareName
- FilestreamConfiguredLevel
- FilestreamEffectiveLevel

Funções de segurança

- CERTPRIVATEKEY
- LOGINPROPERTY

Instruções, operadores, outras funções

- Função EVENTDATA
- GET_TRANSMISSION_STATUS
- OPENXML

Sintaxes que não há suporte

Não há suporte para a seguinte sintaxe:

- ALTER DATABASE

- ALTER DATABASE SCOPED CONFIGURATION
- ALTER DATABASE SCOPED CREDENTIAL
- ALTER DATABASE SET HADR
- ALTER INDEX
- ALTER PARTITION FUNCTION
- ALTER PARTITION SCHEME
- ALTER SCHEMA
- ALTER SERVER CONFIGURATION
- Cláusula ALTER SERVICE, BACKUP/RESTORE SERVICE MASTER KEY
- BEGIN CONVERSATION TIMER
- BEGIN DISTRIBUTED TRANSACTION
- BEGIN DIALOG CONVERSATION
- BULK INSERT
- CREATE COLUMNSTORE INDEX
- CREATE EXTERNAL FILE FORMAT
- CREATE EXTERNAL TABLE
- CREATE, ALTER, DROP APPLICATION ROLE
- CREATE, ALTER, DROP ASSEMBLY
- CREATE, ALTER, DROP ASYMMETRIC KEY
- CREATE, ALTER, DROP CREDENTIAL
- CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER
- CREATE, ALTER, DROP ENDPOINT
- CREATE, ALTER, DROP EVENT SESSION
- CREATE, ALTER, DROP EXTERNAL LANGUAGE
- CREATE, ALTER, DROP EXTERNAL RESOURCE POOL
- CREATE, ALTER, DROP FULLTEXT CATALOG
- CREATE, ALTER, DROP FULLTEXT INDEX
- CREATE, ALTER, DROP FULLTEXT STOPLIST
- CREATE, ALTER, DROP MESSAGE TYPE
- CREATE, ALTER, DROP, OPEN/CLOSE, BACKUP/RESTORE MASTER KEY

- CREATE, ALTER, DROP QUEUE
- CREATE, ALTER, DROP RESOURCE GOVERNOR
- CREATE, ALTER, DROP RESOURCE POOL
- CREATE, ALTER, DROP ROUTE
- CREATE, ALTER, DROP SEARCH PROPERTY LIST
- CREATE, ALTER, DROP SECURITY POLICY
- CREATE, ALTER, DROP SELECTIVE XML INDEX clause
- CREATE, ALTER, DROP SERVICE
- CREATE, ALTER, DROP SPATIAL INDEX
- CREATE, ALTER, DROP TYPE
- CREATE, ALTER, DROP XML INDEX
- CREATE, ALTER, DROP XML SCHEMA COLLECTION
- CREATE/DROP RULE
- CREATE, DROP WORKLOAD CLASSIFIER
- CREATE, ALTER, DROP WORKLOAD GROUP
- ALTER TRIGGER
- CREATE TABLE... Cláusula GRANT
- CREATE TABLE... Cláusula IDENTITY
- CREATE USER: essa sintaxe não é compatível. A instrução PostgreSQL CREATE USER não cria um usuário equivalente à sintaxe de CREATE USER do SQL Server. Para obter mais informações, consulte Diferenças do T-SQL no Babelfish.
- DENY
- END, MOVE CONVERSATION
- EXECUTE with AS LOGIN or AT option
- GET CONVERSATION GROUP
- GROUP BY ALL clause
- GROUP BY CUBE clause
- GROUP BY ROLLUP clause
- INSERT... DEFAULT VALUES
- MERGE
- READTEXT

- REVERT
- SELECT TOP... WITH TIES
- SELECT... FOR BROWSE
- SELECT... FOR XML AUTO
- SELECT... FOR XML EXPLICIT
- SELECT... FOR XML PATH
- SEND
- SET DATEFORMAT
- SET DEADLOCK_PRIORITY
- SET FMTONLY
- SET FORCEPLAN
- SET NUMERIC_ROUNDABORT ON
- SET OFFSETS
- SET REMOTE_PROC_TRANSACTIONS
- SET SHOWPLAN_TEXT
- SET SHOWPLAN_XML
- SET STATISTICS
- SET STATISTICS PROFILE
- SET STATISTICS TIME
- SET STATISTICS XML
- SHUTDOWN statement
- UPDATE STATISTICS
- UPDATETEXT
- Using EXECUTE to call a SQL function
- VIEW... CHECK OPTION clause
- VIEW... VIEW_METADATA clause
- WAITFOR DELAY
- WAITFOR TIME
- WAITFOR, RECEIVE
- WITH XMLNAMESPACES construct

- WRITETEXT
- XPATH expressions

Trabalhar com o Babelfish para Aurora PostgreSQL

Visão geral

Você pode usar o procedimento a seguir para instâncias de banco de dados do Amazon RDS que executam o Babelfish para Aurora PostgreSQL a fim de melhorar a performance de consultas:

- sp_babelfish_volatility
- sp_execute_postgresql

sp_babelfish_volatility

A volatilidade da função do PostgreSQL ajuda o otimizador a melhorar a execução de consultas. Quando usada em partes de determinadas cláusulas, tem um impacto significativo na performance de consultas.

Sintaxe

```
sp_babelfish_volatility 'function_name', 'volatility'
```

Argumentos

function_name (opcional)

Você pode especificar o valor desse argumento como `schema_name.function_name`, com um nome de duas partes, ou somente `function_name`. Se você especificar somente `function_name`, o nome do esquema será o esquema padrão para o usuário atual.

volatility (opcional)

Os valores válidos de volatilidade do PostgreSQL são `stable`, `volatile` ou `immutable`. Para ter mais informações, consulte https://www.postgresql.org/docs/current/xfunc-volatility.html.

> **Note**
>
> Quando `sp_babelfish_volatility` é chamado com `function_name`, que tem várias definições, será gerado um erro.

Conjunto de resultados

Se os parâmetros não forem mencionados, o conjunto de resultados será exibido nas seguintes colunas: `schemaname`, `functionname`, `volatility`.

Observações de uso

A volatilidade da função do PostgreSQL ajuda o otimizador a melhorar a execução de consultas. Quando usada em partes de determinadas cláusulas, tem um impacto significativo na performance de consultas.

Exemplos

Os exemplos a seguir mostram como criar funções simples e, posteriormente, explicam como usar `sp_babelfish_volatility` nessas funções utilizando métodos diferentes.

```
1> create function f1() returns int as begin return 0 end
2> go
```

```
1> create schema test_schema
2> go
```

```
1> create function test_schema.f1() returns int as begin return 0 end
2> go
```

O exemplo a seguir mostra a volatilidade das funções:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           volatile
test_schema f1           volatile
```

O exemplo a seguir mostra como alterar a volatilidade das funções:

```
1> exec sp_babelfish_volatility 'f1','stable'
2> go
1> exec sp_babelfish_volatility 'test_schema.f1','immutable'
```

```
2> go
```

Quando você especifica somente function_name, são exibidos o nome do esquema, o nome da função e a volatilidade da função. O exemplo a seguir mostra a volatilidade das funções depois de alterar os valores:

```
1> exec sp_babelfish_volatility 'test_schema.f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
test_schema f1           immutable
```

```
1> exec sp_babelfish_volatility 'f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
```

Quando você não especifica nenhum argumento, é exibida uma lista de funções (nome do esquema, nome da função, volatilidade das funções) presentes no banco de dados atual:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
test_schema f1           immutable
```

sp_execute_postgresql

É possível executar instruções PostgreSQL do endpoint T-SQL. Isso simplifica suas aplicações, pois você não precisa sair da porta T-SQL para executar essas instruções.

Sintaxe

```
sp_execute_postgresql [ @stmt = ] statement
```

Argumentos

declaração [@stmt]

O argumento é um datatype varchar. Esse argumento aceita declarações do dialeto PG.

> **Note**
>
> Você só pode passar uma declaração do dialeto PG como argumento; caso contrário, ela gerará o seguinte erro.

```
1>exec sp_execute_postgresql 'create extension pg_stat_statements; drop extension
 pg_stat_statements'
2>go
```

```
Msg 33557097, Level 16, State 1, Server BABELFISH, Line 1
expected 1 statement but got 2 statements after parsing
```

Observações de uso

CRIAR EXTENSÃO

Cria e carrega uma nova extensão no banco de dados atual.

```
1>EXEC sp_execute_postgresql 'create extension [ IF NOT EXISTS ] <extension name>
 [ WITH ] [SCHEMA schema_name] [VERSION version]';
2>go
```

O exemplo a seguir mostra como criar uma extensão:

```
1>EXEC sp_execute_postgresql 'create extension pg_stat_statements with schema sys
 version "1.10"';
2>go
```

Use o comando a seguir para acessar objetos de extensão:

```
1>select * from pg_stat_statements;
2>go
```

> **Note**
>
> Se o nome do esquema não for fornecido explicitamente durante a criação da extensão, por padrão, as extensões serão instaladas no esquema público. Você deve fornecer o qualificador de esquema para acessar os objetos de extensão, conforme mencionado abaixo:

```
1>select * from [public].pg_stat_statements;
2>go
```

Extensões compatíveis

As seguintes extensões disponíveis com o Aurora PostgreSQL funcionam com o Babelfish.

- `pg_stat_statements`
- `tds_fdw`
- `fuzzystrmatch`

Limitações

- Os usuários precisam ter a função sysadmin no T-SQL e rds_superuser no postgres para instalar as extensões.
- As extensões não podem ser instaladas em esquemas criados pelo usuário e também em esquemas dbo e guest para bancos de dados master, tempdb e msdb.
- Não há compatibilidade para a opção CASCADE.

ALTERAR EXTENSÃO

Você pode fazer a atualização para uma nova versão de extensão usando a extensão ALTER.

```
1>EXEC sp_execute_postgresql 'alter extension <extension name> UPDATE TO
 <new_version>';
2>go
```

Limitações

- Só é possível fazer a atualização da versão da extensão usando a instrução ALTER Extension. Não há compatibilidade para outras operações.

EXTENSÃO DROP

Elimina a extensão especificada. Você também pode usar as opções `if exists` ou `restrict` para remover a extensão.

```
1>EXEC sp_execute_postgresql 'drop extension <extension name>';
2>go
```

Limitações

- Não há compatibilidade para a opção CASCADE.

O Babelfish comporta métodos de tipo de dados XML.

A partir da versão 5.4.0, o Babelfish agora comporta procedimentos armazenados sp_xml_preparedocument e sp_xml_removedocument, função de conjunto de linhas OPENXML() e o método de tipo de dados xml .VALUE(). Com essas funções e procedimentos, a consulta de dados XML se torna muito mais fácil.

Noções básicas sobre os procedimentos e métodos XML

- sp_xml_preparedocument: o procedimento sp_xml_preparedocument analisa um texto XML fornecido como entrada e exibe um identificador para esse documento. Esse identificador é válido durante a sessão ou até ser removido pelo sp_xml_removedocument.
- sp_xml_removedocument: o procedimento sp_xml_removedocument invalida o identificador que foi criado pelo procedimento sp_xml_preparedocument.
- OPENXML(): o OPENXML fornece uma visualização do conjunto de linhas sobre um documento XML. Como o OPENXML é um provedor de conjunto de linhas e exibe um conjunto de linhas, podemos usar o OPENXML na cláusula FROM da mesma forma que podemos usar qualquer outra tabela, visualização ou função com valor de tabela.
- VALUE():o método VALUE() do tipo de dados XML é usado para extrair um valor de uma instância XML armazenada em uma coluna, parâmetro ou variável do tipo xml.

Limitações nos procedimentos e métodos XML do Babelfish

- O Babelfish aceita apenas a sintaxe XPATH 1.0 para o segundo argumento (ou seja, ROWPATTERN) de OPENXML().
- As metapropriedades e o sinalizador 8 não são atualmente aceitos em OPENXML ().
- O Babelfish aceita apenas a sintaxe XPATH 1.0 para o primeiro argumento (ou seja, XQuery) do método de tipo de dados VALUE().

Performance e escalabilidade do Amazon Aurora PostgreSQL

As seções a seguir descrevem o gerenciamento da performance e da escalabilidade de um cluster de bancos de dados Amazon Aurora PostgreSQL. Também inclui informações sobre outras tarefas de manutenção.

Tópicos

- Dimensionar instâncias de bancos de dados Aurora PostgreSQL
- Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL
- Limites de armazenamento temporário para o Aurora PostgreSQL
- Páginas grandes para o Aurora PostgreSQL
- Teste do Amazon Aurora PostgreSQL usando consultas de injeção de falhas
- Exibir o status do volume para um cluster de bancos de dados Aurora PostgreSQL
- Como especificar o disco de RAM para o stats_temp_directory
- Gerenciar arquivos temporários com o PostgreSQL

Dimensionar instâncias de bancos de dados Aurora PostgreSQL

Você pode escalar instâncias de banco de dados Aurora PostgreSQL de duas maneiras: com escalabilidade de instância e escalabilidade de leitura. Para ter mais informações sobre a escalabilidade de leitura, consulte Escalabilidade de leitura.

Você pode escalar o cluster de bancos de dados Aurora Postgree modificando a classe da instância de banco de dados para cada instância de banco de dados do cluster. O Aurora PostgreSQL é compatível com várias classes de instância de banco de dados otimizada para o Aurora. Não use classes de instância db.t2 ou db.t3 para clusters do Aurora maiores com tamanho superior a 40 terabytes (TB).

> **Note**
>
> Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais detalhes sobre as classes de instâncias T, consulte Tipos de classe de instância de banco de dados.

A escalabilidade não é instantânea. Pode levar 15 minutos ou mais para concluir a alteração em uma classe de instância de banco de dados diferente. Se você usar essa abordagem para modificar a classe da instância de banco de dados, aplique a alteração durante a próxima janela de manutenção programada (em vez de fazer isso imediatamente) para evitar o impacto nos usuários.

Como alternativa à modificação da classe de instância de banco de dados diretamente, você pode minimizar o tempo de inatividade usando os recursos de alta disponibilidade do Amazon Aurora.

Primeiro, adicione uma réplica do Aurora ao cluster. Ao criar a réplica, escolha o tamanho da classe de instância de banco de dados que deseja usar para o cluster. Quando a réplica do Aurora é sincronizada com o cluster, você faz o failover para a réplica recém-adicionada. Para saber mais, consulte Réplicas do Aurora e Failover rápido com o Amazon Aurora PostgreSQL.

Para obter especificações detalhadas das classes de instância de banco de dados compatíveis com o Aurora PostgreSQL, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.

Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL

Um cluster de bancos de dados do Aurora aloca recursos baseados na classe da instância de banco de dados e na memória disponível. Cada conexão com o cluster de banco de dados consome quantidades incrementais desses recursos, como memória e CPU. A memória consumida por conexão varia de acordo com o tipo de consulta, contagem e se tabelas temporárias são usadas. Até mesmo uma conexão ociosa consome memória e CPU. Isso porque quando as consultas são realizadas em uma conexão, mais memória é alocada para cada consulta e ela não é liberada completamente, mesmo quando o processamento é interrompido. Assim, recomendamos que você verifique se suas aplicações não estão segurando conexões ociosas: cada uma desperdiça recursos e afeta negativamente a performance. Para ter mais informações, consulte Recursos consumidos por conexões ociosas do PostgreSQL.

O número máximo de conexões permitido para uma instância de bancos de dados Aurora PostgreSQL é determinado pelo valor de parâmetro `max_connections` especificado no grupo de parâmetros para esta instância de banco de dados. O cenário ideal para o parâmetro `max_connections` é aquele que suporta todas as conexões do cliente que sua aplicação precisa, sem excesso de conexões não utilizadas, além de pelo menos mais 3 conexões para suportar automação da AWS. Antes de modificar a configuração do parâmetro `max_connections`, recomendamos considerar o seguinte:

- Se o valor de `max_connections` é muito baixo, a instância de bancos de dados Aurora PostgreSQL pode não ter conexões suficientes disponíveis quando os clientes tentam se conectar. Se isso acontecer, ele tentará se conectar usando o `psql` para gerar mensagens de erro, como as seguintes:

```
psql: FATAL: remaining connection slots are reserved for non-replication superuser
 connections
```

- Se o valor de `max_connections` excede o número de conexões que são realmente necessárias, as conexões não utilizadas podem causar queda de performance.

O valor padrão de `max_connections` é derivado da seguinte função `LEAST` do Aurora PostgreSQL:

`LEAST({DBInstanceClassMemory/9531392},5000)`.

Se quiser alterar o valor para `max_connections`, você precisará criar um grupo de parâmetros de cluster de banco de dados personalizado e alterar seu valor. Depois de aplicar o grupo de parâmetros de banco de dados personalizado ao cluster, reinicialize a instância principal para que o novo valor seja aplicado. Para obter mais informações, consulte Amazon Aurora PostgreSQL parameters e Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

> **Tip**
>
> Se suas aplicações abrem e fecham conexões com frequência ou mantêm um grande número de conexões de longa duração abertas, recomendamos usar o Amazon RDS Proxy. O RDS Proxy é um proxy de banco de dados totalmente gerenciado e altamente disponível que usa grupos de conexões para compartilhar conexões de banco de dados de forma segura e eficiente. Para saber mais sobre o RDS Proxy, consulte Amazon RDS Proxypara Aurora.

Para obter detalhes sobre como as instâncias do Aurora Serverless v2 lidam com esse parâmetro, consulte Conexões máximas do Aurora Serverless v2.

Limites de armazenamento temporário para o Aurora PostgreSQL

O Aurora PostgreSQL armazena tabelas e índices no subsistema de armazenamento do Aurora. O Aurora PostgreSQL usa armazenamento temporário separado para arquivos temporários não persistentes. Isso inclui arquivos que são usados para fins como classificar grandes conjuntos de dados durante o processamento de consultas ou para operações de compilação de índice. Para obter mais informações, consulte o artigo Como posso solucionar problemas de armazenamento local em instâncias do Aurora compatível com PostgreSQL?.

Esses volumes de armazenamento local são apoiados pelo Amazon Elastic Block Store e podem ser estendidos utilizando uma classe de instância de banco de dados maior. Para ter mais informações sobre armazenamento, consulte Armazenamento do Amazon Aurora. Também é possível aumentar o armazenamento local para objetos temporários usando um tipo de instância habilitado para NVMe

e objetos temporários habilitados para o Aurora Optimized Reads. Para obter mais informações, consulte Melhorar a performance das consultas do Aurora PostgreSQL com o Aurora Optimized Reads.

> **Note**
>
> Você pode ver eventos de `storage-optimization` ao dimensionar instâncias de banco de dados, por exemplo, de db.r5.2xlarge para db.r5.4xlarge.

A tabela a seguir mostra a quantidade máxima de armazenamento temporário disponível para cada classe de instância de bancos de dados Aurora PostgreSQL. Para ter mais informações sobre o suporte a classes de instância de banco de dados para o Aurora, consulte Classes de instâncias de banco de dados do Amazon Aurora.

DB instance class	Armazenamento temporário máximo disponível (GiB)
db.x2g.16xlarge	1.829
db.x2g.12xlarge	1.606
db.x2g.8xlarge	1.071
db.x2g.4xlarge	535
db.x2g.2xlarge	268
db.x2g.xlarge	134
db.x2g.large	67
db.r8g.48xlarge	3.072
db.r8g.24xlarge	1.536
db.r8g.16xlarge	998
db.r8g.12xlarge	749

DB instance class	Armazenamento temporário máximo disponível (GiB)
db.r8g.8xlarge	499
db.r8g.4xlarge	250
db.r8g.2xlarge	125
db.r8g.xlarge	63
db.r8g.large	31
db.r7g.16xlarge	1.008
db.r7g.12xlarge	756
db.r7g.8xlarge	504
db.r7g.4xlarge	252
db.r7g.2xlarge	126
db.r7g.xlarge	63
db.r7g.large	32
db.r7i.48xlarge	3.072
db.r7i.24xlarge	1.500
db.r7i.16xlarge	1.008
db.r7i.12xlarge	748
db.r7i.8xlarge	504
db.r7i.4xlarge	249
db.r7i.2xlarge	124
db.r7i.xlarge	62

DB instance class	Armazenamento temporário máximo disponível (GiB)
db.r7i.large	31
db.r6g.16xlarge	1008
db.r6g.12xlarge	756
db.r6g.8xlarge	504
db.r6g.4xlarge	252
db.r6g.2xlarge	126
db.r6g.xlarge	63
db.r6g.large	32
db.r6i.32xlarge	1.829
db.r6i.24xlarge	1.500
db.r6i.16xlarge	1.008
db.r6i.12xlarge	748
db.r6i.8xlarge	504
db.r6i.4xlarge	249
db.r6i.2xlarge	124
db.r6i.xlarge	62
db.r6i.large	31
db.r5.24xlarge	1.500
db.r5.16xlarge	1.008
db.r5.12xlarge	748

DB instance class	Armazenamento temporário máximo disponível (GiB)
db.r5.8xlarge	504
db.r5.4xlarge	249
db.r5.2xlarge	124
db.r5.xlarge	62
db.r5.large	31
db.r4.16xlarge	960
db.r4.8xlarge	480
db.r4.4xlarge	240
db.r4.2xlarge	120
db.r4.xlarge	60
db.r4.large	30
db.t4g.large	16,5
db.t4g.medium	8,13
db.t3.large	16
db.t3.medium	7,5

> **Note**
>
> Os tipos de instância habilitados para NVMe podem aumentar o espaço temporário disponível em até o tamanho total do NVMe. Para obter mais informações, consulte Melhorar a performance das consultas do Aurora PostgreSQL com o Aurora Optimized Reads.

É possível monitorar o armazenamento temporário disponível para uma instância de banco de dados com a métrica do CloudWatch `FreeLocalStorage`, descrita em Métricas do Amazon CloudWatch para o Amazon Aurora. (Isso não se aplica ao Aurora Serverless v2.)

Para algumas workloads, é possível reduzir a quantidade de armazenamento temporário alocando mais memória para os processos que estão realizando a operação. Para aumentar a memória disponível para uma operação, aumentando os valores dos parâmetros do PostgreSQL work_mem ou maintenance_work_mem.

Páginas grandes para o Aurora PostgreSQL

Páginas grandes são um recurso de gerenciamento de memória que reduz a sobrecarga quando uma instância de banco de dados está trabalhando com grandes blocos contíguos de memória, como os usados por buffers compartilhados. Esse recurso PostgreSQL é compatível com todas as versões do Aurora PostgreSQL atualmente disponíveis.

O parâmetro `Huge_pages` é ativado por padrão para todas as classes de instância de banco de dados que não sejam classes de instância de banco de dados t3.medium, db.t3.large, db.t4g.medium e db.t4g.large. Você não pode alterar o valor do parâmetro `huge_pages` nem desativar esse recurso nas classes de instância compatíveis do Aurora PostgreSQL.

Nas instâncias de banco de dados do Aurora PostgreSQL que não comportam o recurso de memória de páginas enormes, o uso da memória específica do processo pode aumentar sem alterações correspondentes na workload.

O sistema aloca segmentos de memória compartilhada, como o cache do buffer, durante a inicialização do servidor. Quando não há páginas enormes de memória, o sistema não cobra essas alocações do processo postmaster. Em vez disso, ele inclui a memória no processo que primeiro acessou cada página de 4 KB no segmento de memória compartilhada.

> **Note**
>
> As conexões ativas compartilham a memória alocada conforme necessário, independentemente de como o uso da memória compartilhada é rastreado entre os processos.

Teste do Amazon Aurora PostgreSQL usando consultas de injeção de falhas

Você pode testar a tolerância a falhas do cluster de banco de dados Aurora PostgreSQL usando consultas de injeção de falha. Consultas de injeção de falhas são emitidas como comandos SQL para uma instância do Amazon Aurora. As consultas de injeção de falhas permitem que você trave a instância para que consiga testar o failover e a recuperação. Você também pode simular uma falha da réplica do Aurora, uma falha do disco e um congestionamento do disco. As consultas de injeção de falhas são compatíveis com todas as versões disponíveis do Aurora PostgreSQL, conforme mostrado a seguir.

- Aurora PostgreSQL versões 12, 13, 14 e posteriores
- Aurora PostgreSQL versão 11.7 e posteriores
- Aurora PostgreSQL versão 10.11 e posteriores

Tópicos

- Teste de pane da instância
- Teste de falha em uma réplica do Aurora
- Teste de uma falha de disco
- Teste de congestionamento de disco

Quando uma consulta de injeção de falha especifica uma falha, ela força uma falha da instância de bancos de dados Aurora PostgreSQL. As outras consultas de injeção de falha resultam em simulações de eventos de falha, mas não desencadeiam o evento. Ao enviar uma consulta de injeção de falha, especifique também um período para a simulação do evento de falha.

Você pode enviar uma consulta de injeção de falha a uma das instâncias da réplica do Aurora conectando-se ao endpoint da réplica do Aurora. Para obter mais informações, consulte Conexões de endpoints do Amazon Aurora.

Teste de pane da instância

Você pode forçar uma pane em uma instância do Aurora PostgreSQL usando a função de consulta de injeção de falha `aurora_inject_crash()`.

No que tange a essa consulta de injeção de falha, não ocorre um failover. Se desejar testar um failover, é possível escolher a ação de instância Failover para o cluster de banco de dados no console do RDS ou usar o comando failover-db-cluster da AWS CLI ou a operação de API FailoverDBCluster do RDS.

Sintaxe

```
SELECT aurora_inject_crash ('instance' | 'dispatcher' | 'node');
```

Opções

Esta consulta de injeção de falha considera um dos seguintes tipos de pane. O tipo de falha não diferencia maiúsculas de minúsculas:

'instance'

uma pane é simulada no banco de dados compatível com PostgreSQL para a instância do Amazon Aurora.

'dispatcher'

uma falha no dispatcher é simulada na instância primária do cluster de banco de dados de Aurora. O dispatcher grava atualizações no volume do cluster para um cluster de banco de dados do Amazon Aurora.

'node'

uma pane é simulada no banco de dados compatível com PostgreSQL e no dispatcher para a instância do Amazon Aurora.

Teste de falha em uma réplica do Aurora

Você pode simular a falha de uma réplica do Aurora usando a função de consulta de injeção de falha `aurora_inject_replica_failure()`.

Uma falha de réplica do Aurora bloqueia a replicação para a réplica do Aurora ou todas as réplicas do Aurora no cluster de banco de dados pela porcentagem especificada para o intervalo de tempo especificado. Quando o intervalo de tempo acabar, as réplicas afetadas do Aurora são automaticamente sincronizadas com a instância primária.

Sintaxe

```
SELECT aurora_inject_replica_failure(
   percentage_of_failure,
   time_interval,
   'replica_name'
);
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

percentage_of_failure

A porcentagem de replicação para bloquear durante o evento de falha. Esse valor pode ser um duplo entre 0 e 100. Se você especificar 0, nenhuma replicação será bloqueada. Se você especificar 100, toda replicação será bloqueada.

time_interval

O tempo necessário para simular a falha da réplica do Aurora. O intervalo é em segundos. Por exemplo, se o valor é 20, a simulação é executada por 20 segundos.

> **Note**
>
> Especifique o intervalo de tempo do evento de falha da réplica do Aurora; com cautela. Se você especificar um intervalo muito longo e sua instância registrar uma grande quantidade de dados durante o evento de falha, seu cluster de banco de dados Aurora poderá presumir que sua réplica do Aurora entrou em pane e substituí-la.

replica_name

A réplica do Aurora na qual a simulação de falha será injetada. Especifique o nome de uma réplica do Aurora para simular uma falha de uma única réplica do Aurora. Especifique uma string vazia para simular falhas para todas as réplicas do Aurora no cluster de banco de dados.

Para identificar nomes de réplica, consulte a coluna `server_id` na função `aurora_replica_status()`. Por exemplo:

```
postgres=> SELECT server_id FROM aurora_replica_status();
```

Teste de uma falha de disco

Você pode simular a falha de um disco para um cluster de bancos de dados Aurora PostgreSQL usando a função de consulta de injeção de falha `aurora_inject_disk_failure()`.

Durante uma simulação de falha de disco, o cluster de bancos de dados Aurora PostgreSQL marca aleatoriamente segmentos do disco como falhos. As solicitações feitas a esses segmentos são bloqueadas durante a simulação.

Sintaxe

```
SELECT aurora_inject_disk_failure(
   percentage_of_failure,
   index,
   is_disk,
   time_interval
);
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

percentage_of_failure

a porcentagem do disco para marcar como falha durante o evento de falha. Esse valor pode ser um duplo entre 0 e 100. Se você especificar 0, nenhum dos discos será marcado como falhos. Se você especificar 100, o disco todo será marcado como falho.

índice

Um bloco de dados lógico específico para simular o evento de falha. Se você exceder o intervalo de blocos lógicos ou nós de armazenamento de dados disponíveis, receberá um erro informando o valor máximo do índice pode ser especificado. Para evitar esse erro, consulte Exibir o status do volume para um cluster de bancos de dados Aurora PostgreSQL.

is_disk

Indica se a falha de injeção é para um bloco lógico ou para um nó de armazenamento. Especificar como true significa que as falhas de injeção são para um bloco lógico. Especificar como false significa que as falhas de injeção são para um nó de armazenamento.

time_interval

A quantidade de tempo para simular a falha do disco. O intervalo é em segundos. Por exemplo, se o valor é 20, a simulação é executada por 20 segundos.

Teste de congestionamento de disco

Você pode simular o congestionamento de um disco para um cluster de bancos de dados Aurora PostgreSQL usando a função de consulta de injeção de falha `aurora_inject_disk_congestion()`.

Durante uma simulação de congestionamento de disco, o cluster de bancos de dados Aurora PostgreSQL marca aleatoriamente segmentos do disco como congestionados. As solicitações feitas a esses segmentos serão atrasadas entre o tempo de atraso mínimo e máximo especificado enquanto durar a simulação.

Sintaxe

```
SELECT aurora_inject_disk_congestion(
   percentage_of_failure,
   index,
   is_disk,
   time_interval,
   minimum,
   maximum
);
```

Opções

Esta consulta de injeção de falha considera os seguintes parâmetros:

percentage_of_failure

A porcentagem do disco para marcar como congestionada durante o evento de falha. Este é um valor duplo entre 0 e 100. Se você especificar 0, nenhum dos discos será marcado como congestionados. Se você especificar 100, o disco todo será marcado como congestionado.

índice

Um bloco lógico específico de dados ou nó de armazenamento usado para simular o evento de falha.

Se você exceder o intervalo de blocos lógicos ou nós de armazenamento de dados disponíveis, receberá um erro informando o valor máximo do índice que você pode especificar. Para evitar esse erro, consulte Exibir o status do volume para um cluster de bancos de dados Aurora PostgreSQL.

is_disk

Indica se a falha de injeção é para um bloco lógico ou para um nó de armazenamento. Especificar como true significa que as falhas de injeção são para um bloco lógico. Especificar como false significa que as falhas de injeção são para um nó de armazenamento.

time_interval

A quantidade de tempo para simular o congestionamento de disco. O intervalo é em segundos. Por exemplo, se o valor é 20, a simulação é executada por 20 segundos.

mínimo, máximo

A quantidade mínima e máxima de atraso de congestionamento em milissegundos. Os valores válidos variam de 0,0 a 100,0 milissegundos. Os segmentos de disco marcados como congestionados serão atrasados durante um período aleatório de tempo dentro do intervalo mínimo e máximo enquanto durar a simulação. O valor máximo deve ser maior que o valor mínimo.

Exibir o status do volume para um cluster de bancos de dados Aurora PostgreSQL

No Amazon Aurora, um volume de cluster de banco de dados consiste em um acervo de blocos lógicos. Cada um deles representa 10 gigabytes de armazenamento alocado. Esses blocos são chamados de grupos de proteção.

Os dados em cada grupo de proteção são replicados entre seis dispositivos de armazenamento físico, chamados de nós de armazenamento. Esses nós de armazenamento são alocados em três zonas de disponibilidade (AZs) na região onde reside o cluster de banco de dados. Por sua vez, cada nó de armazenamento contém um ou mais blocos lógicos de dados para o volume do cluster de banco de dados. Para obter mais informações sobre os grupos de proteção e os nós de armazenamento, consulte Introducing the Aurora storage engine (Apresentação do mecanismo de armazenamento do Aurora) no blog de banco de dados da AWS. Para saber mais sobre volumes de cluster do Aurora em geral, consulte Armazenamento do Amazon Aurora.

Use a função `aurora_show_volume_status()` para retornar as seguintes variáveis de status do servidor:

- `Disks` — o número total de blocos lógicos de dados para o volume do cluster de banco de dados.
- `Nodes` — o número total de nós de armazenamento para o volume do cluster de banco de dados.

É possível usar a função `aurora_show_volume_status()` para ajudar a evitar um erro ao usar a função de injeção de falha `aurora_inject_disk_failure()`. A função de injeção de falha `aurora_inject_disk_failure()` simula a falha de um nó de armazenamento completo ou um único bloco lógico de dados dentro de um nó de armazenamento Na função, especifique o valor do índice de um bloco lógico de dados ou de um nó de armazenamento específico. No entanto, a instrução retornará um erro se você especificar um valor de índice maior do que o número de blocos lógicos de dados ou de nós de armazenamento usados pelo volume do cluster de banco de dados. Para obter mais informações sobre as consultas de injeção de falha, veja Teste do Amazon Aurora PostgreSQL usando consultas de injeção de falhas.

> **Note**
>
> A função `aurora_show_volume_status()` está disponível para a versão 10.11 do Aurora PostgreSQL. Para obter mais informações sobre as versões do Aurora PostgreSQL, consulte Versões Amazon Aurora PostgreSQL e versões do mecanismo.

Sintaxe

```
SELECT * FROM aurora_show_volume_status();
```

Exemplo

```
customer_database=> SELECT * FROM aurora_show_volume_status();
 disks | nodes
-------+-------
    96 |    45
```

Como especificar o disco de RAM para o stats_temp_directory

Você pode usar o parâmetro `rds.pg_stat_ramdisk_size` do Aurora PostgreSQL para especificar a memória do sistema alocada a um disco de RAM para armazenar o

`stats_temp_directory` do PostgreSQL. O parâmetro de disco de RAM só está disponível no Aurora PostgreSQL 14 e versões anteriores.

Mediante certas workloads, definir este parâmetro pode melhorar a performance e diminuir os requisitos de E/S. Para obter mais informações sobre `stats_temp_directory`, consulte Run-time Statistics na documentação do PostgreSQL. A partir da versão 15 do PostgreSQL, a comunidade do PostgreSQL passou a usar memória compartilhada dinâmica. Portanto, não há necessidade de configurar `stats_temp_directory`.

Para habilitar um disco de RAM para `stats_temp_directory`, configure o parâmetro `rds.pg_stat_ramdisk_size` como um valor diferente de zero no grupo de parâmetros do cluster de banco de dados usado pelo seu cluster de banco de dados. Esse parâmetro denota MB, portanto, você deve usar um valor inteiro. Expressões, fórmulas e funções não são válidas para o parâmetro `rds.pg_stat_ramdisk_size`. Reinicie o cluster de banco de dados para que a alteração entre em vigor. Para obter informações sobre como configurar parâmetros, consulte Grupos de parâmetros para Amazon Aurora. Para obter mais informações sobre como reiniciar o cluster de banco de dados, consulte Reinicializar um cluster de banco de dados do Amazon Aurora ou instância de banco de dados do Amazon Aurora.

Por exemplo, o seguinte comando da AWS CLI define o parâmetro do disco RAM como 256 MB.

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name db-cl-pg-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

Após reiniciar o cluster de banco de dados, execute o comando a seguir para ver o status de `stats_temp_directory`:

```
postgres=> SHOW stats_temp_directory;
```

O comando deve retornar o seguinte:

```
stats_temp_directory
----------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

Gerenciar arquivos temporários com o PostgreSQL

No PostgreSQL, uma consulta complexa pode executar algumas operações de classificação ou de hash simultaneamente, com cada uma utilizando a memória da instância para armazenar resultados até o valor especificado no parâmetro `work_mem`. Quando a memória da instância não é suficiente, arquivos temporários são criados para armazenar os resultados. Eles são gravados em disco para concluir a execução da consulta. Posteriormente, esses arquivos são removidos automaticamente após a conclusão da consulta. No Aurora PostgreSQL. Esses arquivos compartilham armazenamento com outros arquivos de log.. Você pode monitorar o espaço de armazenamento local de seu cluster de banco de dados do Aurora PostgreSQL observando a métrica do Amazon CloudWatch para `FreeLocalStorage`. Para ter mais informações, consulte Solucionar problemas de armazenamento local.

Recomendamos o uso de clusters do Aurora com otimização de memória para workloads com várias consultas simultâneas que aumentam o uso de arquivos temporários. Esses clusters usam o armazenamento ao nível do bloco de unidade de estado sólido (SSD) local baseado em memória expressa não volátil (NVMe) para guardar os arquivos temporários. Para obter mais informações, consulte Melhorar a performance das consultas do Aurora PostgreSQL com o Aurora Optimized Reads.

Você pode usar os parâmetros e as funções a seguir para gerenciar os arquivos temporários em sua instância.

- **`temp_file_limit`** – esse parâmetro cancela qualquer consulta que exceda o tamanho de temp_files em KB. Esse limite impede que qualquer consulta seja executada indefinidamente e consuma espaço em disco com arquivos temporários. Você pode estimar o valor utilizando os resultados do parâmetro `log_temp_files`. Como prática recomendada, examine o comportamento da workload e defina o limite de acordo com a estimativa. O exemplo a seguir mostra como uma consulta é cancelada quando ela excede o limite.

```
postgres=>select * from pgbench_accounts, pg_class, big_table;
```

```
ERROR: temporary file size exceeds temp_file_limit (64kB)
```

- **`log_temp_files`** – esse parâmetro envia mensagens ao postgresql.log quando os arquivos temporários de uma sessão são removidos. Esse parâmetro produz logs após a conclusão bem-

sucedida de uma consulta. Portanto, isso pode não ajudar na solução de problemas de consultas ativas e de longa duração.

O exemplo a seguir mostra que, quando a consulta é concluída com êxito, as entradas são registradas no arquivo postgresql.log enquanto os arquivos temporários são limpos.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir** – essa função que está disponível no RDS para PostgreSQL 13 e versões posteriores oferece visibilidade sobre o uso atual de arquivos temporários. A consulta concluída não aparece nos resultados da função. No exemplo a seguir, você pode visualizar os resultados dessa função.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;
```

```
query
------------------------------------------------------------------------------------
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

O nome do arquivo inclui o ID de processamento (PID) da sessão que gerou o arquivo temporário. Uma consulta mais avançada, como no exemplo a seguir, executa uma soma dos arquivos temporários para cada PID.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements** – se você ativar o parâmetro pg_stat_statements, poderá visualizar o uso médio de arquivos temporários por chamada. Você pode identificar o query_id da consulta e usá-lo para examinar o uso do arquivo temporário, conforme mostrado no exemplo a seguir.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                     388678
(1 row)
```

- **`Performance Insights`**: no painel do Performance Insights, você pode visualizar o uso temporário de arquivos ativando as métricas temp_bytes e temp_files. Depois, você pode ver a média dessas duas métricas e como elas correspondem à workload da consulta. A exibição no Performance Insights não mostra especificamente as consultas que estão gerando os arquivos temporários. No entanto, ao combinar o Performance Insights com a consulta mostrada para `pg_ls_tmpdir`, você pode solucionar problemas, analisar e determinar as alterações em sua workload de consulta.

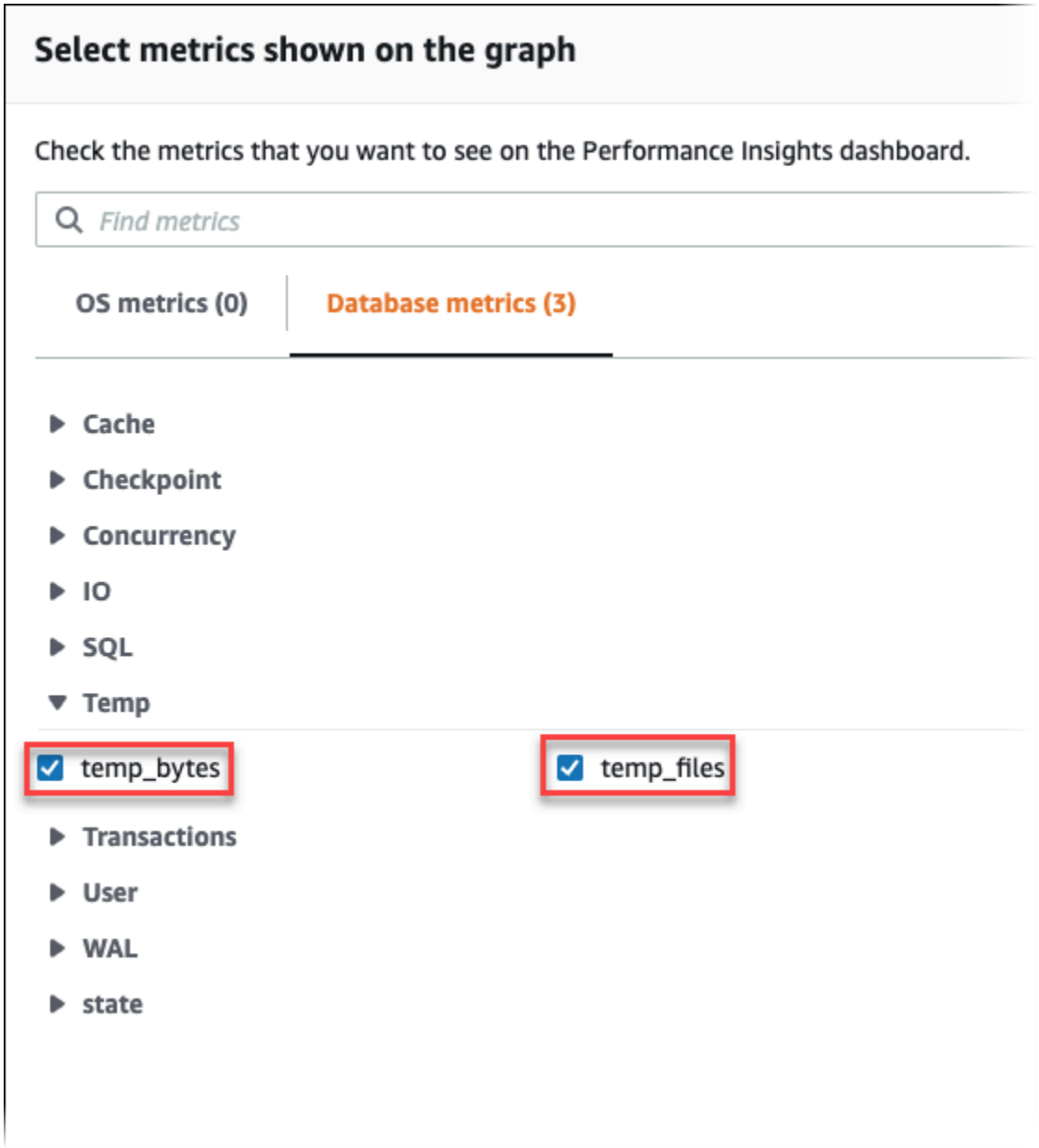
  Para ter mais informações sobre como analisar as métricas e as consultas com o Insights de Performance, consulte Análise de métricas usando o painel do Performance Insights.

  Para ver um exemplo de como visualizar o uso de arquivos temporários com o Insights de Performance, consulte Visualizar o uso de arquivos temporários com o Insights de Performance.

Visualizar o uso de arquivos temporários com o Insights de Performance

É possível usar o Insights de Performance para visualizar o uso de arquivos temporários ativando as métricas temp_bytes e temp_files. A visualização no Insights de Performance não mostra as consultas específicas que geram arquivos temporários, no entanto, ao combinar o Insights de Performance com a consulta mostrada para `pg_ls_tmpdir`, é possível solucionar problemas, analisar e determinar as alterações na workload de consulta.

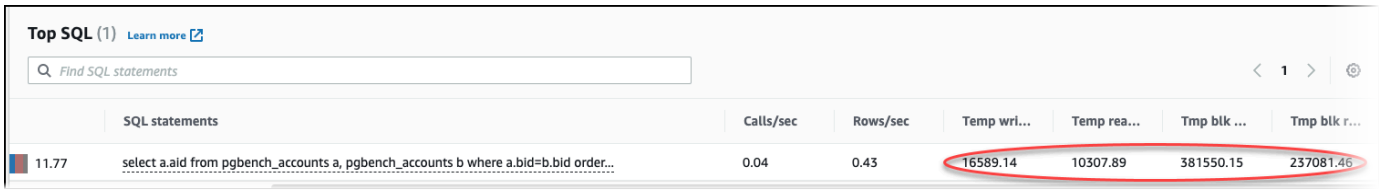
1. No painel do Performance Insights, selecione Gerenciar métricas.
2. Escolha Métricas de banco de dados e selecione as métricas temp_bytes e temp_files como mostrado na imagem a seguir.



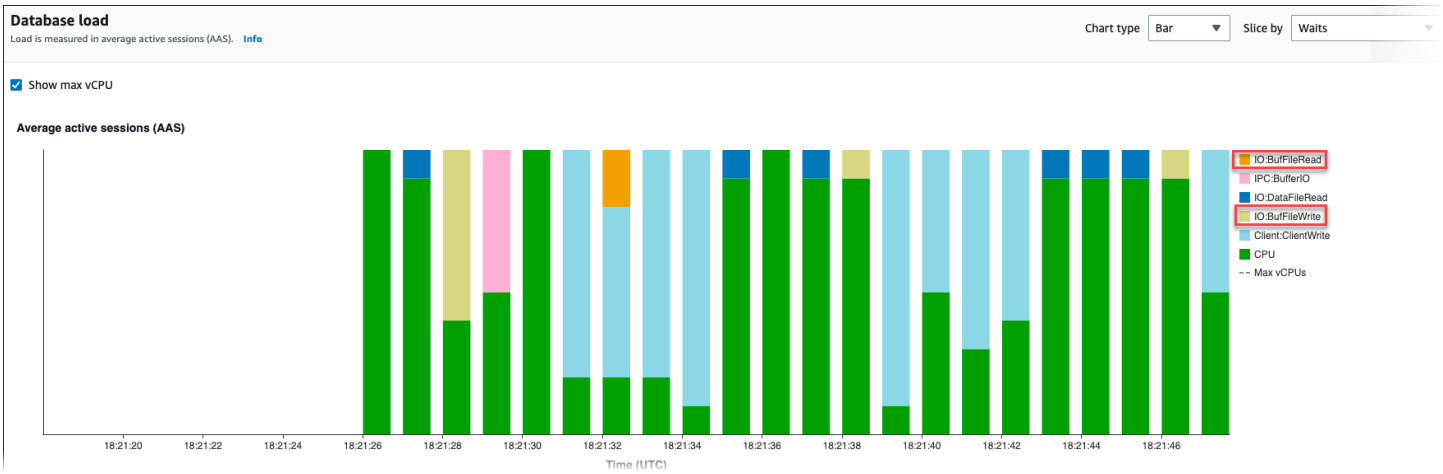
3. Na guia Top SQL, selecione o ícone Preferências.
4. Na janela Preferências, ative as estatísticas a seguir para serem exibidas na guia Top SQL e selecione Continuar.

   - Gravações temporárias/segundo
   - Leituras de temperatura/segundo
   - Gravação/chamada em bloco temporário
   - Leitura/chamada em bloco temporário

5. O arquivo temporário é dividido quando combinado com a consulta mostrada para `pg_ls_tmpdir`, conforme exibido no exemplo a seguir.



Os eventos `IO:BufFileRead` e `IO:BufFileWrite` ocorrem porque as principais consultas na workload geralmente criam arquivos temporários. Você pode usar o Insights de Performance para identificar as principais consultas que aguardam `IO:BufFileRead` e `IO:BufFileWrite` revisando "Média de sessões ativas (AAS)" nas seções "Carga do banco de dados" e "SQL principal".



Para obter mais informações sobre como usar o Insights de Performance para analisar as principais consultas e a carga por eventos de espera, consulte Visão geral da guia Top SQL (SQL principal) Você deve identificar e ajustar as consultas que aumentam o uso de arquivos temporários e os eventos de espera correspondentes. Para ter mais informações sobre esses eventos de espera e a correção, consulte, IO:BufFileRead e IO:BufFileWrite.

> **Note**
>
> O parâmetro `work_mem` controla quando a operação de classificação fica sem memória e os resultados são gravados em arquivos temporários. Recomendamos que você não altere a configuração desse parâmetro acima do valor padrão, pois isso permitiria que cada sessão do banco de dados consumisse mais memória. Além disso, uma única sessão que executa junções e classificações complexas pode realizar operações paralelas nas quais cada operação consome memória.

Como prática recomendada, quando você tem um relatório grande com várias junções e classificações, defina esse parâmetro no nível da sessão usando o comando `SET work_mem`. Depois, a alteração é aplicada somente à sessão atual e não altera o valor globalmente.

Ajustar com eventos de espera do Aurora PostgreSQL

Eventos de espera são uma ferramenta de ajuste importante para o Aurora PostgreSQL. Se você puder descobrir por que as sessões estão aguardando recursos e o que elas estão fazendo, poderá reduzir melhor os gargalos. Use as informações nesta seção para encontrar possíveis causas e ações corretivas. Antes de se aprofundar nesta seção, é altamente recomendável que você entenda os conceitos básicos do Aurora, principalmente os seguintes tópicos:

- Armazenamento do Amazon Aurora
- Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora

Important

Os eventos de espera nesta seção são específicos para o Aurora PostgreSQL. Utilize as informações nesta seção para ajustar somente o Amazon Aurora, e não o RDS para PostgreSQL.
Alguns eventos de espera nesta seção não têm análogos nas versões de código aberto desses mecanismos de banco de dados. Outros eventos de espera têm os mesmos nomes que os eventos em mecanismos de código aberto, mas se comportam de maneira diferente. Por exemplo, o armazenamento do Amazon Aurora funciona de maneira diferente do armazenamento de código aberto e, portanto, eventos de espera relacionados a armazenamento indicam condições de recursos diferentes.

Tópicos

- Conceitos essenciais para ajuste do Aurora PostgreSQL
- Eventos de espera do Aurora PostgreSQL
- Client:ClientRead
- Client:ClientWrite

- CPU
- IO:BufFileRead and IO:BufFileWrite
- IO:DataFileRead
- IO:XactSync
- IPC:DamRecordTxAck
- IPC: eventos de espera paralelos
- IPC:ProcArrayGroupUpdate
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:buffer_content (BufferContent)
- LWLock:buffer_mapping
- LWLock:BufferIO (IPC:BufferIO)
- LWLock:lock_manager
- LWLock:MultiXact
- LWLock:pg_stat_statements
- Tempo limite:PgSleep

Conceitos essenciais para ajuste do Aurora PostgreSQL

Antes de ajustar seu banco de dados Aurora PostgreSQL, aprenda o que são eventos de espera e por que eles ocorrem. Reveja também a arquitetura básica de memória e disco do Aurora PostgreSQL. Para obter um diagrama de arquitetura útil, consulte o wikibook PostgreSQL.

Tópicos

- Eventos de espera do Aurora PostgreSQL
- Memória do Aurora PostgreSQL
- Processos do Aurora PostgreSQL

Eventos de espera do Aurora PostgreSQL

Um evento de espera indica um recurso pelo qual uma sessão está aguardando. Por exemplo, o evento de espera `Client:ClientRead` ocorre quando o Aurora PostgreSQL está aguardando para receber dados do cliente. Os recursos comuns que uma sessão aguarda incluem:

- Acesso com thread único a um buffer, por exemplo, quando uma sessão está tentando modificar um buffer
- Uma linha que está bloqueada por outra sessão
- Uma leitura de arquivo de dados
- Uma gravação em arquivo de log

Por exemplo, para satisfazer uma consulta, a sessão pode realizar uma varredura de tabela completa. Se esses dados ainda não estiverem na memória, a sessão aguardará a conclusão da E/S do disco. Quando os buffers são lidos na memória, talvez a sessão precise aguardar, pois outras sessões estão acessando os mesmos buffers. O banco de dados registra as esperas utilizando um evento de espera predefinido. Esses eventos estão agrupados em categorias.

Por si só, um evento de espera não mostra um problema de performance. Por exemplo, se os dados solicitados não estão na memória, é necessário ler dados do disco. Se uma sessão bloquear uma linha para uma atualização, outra sessão aguardará que essa linha seja desbloqueada para poder atualizá-la. Uma confirmação exige a conclusão da gravação em um arquivo de log. Esperas são componentes integrais do funcionamento normal de um banco de dados.

Em geral, muitos eventos de espera indicam um problema de performance. Nesses casos, é possível utilizar os dados dos eventos de espera para determinar onde as sessões estão perdendo tempo. Por exemplo, se um relatório que é normalmente executado em minutos passou a demorar várias horas, é possível identificar os eventos de espera que mais contribuem para o tempo de espera total. Se você puder determinar as causas dos principais eventos de espera, às vezes pode aplicar alterações que melhoram a performance. Por exemplo, se a sua sessão está aguardando uma linha que foi bloqueada por outra sessão, é possível encerrar a sessão responsável pelo bloqueio.

Memória do Aurora PostgreSQL

A memória do Aurora PostgreSQL está dividida em compartilhada e local.

Tópicos

- Memória compartilhada no Aurora PostgreSQL
- Memória local no Aurora PostgreSQL

Memória compartilhada no Aurora PostgreSQL

O Aurora PostgreSQL aloca memória compartilhada quando a instância é iniciada. A memória compartilhada está dividida em várias subáreas. A seguir, você encontrará uma descrição das mais importantes.

Tópicos

- Buffers compartilhados
- Buffers de log de gravação antecipada (WAL)

Buffers compartilhados

O grupo de buffer compartilhado é uma área de memória do Aurora PostgreSQL que contém todas as páginas que estão ou estavam sendo utilizadas por conexões de aplicações. Uma página é a versão de memória de um bloco de disco. O grupo de buffer compartilhado armazena em cache os blocos de dados lidos do disco. O grupo reduz a necessidade de reler dados do disco, fazendo com que o banco de dados opere de maneira mais eficiente.

Cada tabela e índice são armazenados como uma matriz de páginas com tamanho fixo. Cada bloco contém várias tuplas, que correspondem a linhas. Uma tupla pode ser armazenada em qualquer página.

O grupo de buffer compartilhado possui memória finita. Se uma nova solicitação exigir uma página que não esteja na memória e não houver mais memória, o Aurora PostgreSQL removerá uma página utilizada com menos frequência para acomodar essa solicitação. A política de despejo é implementada por um algoritmo de varredura de relógio.

O parâmetro `shared_buffers` determina a quantidade de memória que o servidor dedica ao armazenamento em cache de dados.

Buffers de log de gravação antecipada (WAL)

Um buffer de log de gravação antecipada (WAL) mantém dados de transação que o Aurora PostgreSQL grava posteriormente no armazenamento persistente. Utilizando o mecanismo WAL, o Aurora PostgreSQL pode fazer o seguinte:

- Recupere dados após uma falha
- Reduzir a E/S de disco, evitando gravações frequentes em disco

Quando um cliente modifica dados, o Aurora PostgreSQL grava as alterações no buffer de WAL. Quando o cliente emite um `COMMIT`, o processo gravador WAL grava dados de transação no arquivo de WAL.

O parâmetro `wal_level` determina quantas informações são gravadas no WAL.

Memória local no Aurora PostgreSQL

Todo processo de backend aloca memória local para processamento de consultas.

Tópicos

- Área de memória de trabalho
- Área de memória de trabalho para manutenção
- Área de buffer temporária

Área de memória de trabalho

A área de memória de trabalhocontém dados temporários para consultas que executam classificações e hashes. Por exemplo, uma consulta com uma cláusula `ORDER BY` executa uma classificação. Consultas usam tabelas de hash em agregações e junções de hash.

O parâmetro `work_mem` é a quantidade de memória a ser utilizada por operações de classificação internas e tabelas de hash antes da gravação em arquivos de disco temporários. O valor padrão é 4 MB. Várias sessões podem ser executadas simultaneamente, e cada uma pode executar operações de manutenção em paralelo. Por esse motivo, a memória de trabalho total utilizada pode ser múltiplos da configuração `work_mem`.

Área de memória de trabalho para manutenção

A área de memória de trabalho para manutenção armazena dados em cache para operações de manutenção. Essas operações incluem aspiração, criação de índices e adição de chaves externas.

O parâmetro `maintenance_work_mem` especifica a quantidade máxima de memória a ser utilizada por operações de manutenção. O valor padrão é 64 MB. Uma sessão de banco de dados apenas pode executar uma operação de manutenção de cada vez.

Área de buffer temporária

A área de buffer temporária armazena tabelas temporárias em cache para cada sessão de banco de dados.

Cada sessão aloca buffers temporários conforme necessário até o limite especificado. Quando a sessão termina, o servidor limpa os buffers.

O parâmetro `temp_buffers` define o número máximo de buffers temporários utilizados por cada sessão. Antes do primeiro uso de tabelas temporárias em uma sessão, é possível alterar o valor de `temp_buffers`.

Processos do Aurora PostgreSQL

O Aurora PostgreSQL utiliza vários processos.

Tópicos

- Processo Postmaster
- Processos de backend
- Processos em segundo plano

Processo Postmaster

O processo de postmaster é o primeiro a ser iniciado quando você inicia o Aurora PostgreSQL. Ele tem as seguintes responsabilidades principais:

- Bifurcar e monitorar processos em segundo plano
- Receba solicitações de autenticação dos processos do cliente e autentique-as antes de permitir que o banco de dados atenda às solicitações

Processos de backend

Se o postmaster autenticar uma solicitação de cliente, o postmaster bifurcará um novo processo de backend, também chamado de processo postgres. Um processo de cliente conecta-se exatamente a um processo de backend. O processo de cliente e o processo de backend se comunicam diretamente sem a intervenção do processo postmaster.

Processos em segundo plano

O processo postmaster bifurca vários processos que realizam diferentes tarefas de backend. Alguns dos mais importantes incluem:

- Gravador WAL

  O Aurora PostgreSQL grava dados no buffer de WAL (gravação antecipada) nos arquivos de log. O princípio do registro em log de gravação antecipada determina que o banco de dados não pode gravar alterações nos arquivos de dados até que o banco de dados grave registros de log descrevendo essas alterações no disco. O mecanismo WAL reduz a E/S do disco e permite que o Aurora PostgreSQL utilize os logs para recuperar o banco de dados após uma falha.

- Gravador em segundo plano

  Esse processo grava periodicamente páginas sujas (modificadas) dos buffers de memória nos arquivos de dados. Uma página fica suja quando um processo de backend a modifica na memória.

- Daemon autovacuum

  O daemon consiste no seguinte:

  - O launcher de autovacuum
  - Os processos de operador de autovacuum

  Quando o autovacuum está ativado, ele procura tabelas que tiveram um grande número de tuplas inseridas, atualizadas ou excluídas. Esse daemon tem as seguintes responsabilidades:

  - Recuperar ou reutilizar o espaço em disco ocupado por linhas atualizadas ou excluídas
  - Atualizar estatísticas utilizadas pelo planejador
  - Proteger contra a perda de dados antigos devido à recorrência de IDs de transação

  O recurso e autovacuum automatiza a execução de comandos `VACUUM` e `ANALYZE`. `VACUUM` tem as seguintes variantes: padrão e completo. O vacuum padrão é executado em paralelo com outras operações de banco de dados. `VACUUM FULL` requer um bloqueio exclusivo na tabela em que está trabalhando. Portanto, ele não pode ser executado em paralelo com operações que acessam a mesma tabela. `VACUUM` cria uma quantidade substancial de tráfego de E/S, podendo piorar a performance para outras sessões ativas.

Eventos de espera do Aurora PostgreSQL

A tabela a seguir lista os eventos de espera do Aurora PostgreSQL que indicam mais comumente problemas de performance e resume as causas e medidas corretivas mais comuns. Os eventos de espera a seguir representam um subconjunto da lista em Eventos de espera do Amazon Aurora PostgreSQL.

Eventos de espera	Definição
Client:ClientRead	Esse evento ocorre quando o Aurora PostgreSQL está aguardando para receber dados do cliente.
Client:ClientWrite	Esse evento ocorre quando o Aurora PostgreSQL está aguardando para gravar dados no cliente.
CPU	Ocorre quando um thread está ativo na CPU ou está aguardando a CPU.
IO:BufFileRead and IO:BufFileWrite	Esses eventos ocorrem quando o Aurora PostgreSQL cria arquivos temporários.
IO:DataFileRead	Esse evento ocorre quando uma conexão aguarda em um processo de backend para ler uma página necessária do armazenamento porque essa página não está disponível na memória compartilhada.
IO:XactSync	Esse evento ocorre quando o banco de dados está aguardando o subsistema de armazenamento do Aurora confirmar uma transação regular ou a reversão de uma transação preparada.
IPC:DamRecordTxAck	Esse evento ocorre quando o Aurora PostgreSQ L em uma sessão que utiliza fluxos de atividade s do banco de dados gera um evento de fluxo de atividades e espera que o evento se torne durável.
Lock:advisory	Esse evento ocorre quando uma aplicação PostgreSQL utiliza um bloqueio para coordenar as atividades em várias sessões.

Eventos de espera	Definição
Lock:extend	Esse evento ocorre quando um processo de backend está aguardando para bloquear uma relação com o objetivo de a estender, enquanto outro processo tem um bloqueio nessa relação para o mesmo objetivo.
Lock:Relation	Esse evento ocorre quando uma consulta está aguardando para adquirir um bloqueio em uma tabela ou visualização que está atualmente bloqueada por outra transação.
Lock:transactionid	Esse evento ocorre quando uma transação está aguardando um bloqueio em nível de linha.
Lock:tuple	Esse evento ocorre quando um processo de backend está aguardando para adquirir um bloqueio em uma tupla.
LWLock:buffer_content (BufferContent)	Esse evento ocorre quando uma sessão aguarda para ler ou gravar uma página de dados na memória enquanto outra sessão fica com a página bloqueada para gravação.
LWLock:buffer_mapping	Esse evento ocorre quando uma sessão está aguardando para associar um bloco de dados a um buffer no grupo de buffer compartilhado.
LWLock:BufferIO (IPC:BufferIO)	Esse evento ocorre quando o Aurora PostgreSQL ou o RDS for PostgreSQL aguarda outros processos terminarem suas operações de entrada/saída (E/S) ao tentarem acessar simultaneamente uma página.

Eventos de espera	Definição
LWLock:lock_manager	Esse evento ocorre quando o mecanismo Aurora PostgreSQL mantém a área de memória do bloqueio compartilhado para alocar, verificar e desalocar um bloqueio nos casos em que um bloqueio de caminho rápido não é possível.
LWLock:MultiXact	Esse tipo de evento ocorre quando o Aurora PostgreSQL mantém uma sessão aberta para concluir várias transações que envolvem a mesma linha em uma tabela. O evento de espera indica qual aspecto do processamento de várias transações está gerando o evento de espera, ou seja, LWLock:MultiXactOffsetSLRU, LWLock:MultiXactOffsetBuffer, LWLock:MultiXactMemberSLRU ou LWLock:MultiXactMemberBuffer.
Tempo limite:PgSleep	Esse evento ocorre quando um processo do servidor chama a função `pg_sleep` e está aguardando o tempo limite de suspensão expirar.

Client:ClientRead

O evento `Client:ClientRead` ocorre quando o Aurora PostgreSQL está aguardando para receber dados do cliente.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações sobre eventos de espera têm suporte para o Aurora PostgreSQL versão 10 e versões superiores.

Contexto

Um cluster de banco de dados Aurora PostgreSQL está aguardando para receber dados do cliente. O cluster de banco de dados Aurora PostgreSQL deve receber os dados do cliente antes de poder enviar mais dados para esse cliente. O tempo de espera do cluster antes de receber dados do cliente é um evento `Client:ClientRead`.

Possíveis causas do maior número de esperas

Causas comuns do surgimento do evento `Client:ClientRead` nas principais esperas incluem:

Maior latência de rede

Pode haver maior latência de rede entre o cluster de banco de dados Aurora PostgreSQL e o cliente. A latência de rede mais alta aumenta o tempo necessário para o cluster de banco de dados receber dados do cliente.

Maior carga no cliente

Pode haver pressão da CPU ou saturação da rede no lado do cliente. Um aumento na carga no cliente pode atrasar a transmissão de dados do cliente para o cluster de banco de dados Aurora PostgreSQL.

Excesso de viagens de ida e volta na rede

Um número elevado de viagens de ida e volta na rede entre o cluster de banco de dados Aurora PostgreSQL e o cliente pode atrasar a transmissão de dados do último para o primeiro.

Operação de cópia extensa

Durante uma operação de cópia, os dados são transferidos do sistema de arquivos do cliente para o cluster de banco de dados Aurora PostgreSQL. O envio de uma muitos dados para o cluster de banco de dados pode atrasar a transmissão de dados do cliente para esse cluster.

Desconexão de um cliente inativo

Uma conexão com uma instância de banco de dados do Aurora PostgreSQL está inativa no estado da transação e está aguardando que um cliente envie mais dados ou emita um comando. Esse estado pode levar a um aumento em eventos `Client:ClientRead`.

PgBouncer utilizado para agrupamento de conexões

PgBouncer tem uma configuração de rede de baixo nível chamada `pkt_buf` e que está definida como 4.096 por padrão. Se a workload estiver enviando pacotes de consulta maiores que 4.096 bytes por meio de PgBouncer, convém aumentar a configuração `pkt_buf` para 8.192. Se a nova configuração não diminuir o número de eventos `Client:ClientRead`, convém aumentar a configuração `pkt_buf` para valores maiores, como 16.384 ou 32.768. Se o texto da consulta for grande, a configuração maior pode ser particularmente útil.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Colocar os clientes na mesma zona de disponibilidade e sub-rede VPC que o cluster
- Escalar seu cliente
- Utilizar instâncias da geração atual
- Aumentar a largura de banda da rede
- Monitorar máximos de performance da rede
- Monitorar transações no estado de “inatividade em transação”

Colocar os clientes na mesma zona de disponibilidade e sub-rede VPC que o cluster

Para reduzir a latência da rede e aumentar a taxa de transferência da rede, coloque clientes na mesma zona de disponibilidade e na mesma sub-rede de nuvem privada virtual (VPC) que o cluster de banco de dados Aurora PostgreSQL. Assegure-se de que os clientes estejam o mais geograficamente próximos do cluster de banco de dados quanto possível.

Escalar seu cliente

Utilizando o Amazon CloudWatch ou outras métricas de host, determine se o cliente está atualmente restrito pela CPU ou pela largura de banda da rede, ou por ambas. Se o cliente estiver restrito, escale-o de acordo.

Utilizar instâncias da geração atual

Em alguns casos, talvez você não esteja utilizando uma classe de instância de banco de dados que ofereça suporte a quadros jumbo. Se estiver executando sua aplicação no Amazon EC2, considere

utilizar uma instância de geração atual para o cliente. Além disso, configure a MTU (unidade de transmissão máxima) no sistema operacional cliente. Essa técnica pode reduzir o número de idas e voltas pela rede e aumentar a taxa de transferência da rede. Consulte mais informações em Frames jumbo (9001 MTU) no Guia do usuário do Amazon EC2.

Para obter informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora. Para determinar a classe de instância de banco de dados equivalente a um tipo de instância do Amazon EC2, coloque `db.` antes do nome do tipo de instância do Amazon EC2. Por exemplo, a instância `r5.8xlarge` do Amazon EC2 é equivalente à classe de instância de banco de dados `db.r5.8xlarge`.

Aumentar a largura de banda da rede

Utilize as métricas `NetworkReceiveThroughput` e `NetworkTransmitThroughput` do Amazon CloudWatch para monitorar o tráfego de rede de entrada e saída no cluster de banco de dados. Essas métricas podem ajudar você a determinar se a largura de banda da rede é suficiente para a sua workload.

Se a largura de banda da rede não suficiente, aumente-a. Se o cliente AWS ou sua instância de banco de dados estiver atingindo os limites de largura de banda da rede, a única maneira de aumentar a largura de banda será ampliar o tamanho da instância de banco de dados.

Para ter mais informações sobre métricas do CloudWatch, consulte Métricas do Amazon CloudWatch para o Amazon Aurora.

Monitorar máximos de performance da rede

Se você utiliza clientes do Amazon EC2, o Amazon EC2 fornece máximos para métricas de performance da rede, incluindo largura de banda de rede agregada de entrada e saída. Ele também fornece rastreamento de conexões para garantir que os pacotes sejam retornados conforme esperado e vinculem localmente o acesso para serviços como o Sistema de Nomes de Domínio (DNS). Para monitorar esses máximos, utilize um driver de rede avançado atual e monitore a performance de rede do seu cliente.

Consulte mais informações em Monitorar a performance de rede de sua instância do EC2 no Guia do usuário do Amazon EC2.

Monitorar transações no estado de “inatividade em transação”

Verifique se você tem um número cada vez maior de conexões `idle in transaction`. Para isso, monitore a coluna `state` na tabela `pg_stat_activity`. Talvez seja possível identificar a origem da conexão executando uma consulta semelhante à seguinte.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

Client:ClientWrite

O evento `Client:ClientWrite` ocorre quando o Aurora PostgreSQL está aguardando para gravar dados no cliente.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações sobre eventos de espera têm suporte para o Aurora PostgreSQL versão 10 e versões superiores.

Contexto

Um processo de cliente deve ler todos os dados recebidos de um cluster de banco de dados Aurora PostgreSQL para que esse cluster possa enviar mais dados. O tempo de espera do cluster antes de enviar mais dados ao cliente é um evento `Client:ClientWrite`.

A taxa de transferência da rede reduzida entre o cluster de banco de dados Aurora PostgreSQL e o cliente pode causar esse evento. A pressão da CPU e a saturação da rede no cliente também podem causar esse evento. Pressão da CPU é quando a CPU está totalmente utilizada e existem tarefas aguardando o tempo da CPU. Saturação de rede é quando a rede entre o banco de dados e o cliente está transportando mais dados do que ela pode manipular.

Possíveis causas do maior número de esperas

Causas comuns do surgimento do evento `Client:ClientWrite` nas principais esperas incluem:

Maior latência de rede

Pode haver maior latência de rede entre o cluster de banco de dados do Aurora PostgreSQL e o cliente. A latência de rede mais alta aumenta o tempo necessário para o cliente receber os dados.

Maior carga no cliente

Pode haver pressão da CPU ou saturação da rede no lado do cliente. Um aumento na carga do cliente atrasa o recebimento de dados do cluster de banco de dados Aurora PostgreSQL.

Grande volume de dados enviados ao cliente

O cluster de banco de dados Aurora PostgreSQL pode estar enviando uma grande quantidade de dados ao cliente. É possível que um cliente não consiga receber os dados tão rápido quanto o cluster os está enviando. Atividades como cópias de tabelas grandes podem resultar no aumento de eventos `Client:ClientWrite`.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Colocar os clientes na mesma zona de disponibilidade e sub-rede VPC que o cluster
- Utilizar instâncias da geração atual
- Reduzir a quantidade de dados enviados ao cliente
- Escalar seu cliente

Colocar os clientes na mesma zona de disponibilidade e sub-rede VPC que o cluster

Para reduzir a latência da rede e aumentar a taxa de transferência da rede, coloque clientes na mesma zona de disponibilidade e na mesma sub-rede de nuvem privada virtual (VPC) que o cluster de banco de dados do Aurora PostgreSQL.

Utilizar instâncias da geração atual

Em alguns casos, talvez você não esteja utilizando uma classe de instância de banco de dados que ofereça suporte a quadros jumbo. Se estiver executando sua aplicação no Amazon EC2, considere

utilizar uma instância de geração atual para o cliente. Além disso, configure a MTU (unidade de transmissão máxima) no sistema operacional cliente. Essa técnica pode reduzir o número de idas e voltas pela rede e aumentar a taxa de transferência da rede. Consulte mais informações em Frames jumbo (9001 MTU) no Guia do usuário do Amazon EC2.

Para obter informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora. Para determinar a classe de instância de banco de dados equivalente a um tipo de instância do Amazon EC2, coloque `db.` antes do nome do tipo de instância do Amazon EC2. Por exemplo, a instância `r5.8xlarge` do Amazon EC2 é equivalente à classe de instância de banco de dados `db.r5.8xlarge`.

Reduzir a quantidade de dados enviados ao cliente

Quando possível, ajuste a aplicação para reduzir a quantidade de dados que o cluster de banco de dados Aurora PostgreSQL envia ao cliente. Fazer esses ajustes alivia a contenção da CPU e da rede no lado do cliente.

Escalar seu cliente

Utilizando o Amazon CloudWatch ou outras métricas de host, determine se o cliente está atualmente restrito pela CPU ou pela largura de banda da rede, ou por ambas. Se o cliente estiver restrito, escale-o de acordo.

CPU

Ocorre quando um thread está ativo na CPU ou está aguardando a CPU.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

As informações sobre eventos de espera são relevantes para o Aurora PostgreSQL versão 9.6 e versões superiores.

Contexto

A unidade de processamento central (CPU) é o componente de um computador que executa instruções. Por exemplo, instruções de CPU realizam operações aritméticas e trocam dados na memória. Se uma consulta aumentar o número de instruções que ela executa por meio do mecanismo de banco de dados, o tempo gasto na execução dessa consulta aumentará. Programação da CPU refere-se a alocar tempo de CPU a um processo. A programação é orquestrada pelo kernel do sistema operacional.

Tópicos

- Como saber quando essa espera ocorre
- Métrica DBLoadCPU
- Métricas os.cpuUtilization
- Provável causa da programação da CPU

Como saber quando essa espera ocorre

Esse evento de espera `CPU` indica que um processo de backend está ativo na CPU ou aguardando a CPU. É possível determinar que isso está ocorrendo quando uma consulta mostra as seguintes informações:

- A coluna `pg_stat_activity.state` tem o valor `active`.
- As colunas `wait_event_type` e `wait_event` em `pg_stat_activity` são ambas `null`.

Para ver os processos de backend que estão utilizando ou aguardando CPU, execute a seguinte consulta.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

Métrica DBLoadCPU

A métrica do Performance Insights para CPU é `DBLoadCPU`. O valor de `DBLoadCPU` pode diferir do valor da métrica `CPUUtilization` do Amazon CloudWatch. A última métrica é coletada do HyperVisor para uma instância de banco de dados.

Métricas os.cpuUtilization

As métricas do Performance Insights para o sistema operacional fornecem informações detalhadas sobre a utilização da CPU. Por exemplo, é possível exibir as seguintes métricas:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

O Performance Insights relata o uso da CPU pelo mecanismo de banco de dados como `os.cpuUtilization.nice.avg`.

Provável causa da programação da CPU

Do ponto de vista do sistema operacional, a CPU está ativa quando não executa o thread ocioso. A CPU está ativa enquanto faz um cálculo, mas também está ativa quando aguarda E/S de memória. Esse tipo de E/S domina uma workload típica de banco de dados.

É provável que os processos aguardem para serem programados em uma CPU quando as seguintes condições forem atendidas:

- A métrica `CPUUtilization` do CloudWatch está próxima dos 100%.
- A carga média é maior que o número de vCPUs, indicando uma carga pesada. Você pode encontrar a métrica `loadAverageMinute` na seção de métricas do sistema operacional do Performance Insights.

Possíveis causas do maior número de esperas

Quando o evento de espera de CPU ocorre mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Tópicos

- Possíveis causas de picos súbitos
- Possíveis causas de alta frequência a longo prazo
- Casos excepcionais

Possíveis causas de picos súbitos

As causas mais prováveis de picos súbitos são as seguintes:

- Sua aplicação abriu muitas conexões simultâneas com o banco de dados. Esse cenário é conhecido como “tempestade de conexões”.
- A workload da sua aplicação foi alterada de qualquer uma das seguintes maneiras:
  - Novas consultas
  - Um aumento no tamanho do conjunto de dados
  - Manutenção ou criação de índices
  - Novas funções
  - Novos operadores
  - Um aumento na execução de consultas paralelas
- Seus planos de execução de consultas foram modificados. Em certos casos, uma alteração pode causar um aumento nos buffers. Por exemplo, a consulta agora está utilizando uma varredura sequencial quando utilizava anteriormente um índice. Nesse caso, ela precisa de mais CPU para atingir o mesmo objetivo.

Possíveis causas de alta frequência a longo prazo

As causas mais prováveis de eventos que se repetem por um longo período são:

- Muitos processos de backend estão em execução simultaneamente na CPU. Esses processos podem ser operadores paralelos.
- Consultas estão sendo executadas com performance abaixo do ideal porque precisam de um grande número de buffers.

Casos excepcionais

Se nenhuma das causas prováveis revelarem ser causas reais, as seguintes situações podem estar ocorrendo:

- A CPU está alternando processos para dentro e para fora.
- A alternância de contexto da CPU aumentou.
- O código do Aurora PostgreSQL não inclui eventos de espera.

Ações

Se o evento de espera CPU domina a atividade do banco de dados, isso não indica necessariamente um problema de performance. Responda a esse evento somente quando a performance diminuir.

Tópicos

- Investigar se o banco de dados está causando o aumento da CPU
- Determinar se o número de conexões aumentou
- Responder a alterações de workload

Investigar se o banco de dados está causando o aumento da CPU

Examine a métrica `os.cpuUtilization.nice.avg` no Performance Insights. Se esse valor for muito menor que o uso da CPU, processos que não são do banco de dados são os principais colaboradores para a CPU.

Determinar se o número de conexões aumentou

Examine a métrica `DatabaseConnections` no Amazon CloudWatch. Sua ação depende se o número aumentou ou diminuiu durante o período de maior número de eventos de espera de CPU.

As conexões aumentaram

Se o número de conexões aumentou, compare o número de processos de backend que consumem CPU com o número de vCPUs. Os cenários a seguir são possíveis:

- O número de processos de backend que consomem CPU é menor que o número de vCPUs.

  Nesse caso, o número de conexões não é um problema. Porém, você ainda pode tentar reduzir a utilização da CPU.
- O número de processos de backend que consomem CPU é maior que o número de vCPUs.

  Nesse caso, considere as opções a seguir:
  - Diminua o número de processos de backend conectados ao banco de dados. Por exemplo, implemente uma solução de agrupamento de conexões, como o RDS Proxy. Para saber mais, consulte Amazon RDS Proxypara Aurora.
  - Atualize o tamanho da sua instância para obter um número maior de vCPUs.
  - Se aplicável, redirecione algumas workloads somente leitura para nós de leitor.

As conexões não aumentaram

Examine as métricas `blks_hit` no Performance Insights. Procure uma correlação entre um aumento em `blks_hit` e o uso da CPU. Os cenários a seguir são possíveis:

- O uso da CPU e `blks_hit` estão correlacionados.

  Nesse caso, encontre as principais instruções SQL vinculadas ao uso da CPU e procure modificações no plano. Você pode usar uma das seguintes técnicas:

  - Explicar os planos manualmente e compare-os com o plano de execução esperado.
  - Procurar um aumento nos acertos de bloco por segundo e nos acertos de blocos locais por segundo. Na seção Top SQL (SQL principal) do painel do Performance Insights, escolha Preferences (Preferências).
- O uso da CPU e `blks_hit` não estão correlacionados.

  Nesse caso, determine se alguma das seguintes situações ocorre:

  - A aplicação está se conectando e se desconectando rapidamente ao/do banco de dados.

    Diagnostique esse comportamento ativando `log_connections` e `log_disconnections` e analisando os logs do PostgreSQL. Considere utilizar o analisador de logs `pgbadger`. Para obter mais informações, consulte https://github.com/darold/pgbadger.
  - O sistema operacional está sobrecarregado.

    Nesse caso, o Performance Insights mostra que processos de backend estão consumindo CPU por mais tempo que o normal. Procure evidências nas métricas `os.cpuUtilization` do Performance Insights ou na métrica `CPUUtilization` do CloudWatch. Se o sistema operacional estiver sobrecarregado, consulte as métricas de monitoramento avançado para aprofundar o diagnóstico. Especificamente, observe a lista de processos e a porcentagem de CPU consumida por cada um.
  - As principais instruções SQL estão consumindo muita CPU.

    Examine instruções que estão vinculadas ao uso da CPU para verificar se elas podem utilizar menos CPU. Execute um comando `EXPLAIN` e concentre-se nos nós do plano que têm o maior impacto. Considere utilizar um visualizador de plano de execução do PostgreSQL. Para experimentar essa ferramenta, consulte http://explain.dalibo.com/.

Responder a alterações de workload

Se a sua workload mudou, procure os seguintes tipos de alterações:

Novas consultas

Verifique se novas consultas são esperadas. Em caso positivo, verifique se os planos de execução dessas consultas e o número de execuções por segundo são os esperados.

Um aumento no tamanho do conjunto de dados

Determine se o particionamento, caso ainda não esteja implementado, pode ajudar. Essa estratégia é capaz de reduzir o número de páginas que uma consulta precisa recuperar.

Manutenção ou criação de índices

Verifique se a programação de manutenção é a esperada. Uma prática recomendada é agendar atividades de manutenção fora das atividades de pico.

Novas funções

Confirme se essas funções são executadas conforme o esperado durante o teste. Especificamente, confira se o número de execuções por segundo é o esperado.

Novos operadores

Verifique se eles funcionam conforme o esperado durante os testes.

Um aumento na execução de consultas paralelas

Determine se alguma das situações a seguir ocorreu:

- As relações ou os índices envolvidos cresceram de repente a ponto de diferirem significativamente de `min_parallel_table_scan_size` ou `min_parallel_index_scan_size`.
- Alterações recentes foram feitas em `parallel_setup_cost` ou `parallel_tuple_cost`.
- Alterações recentes foram feitas em `max_parallel_workers` ou `max_parallel_workers_per_gather`.

IO:BufFileRead and IO:BufFileWrite

Os eventos `IO:BufFileRead` e `IO:BufFileWrite` ocorrem quando o Aurora PostgreSQL cria arquivos temporários. Quando as operações requerem mais memória do que os parâmetros de

memória de trabalho definidos atualmente, elas gravam dados temporários no armazenamento persistente. Essa operação é chamada às vezes de derramamento no disco. Para obter mais informações sobre os arquivos temporários e o uso, consulte Gerenciar arquivos temporários com o PostgreSQL.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

`IO:BufFileRead` e `IO:BufFileWrite` estão relacionados à área de memória de trabalho e a área de memória de trabalho de manutenção. Para saber mais sobre essas áreas de memória local, consulte Área de memória de trabalho e Área de memória de trabalho para manutenção.

O valor padrão para `work_mem` é 4 MB. Se uma sessão executar operações em paralelo, cada operador que lidar com o paralelismo usará 4 MB de memória. Por essa razão, defina `work_mem` com cautela. Se você aumentar demais esse valor, um banco de dados que execute muitas sessões poderá consumir muita memória. Se você definir um valor muito baixo, o Aurora PostgreSQL criará arquivos temporários no armazenamento local. A E/S de disco desses arquivos temporários pode reduzir a performance.

Se você observar a seguinte sequência de eventos, é possível que seu banco de dados esteja gerando arquivos temporários:

1. Redução súbita e acentuada na disponibilidade
2. Recuperação rápida para o espaço livre

Você também pode observar um padrão de “motosserra”. Esse padrão pode indicar que o banco de dados está criando arquivos pequenos constantemente.

Possíveis causas do maior número de esperas

Em geral, esses eventos de espera são causados por operações que consomem mais memória do que é alocado pelos parâmetros `work_mem` ou `maintenance_work_mem`. Para compensar isso, as operações gravam em arquivos temporários. Causas comuns dos eventos `IO:BufFileRead` e `IO:BufFileWrite` incluem:

Consultas que necessitam de mais memória do que existe na área de memória de trabalho

Consultas com as seguintes características utilizam a área de memória de trabalho:

- Junções de hash
- `ORDER BY`Cláusula
- `GROUP BY`Cláusula
- `DISTINCT`
- Funções de janela
- `CREATE TABLE AS SELECT`
- Atualização de visualizações materializadas

Instruções que necessitam de mais memória do que existe na área de memória do trabalho de manutenção

As seguintes instruções usam a área de memória do trabalho de manutenção:

- `CREATE INDEX`
- `CLUSTER`

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Identificar o problema
- Examinar suas consultas de junção
- Examinar suas consultas ORDER BY e GROUP BY
- Evite utilizar a operação DISTINCT
- Considere utilizar funções de janela em vez de funções GROUP BY

- Investigar visualizações materializadas e instruções CTAS
- Usar pg_repack ao criar índices
- Aumentar maintenance_work_mem ao agrupar tabelas
- Ajustar a memória para evitar IO:BufFileRead e IO:BufFileWrite

Identificar o problema

Você pode visualizar o uso de arquivos temporários diretamente no Insights de Performance. Para obter mais informações, consulte Visualizar o uso de arquivos temporários com o Insights de Performance. Quando o Insights de Performance está desabilitado, é possível observar um aumento nas operações `IO:BufFileRead` e `IO:BufFileWrite`. Para solucionar problemas, faça o seguinte:

1. Examine a métrica `FreeLocalStorage` no Amazon CloudWatch.
2. Observe se há um padrão de motosserra, ou seja, uma série de picos irregulares.

Um padrão de motosserra indica um rápido consumo e liberação de armazenamento, muitas vezes associados a arquivos temporários. Se você perceber esse padrão, ative o Performance Insights. Ao utilizar o Performance Insights, é possível identificar quando os eventos de espera ocorrem e quais consultas estão associadas a eles. A solução depende da consulta específica que está causando os eventos.

Ou defina o parâmetro `log_temp_files`. Esse parâmetro registra todas as consultas que estão gerando mais do que o limite de KB de arquivos temporários. Se o valor for `0`, o Aurora PostgreSQL registrará todos os arquivos temporários. Se o valor for `1024`, o Aurora PostgreSQL registrará todas as consultas que geram arquivos temporários maiores que 1 MB. Para obter mais informações sobre `log_temp_files`, consulte o tópico sobre Relatórios de erros e registro em log, na documentação do PostgreSQL.

Examinar suas consultas de junção

Sua aplicação provavelmente utiliza junções. Por exemplo, a consulta a seguir une quatro tabelas.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
```

```
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

Uma causa possível dos picos no uso temporário de arquivos é um problema na própria consulta. Por exemplo, uma cláusula quebrada talvez não esteja filtrando as junções corretamente. Considere a segunda junção interna no exemplo a seguir.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

A consulta anterior junta `customer.id` com `customer.id` por engano, gerando um produto cartesiano entre cada cliente e cada pedido. Esse tipo de junção acidental gera arquivos temporários grandes. Dependendo do tamanho das tabelas, uma consulta cartesiana pode até mesmo lotar o armazenamento. Sua aplicação pode ter junções cartesianas quando as seguintes condições são atendidas:

- Você percebe reduções grandes e acentuadas na disponibilidade do armazenamento, seguidas de uma rápida recuperação.
- Nenhum índice está sendo criado.
- Nenhuma instrução `CREATE TABLE FROM SELECT` está sendo emitida.
- Nenhuma visualização materializada está sendo atualizada.

Para verificar se as tabelas estão sendo unidas utilizando as chaves apropriadas, inspecione suas diretivas de mapeamento de consultas e objetos relacionais. Lembre-se de que certas consultas da sua aplicação não são chamadas o tempo todo e que algumas consultas são geradas dinamicamente.

Examinar suas consultas ORDER BY e GROUP BY

Em alguns casos, uma cláusula `ORDER BY` pode resultar no excesso de arquivos temporários. Considere as seguintes diretrizes:

- Inclua somente colunas em uma cláusula `ORDER BY` quando elas precisarem ser ordenadas. Essa orientação é especialmente importante para consultas que retornam milhares de linhas e especificam muitas colunas na cláusula `ORDER BY`.
- Considere criar índices para acelerar cláusulas `ORDER BY` quando elas correspondem a colunas que tenham a mesma ordem crescente ou decrescente. Índices parciais são preferíveis, pois são menores. Índices menores são lidos e percorridos com mais rapidez.
- Se você criar índices para colunas que podem aceitar valores nulos, considere se deseja que esses valores nulos sejam armazenados no final ou no início dos índices.

  Se possível, reduza o número de linhas que precisam ser ordenadas, filtrando o conjunto de resultados. Se você usar instruções de cláusula `WITH` ou subconsultas, lembre-se de que uma consulta interna gera um conjunto de resultados e o transmite à consulta externa. Quanto mais linhas uma consulta puder remover, menos ordenação ela precisará fazer.
- Se não precisar obter o conjunto completo de resultados, utilize a cláusula `LIMIT`. Por exemplo, se quiser apenas as cinco principais linhas, uma consulta utilizando a cláusula `LIMIT` não continuará gerando resultados. Dessa forma, essa consulta requer menos memória e arquivos temporários.

Uma consulta que usa uma cláusula `GROUP BY` também pode exigir arquivos temporários. Consultas `GROUP BY` resumem valores utilizando funções como as seguintes:

- `COUNT`
- `AVG`
- `MIN`
- `MAX`
- `SUM`
- `STDDEV`

Para ajustar consultas `GROUP BY`, siga as recomendações para consultas `ORDER BY`.

Evite utilizar a operação DISTINCT

Se possível, evite utilizar a operação `DISTINCT` para remover linhas duplicadas. Quanto mais linhas desnecessárias e duplicadas sua consulta retornar, mais cara a operação `DISTINCT` se tornará. Se possível, adicione filtros à cláusula `WHERE` mesmo que você utilize os mesmos filtros para tabelas diferentes. Filtrar a consulta e a junção corretamente melhora a performance e reduz o uso de recursos. Isso também evita relatórios e resultados incorretos.

Se precisar usar `DISTINCT` para várias linhas de uma mesma tabela, considere criar um índice composto. O agrupamento de várias colunas em um índice pode melhorar o tempo para avaliar linhas distintas. Além disso, se utilizar o Amazon Aurora PostgreSQL versão 10 ou superior, você poderá correlacionar estatísticas entre várias colunas utilizando o comando `CREATE STATISTICS`.

Considere utilizar funções de janela em vez de funções GROUP BY

Usando `GROUP BY`, você altera o conjunto de resultados e, em seguida, recupera o resultado agregado. Usando funções de janela, você agrega dados sem modificar o conjunto de resultados. Uma função de janela usa a cláusula `OVER` para fazer cálculos entre os conjuntos definidos pela consulta, correlacionando uma linha com outra. Você pode utilizar todas as funções `GROUP BY` em funções de janela, mas também utilizar funções como as seguintes:

- `RANK`
- `ARRAY_AGG`
- `ROW_NUMBER`
- `LAG`
- `LEAD`

Para minimizar o número de arquivos temporários gerados por uma função de janela, remova duplicatas do mesmo conjunto de resultados quando precisar de duas agregações distintas. Considere a seguinte consulta.

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

Você pode reescrever essa consulta com a cláusula `WINDOW` da seguinte maneira.

```
SELECT sum(salary) OVER w as sum_salary
```

```
        , avg(salary) OVER w as_avg_salary
    FROM empsalary
WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

Por padrão, o planejador de execução do Aurora PostgreSQL consolida nós semelhantes para que ele não duplique operações. No entanto, utilizando uma declaração explícita para o bloco de janelas, é possível manter a consulta com mais facilidade. Também é possível melhorar a performance ao evitar a duplicação.

Investigar visualizações materializadas e instruções CTAS

Quando uma visualização materializada é atualizada, ela executa uma consulta. Essa consulta pode conter uma operação como `GROUP BY`, `ORDER BY` ou `DISTINCT`. Durante uma atualização, é possível observar um grande número de arquivos temporários e os eventos de espera `IO:BufFileWrite` e `IO:BufFileRead`. Da mesma forma, quando você cria uma tabela com base em uma instrução `SELECT`, a instrução `CREATE TABLE` executa uma consulta. Para reduzir os arquivos temporários necessários, otimize a consulta.

Usar pg_repack ao criar índices

Quando você cria um índice, o mecanismo ordena o conjunto de resultados. À medida que o tamanho das tabelas aumenta e à medida que os valores na coluna indexada se tornam mais diversificados, os arquivos temporários exigem mais espaço. Na maioria dos casos, não é possível impedir a criação de arquivos temporários para tabelas grandes sem modificar a área de memória do trabalho de manutenção. Para obter mais informações, consulte Área de memória de trabalho para manutenção.

Uma possível solução alternativa ao recriar um índice grande é utilizar a ferramenta pg_repack. Para obter mais informações, consulte o tópico sobre como Reorganizar tabelas em bancos de dados PostgreSQL com bloqueios mínimos, na documentação de pg_repack.

Aumentar maintenance_work_mem ao agrupar tabelas

O comando `CLUSTER` agrupa a tabela especificada por table_name com base em um índice existente especificado por index_name. O Aurora PostgreSQL recria fisicamente essa tabela para corresponder à ordem de um determinado índice.

Quando o armazenamento magnético era predominante, o agrupamento era comum, pois a taxa de transferência de armazenamento era limitada. Agora que o armazenamento baseado em SSD é comum, o agrupamento tornou-se menos popular. No entanto, se você agrupar tabelas, ainda

poderá aumentar a performance ligeiramente, dependendo do tamanho da tabela, do índice, da consulta e assim por diante.

Se você executar o comando `CLUSTER` e observar os eventos de espera `IO:BufFileWrite` e `IO:BufFileRead`, ajuste `maintenance_work_mem`. Aumente o tamanho da memória para uma quantidade relativamente grande. Um valor alto significa que o mecanismo pode utilizar mais memória para a operação de agrupamento.

Ajustar a memória para evitar IO:BufFileRead e IO:BufFileWrite

Em uma determinada situação, você precisa ajustar a memória. A meta é equilibrar os seguintes requisitos:

- O valor de `work_mem` (consulte Área de memória de trabalho)
- A memória restante após descontar o valor de `shared_buffers` (consulte Grupo de buffer)
- As conexões máximas abertas e em uso, o que é limitado por `max_connections`

Aumentar o tamanho da área de memória de trabalho

Em algumas situações, a única opção é aumentar a memória utilizada pela sessão. Se as consultas estiverem gravadas corretamente e utilizando as chaves corretas para junções, considere aumentar o valor de `work_mem`. Para obter mais informações, consulte Área de memória de trabalho.

Para descobrir quantos arquivos temporários são gerados por uma consulta, defina `log_temp_files` como `0`. Se você aumentar o valor de `work_mem` para o valor máximo identificado nos logs, impedirá que a consulta gere arquivos temporários. No entanto, `work_mem` define o máximo por nó de plano para cada conexão ou operador paralelo. Se o banco de dados tiver 5.000 conexões e cada uma utilizar 256 MiB de memória, o mecanismo precisará de 1,2 TiB de RAM. Portanto, sua instância pode ficar sem memória.

Reservar memória suficiente para o grupo de buffer compartilhado

Seu banco de dados usa áreas de memória, como o grupo de buffer compartilhado, e não apenas a área de memória de trabalho. Considere os requisitos dessas áreas de memória adicionais antes de aumentar `work_mem`. Para obter mais informações sobre o grupo de buffer, consulte Grupo de buffer.

Por exemplo, suponha que a sua classe de instância do Aurora PostgreSQL seja db.r5.2xlarge. Essa classe tem 64 GiB de memória. Por padrão, 75% da memória são reservados para o grupo de

buffer compartilhado. Depois de subtrair a quantidade alocada à área de memória compartilhada, permanecem 16.384 MB. Não aloque a memória restante exclusivamente à área de memória de trabalho, pois o sistema operacional e o mecanismo também precisam de memória.

A memória que é possível alocar a `work_mem` depende da classe da instância. Se você utilizar uma classe de instância maior, mais memória estará disponível. No entanto, no exemplo anterior, não é possível utilizar mais de 16 GiB. Caso contrário, sua instância estará indisponível quando ficar sem memória. Para recuperar a instância e retirá-la do estado indisponível, os serviços de automação do Aurora PostgreSQL são reiniciados automaticamente.

Gerenciar o número de conexões

Imagine que a sua instância de banco de dados tenha 5.000 conexões simultâneas. Cada conexão usa pelo menos 4 MiB de `work_mem`. O alto consumo de memória das conexões provavelmente diminuirá a performance. Em resposta, existem as seguintes opções:

- Faça upgrade para uma classe de instância maior.
- Diminua o número de conexões de banco de dados simultâneas utilizando um proxy de conexão ou pooler.

Para proxies, considere o Amazon RDS Proxy, o pgBouncer ou um pooler de conexão baseado na sua aplicação. Essa solução alivia a carga da CPU. Ela também reduz o risco quando todas as conexões exigem a área de memória de trabalho. Quando há menos conexões de banco de dados, é possível aumentar o valor de `work_mem`. Dessa forma, você reduz a ocorrência dos eventos de espera `IO:BufFileRead` e `IO:BufFileWrite`. Além disso, as consultas que aguardam a área de memória de trabalho são aceleradas significativamente.

IO:DataFileRead

O evento `IO:DataFileRead` ocorre quando uma conexão aguarda em um processo de backend para ler uma página necessária do armazenamento porque essa página não está disponível na memória compartilhada.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

Todas as consultas e operações de manipulação de dados (DML) acessam páginas no grupo de buffer. Instruções que podem induzir leituras incluem `SELECT`, `UPDATE` e `DELETE`. Por exemplo, um `UPDATE` pode ler páginas de tabelas ou índices. Se a página que está sendo solicitada ou atualizada não estiver no grupo de buffer compartilhado, essa leitura poderá gerar o evento `IO:DataFileRead`.

Como o grupo de buffer compartilhado é finito, ele pode ficar lotado. Nesse caso, solicitações de páginas que não estão na memória forçam o banco de dados a ler blocos do disco. Se o evento `IO:DataFileRead` ocorre com frequência, o grupo de buffer compartilhado pode ser pequeno demais para acomodar sua workload. Esse problema é grave para consultas `SELECT` que fazem a leitura de um grande número de linhas que não cabem no grupo de buffer. Para obter mais informações sobre o grupo de buffer, consulte Grupo de buffer.

Possíveis causas do maior número de esperas

As causas comuns do evento `IO:DataFileRead` incluem:

Picos de conexão

Você pode encontrar várias conexões gerando o mesmo número de eventos de espera IO:DatafileRead. Nesse caso, pode ocorrer um pico (aumento súbito e grande) em eventos `IO:DataFileRead`.

Instruções SELECT e DML realizando varreduras sequenciais

Sua aplicação pode estar executando uma nova operação. Ou uma operação existente pode mudar por conta de um novo plano de execução. Nesses casos, procure tabelas (particularmente grandes) que tenham um valor de `seq_scan` maior. Encontre-as consultando `pg_stat_user_tables`. Para rastrear consultas que estão gerando mais operações de leitura, utilize a extensão `pg_stat_statements`.

CTAS e CREATE INDEX para conjuntos de dados grandes

Um CTAS é uma instrução `CREATE TABLE AS SELECT`. Se você executar um CTAS utilizando um conjunto de dados grande como fonte ou criar um índice em uma tabela grande, o evento `IO:DataFileRead` poderá ocorrer. Quando você cria um índice, talvez o banco de dados

precise ler o objeto inteiro utilizando uma varredura sequencial. Um CTAS gera leituras de `IO:DataFile` quando as páginas não estão na memória.

Vários operadores de vacuum em execução ao mesmo tempo

Operadores de vacuum podem ser acionados manual ou automaticamente. Convém adotar uma estratégia de vacuum agressiva. No entanto, quando uma tabela possui muitas linhas atualizadas ou excluídas, as esperas de `IO:DataFileRead` aumentam. Depois que o espaço é recuperado, o tempo de vacuum gasto em `IO:DataFileRead` diminui.

Ingestão de grandes quantidades de dados

Quando a aplicação ingere grandes quantidades de dados, operações `ANALYZE` podem ocorrer com mais frequência. O processo `ANALYZE` pode ser acionado por um launcher de autovacuum ou chamado manualmente.

A operação `ANALYZE` lê um subconjunto da tabela. O número de páginas que devem ser varridas é calculado multiplicando-se 30 pelo valor de `default_statistics_target`. Para obter mais informações, consulte a Documentação do PostgreSQL. O parâmetro `default_statistics_target` aceita valores entre 1 e 10.000, em que o padrão é 100.

Inanição de recursos

Se a largura de banda ou a CPU da rede da instância forem consumidas, o evento `IO:DataFileRead` poderá ocorrer com mais frequência.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Verificar filtros de predicados em busca de consultas que geram esperas
- Minimizar o efeito de operações de manutenção
- Responder a um alto número de conexões

Verificar filtros de predicados em busca de consultas que geram esperas

Suponha que você identifique consultas específicas que estão gerando eventos de espera `IO:DataFileRead`. É possível identificá-los utilizando as seguintes técnicas:

- Insights de Performance

- Visualizações de catálogo, como a fornecida pela extensão `pg_stat_statements`
- A visualização de catálogo `pg_stat_all_tables`, se ela mostrar periodicamente um número mais alto de leituras físicas
- A visualização `pg_statio_all_tables`, se ela mostrar que os contadores de `_read` estão aumentando

Convém determinar quais filtros são utilizados no predicado (cláusula `WHERE`) dessas consultas. Siga estas diretrizes:

- Execute o comando `EXPLAIN`. Na saída, identifique quais tipos de varreduras são utilizados. Uma varredura sequencial não indica necessariamente um problema. Consultas que utilizam varreduras sequenciais naturalmente produzem mais eventos `IO:DataFileRead` quando comparados a consultas que utilizam filtros.

  Descubra se a coluna listada na cláusula `WHERE` é indexada. Caso contrário, considere criar um índice para essa coluna. Essa abordagem evita varreduras sequenciais e reduz eventos `IO:DataFileRead`. Se uma consulta tiver filtros restritivos e ainda produzir varreduras sequenciais, avalie se os índices adequados estão sendo utilizados.
- Descubra se a consulta está acessando uma tabela muito grande. Em alguns casos, o particionamento de uma tabela pode melhorar a performance, permitindo que a consulta leia apenas as partições necessárias.
- Observe a cardinalidade (número total de linhas) das suas operações de junção. Observe o quão restritivos são os valores que você está transmitindo nos filtros para a sua cláusula `WHERE`. Se possível, ajuste sua consulta para reduzir o número de linhas que são transmitidas em cada etapa do plano.

Minimizar o efeito de operações de manutenção

Operações de manutenção, como `VACUUM` e `ANALYZE`, são importantes. Recomendamos que você não as desative ao encontrar eventos de espera `IO:DataFileRead` relacionados a essas operações de manutenção. As abordagens a seguir podem minimizar o efeito dessas operações:

- Execute operações de manutenção manualmente fora do horário de pico. Essa técnica impede que o banco de dados atinja o limite para operações automáticas.
- Considere particionar tabelas muito grandes. Essa técnica reduz a sobrecarga das operações de manutenção. O banco de dados somente acessa as partições que exigem manutenção.

- Ao ingerir grandes quantidades de dados, considere desativar o recurso de análise automática.

O recurso de autovacuum é acionado automaticamente para uma tabela quando a seguinte fórmula é verdadeira.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

A visualização `pg_stat_user_tables` e o catálogo `pg_class` têm várias linhas. Uma linha pode corresponder a uma linha na sua tabela. Essa fórmula pressupõe que os `reltuples` sejam para uma tabela específica. Os parâmetros `autovacuum_vacuum_scale_factor` (0,20 por padrão) e `autovacuum_vacuum_threshold` (50 tuplas por padrão) geralmente são definidos globalmente para toda a instância. Porém, é possível definir valores diferentes para uma tabela específica.

Tópicos

- Localizar tabelas que consomem espaço desnecessariamente
- Localizar índices que consomem espaço desnecessário
- Localizar tabelas elegíveis qualificadas para receber autovacuum

Localizar tabelas que consomem espaço desnecessariamente

Para localizar tabelas que consomem mais espaço do que o necessário, execute a consulta a seguir. Quando essa consulta é executada por uma função de usuário do banco de dados que não tenha a função `rds_superuser`, ela retorna informações somente sobre as tabelas que a função de usuário tem permissão para ler. Essa consulta é compatível com o PostgreSQL versão 12 e versões posteriores.

```
WITH report AS (
   SELECT   schemaname
            ,tblname
            ,n_dead_tup
            ,n_live_tup
            ,block_size*tblpages AS real_size
            ,(tblpages-est_tblpages)*block_size AS extra_size
            ,CASE WHEN tblpages - est_tblpages > 0
               THEN 100 * (tblpages - est_tblpages)/tblpages::float
               ELSE 0
```

```
            END AS extra_ratio, fillfactor, (tblpages-est_tblpages_ff)*block_size AS
 bloat_size
           ,CASE WHEN tblpages - est_tblpages_ff > 0
              THEN 100 * (tblpages - est_tblpages_ff)/tblpages::float
              ELSE 0
            END AS bloat_ratio
           ,is_na
    FROM (
           SELECT  ceil( reltuples / ( (block_size-page_hdr)/tpl_size ) ) +
 ceil( toasttuples / 4 ) AS est_tblpages
                  ,ceil( reltuples / ( (block_size-page_hdr)*fillfactor/
(tpl_size*100) ) ) + ceil( toasttuples / 4 ) AS est_tblpages_ff
                  ,tblpages
                  ,fillfactor
                  ,block_size
                  ,tblid
                  ,schemaname
                  ,tblname
                  ,n_dead_tup
                  ,n_live_tup
                  ,heappages
                  ,toastpages
                  ,is_na
             FROM (
                    SELECT ( 4 + tpl_hdr_size + tpl_data_size + (2*ma)
                               - CASE WHEN tpl_hdr_size%ma = 0 THEN ma ELSE
 tpl_hdr_size%ma END
                               - CASE WHEN ceil(tpl_data_size)::int%ma = 0 THEN ma ELSE
 ceil(tpl_data_size)::int%ma END
                           ) AS tpl_size
                           ,block_size - page_hdr AS size_per_block
                           ,(heappages + toastpages) AS tblpages
                           ,heappages
                           ,toastpages
                           ,reltuples
                           ,toasttuples
                           ,block_size
                           ,page_hdr
                           ,tblid
                           ,schemaname
                           ,tblname
                           ,fillfactor
                           ,is_na
                           ,n_dead_tup
```

```
                            ,n_live_tup
                           FROM (
                                 SELECT  tbl.oid                       AS tblid
                                       ,ns.nspname                     AS schemaname
                                       ,tbl.relname                    AS tblname
                                       ,tbl.reltuples                  AS reltuples
                                       ,tbl.relpages                   AS heappages
                                       ,coalesce(toast.relpages, 0)    AS toastpages
                                       ,coalesce(toast.reltuples, 0)   AS toasttuples
                                       ,psat.n_dead_tup                AS n_dead_tup
                                       ,psat.n_live_tup                AS n_live_tup
                                       ,24                             AS page_hdr
                                       ,current_setting('block_size')::numeric AS
 block_size

 ,coalesce(substring( array_to_string(tbl.reloptions, ' ') FROM
 'fillfactor=([0-9]+)')::smallint, 100) AS fillfactor
                                       ,CASE WHEN version()~'mingw32' OR version()~'64-
bit|x86_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END        AS ma
                                       ,23 + CASE WHEN MAX(coalesce(null_frac,0)) > 0
 THEN ( 7 + count(*) ) / 8 ELSE 0::int END              AS tpl_hdr_size
                                       ,sum( (1-coalesce(s.null_frac, 0)) *
 coalesce(s.avg_width, 1024) )                                  AS tpl_data_size
                                       ,bool_or(att.atttypid =
 'pg_catalog.name'::regtype) OR count(att.attname) <> count(s.attname)         AS is_na
                                   FROM  pg_attribute       AS att
                                   JOIN  pg_class           AS tbl    ON (att.attrelid =
 tbl.oid)
                                   JOIN  pg_stat_all_tables AS psat   ON (tbl.oid =
 psat.relid)
                                   JOIN  pg_namespace       AS ns     ON (ns.oid =
 tbl.relnamespace)
                                LEFT JOIN  pg_stats         AS s      ON
 (s.schemaname=ns.nspname AND s.tablename = tbl.relname AND s.inherited=false AND
 s.attname=att.attname)
                                LEFT JOIN  pg_class         AS toast  ON
 (tbl.reltoastrelid = toast.oid)
                                    WHERE  att.attnum > 0
                                      AND  NOT att.attisdropped
                                      AND  tbl.relkind = 'r'
                                 GROUP BY  tbl.oid, ns.nspname, tbl.relname,
 tbl.reltuples, tbl.relpages, toastpages, toasttuples, fillfactor, block_size, ma,
 n_dead_tup, n_live_tup
                                 ORDER BY  schemaname, tblname
```

```
                            ) AS s
                ) AS s2
       ) AS s3
 ORDER BY bloat_size DESC
)
  SELECT *
    FROM report
   WHERE bloat_ratio != 0
 -- AND schemaname = 'public'
 -- AND tblname = 'pgbench_accounts'
;

-- WHERE NOT is_na
--   AND tblpages*((pst).free_percent + (pst).dead_tuple_percent)::float4/100 >= 1
```

É possível verificar se há sobrecarga na tabela e no índice em sua aplicação. Para obter mais informações, consulte Diagnosticar a sobrecarga na tabela e no índice.

Localizar índices que consomem espaço desnecessário

Para localizar índices que consomem espaço desnecessário, execute a seguinte consulta.

```
-- WARNING: run with a nonsuperuser role, the query inspects
-- only indexes on tables you have permissions to read.
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
```

```
      ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
 MAXALIGN
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
            ELSE nulldatawidth::integer%maxalign
          END
      )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
      -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
    FROM (
      SELECT
        i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
 AS table_oid,
        current_setting('block_size')::numeric AS bs, fillfactor,
        CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
          WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
 THEN 8
          ELSE 4
        END AS maxalign,
        /* per page header, fixed size: 20 for 7.X, 24 for others */
        24 AS pagehdr,
        /* per page btree opaque data */
        16 AS pageopqdata,
        /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
        CASE WHEN max(coalesce(s.null_frac,0)) = 0
          THEN 2 -- IndexTupleData size
          ELSE 2 + (( 32 + 8 - 1 ) / 8)
```

```
          -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
 index + 8 - 1 /8)
       END AS index_tuple_hdr_bm,
       /* data len: we remove null values save space using it fractionnal part from
 stats */
       sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
 nulldatawidth,
       max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
 AS is_na
     FROM pg_attribute AS a
       JOIN (
         SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
           idx.reltuples, idx.relpages, idx.relam,
           indrelid, indexrelid, indkey::smallint[] AS attnum,
           coalesce(substring(
             array_to_string(idx.reloptions, ' ')
              from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
         FROM pg_index
           JOIN pg_class idx ON idx.oid=pg_index.indexrelid
           JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
           JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
         WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
       ) AS i ON a.attrelid = i.indexrelid
       JOIN pg_stats AS s ON s.schemaname = i.nspname
         AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
         -- stats from tbl
         OR  (s.tablename = i.idxname AND s.attname = a.attname))
         -- stats from functionnal cols
       JOIN pg_type AS t ON a.atttypid = t.oid
     WHERE a.attnum > 0
     GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
   ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Localizar tabelas elegíveis qualificadas para receber autovacuum

Para localizar tabelas qualificadas para receber autovacuum, execute a seguinte consulta.

```
--This query shows tables that need vacuuming and are eligible candidates.
```

```
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
        coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
        as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
```

```
ORDER BY age(relfrozenxid) DESC;
```

Responder a um alto número de conexões

Ao monitorar o Amazon CloudWatch, você pode descobrir que a métrica `DatabaseConnections` atinge picos. Esse aumento indica um número maior de conexões com o seu banco de dados. Recomendamos a seguinte abordagem:

- Limite o número de conexões que a aplicação pode abrir com cada instância. Se a aplicação tiver um recurso de grupo de conexões incorporado, defina um número razoável de conexões. Baseie o número no que as vCPUs na instância podem paralelizar de maneira eficiente.

  Se a aplicação não utilizar um recurso de grupo de conexões, considere utilizar o Amazon RDS Proxy ou uma alternativa. Essa abordagem permite que a aplicação abra várias conexões com o balanceador de carga. O balanceador pode então abrir um número restrito de conexões com o banco de dados. À medida que menos conexões são executadas em paralelo, sua instância de banco de dados realiza menos alternâncias de contexto no kernel. As consultas devem progredir com mais rapidez, resultando em menos eventos de espera. Para obter mais informações, consulte Amazon RDS Proxypara Aurora.
- Sempre que possível, aproveite nós de leitura para o Aurora PostgreSQL e réplicas de leitura do RDS for PostgreSQL. Quando a aplicação executar uma operação somente leitura, envie essas solicitações ao endpoint somente leitura. Essa técnica dispersa as solicitações de aplicações em todos os nós de leitor, reduzindo a pressão de E/S no nó de gravador.
- Considere aumentar a escala vertical da sua instância de banco de dados. Uma classe de instância com maior capacidade fornece mais memória, o que dá ao Aurora PostgreSQL um grupo de buffer compartilhado maior para conter páginas. O tamanho maior também dá à instância de banco de dados mais vCPUs para lidar com conexões. Mais vCPUs são particularmente úteis quando as operações que estão gerando eventos de espera `IO:DataFileRead` são gravações.

IO:XactSync

O evento `IO:XactSync` ocorre quando o banco de dados está aguardando o subsistema de armazenamento do Aurora confirmar uma transação regular ou a reversão de uma transação preparada. Uma transação preparada faz parte do suporte do PostgreSQL para uma confirmação em duas fases. Esse evento também pode ocorrer quando uma consulta está aguardando a confirmação de outra transação, principalmente nos casos em que a confirmação automática está desativada.

Nesses cenários, pode parecer que as atualizações estão aguardando XactSync, mesmo que ainda não tenham sido confirmadas.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

O evento `IO:XactSync` indica que a instância está gastando tempo à espera do subsistema de armazenamento do Aurora para confirmar que os dados da transação foram processados.

Possíveis causas do maior número de esperas

Quando o evento `IO:XactSync` aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Saturação da rede

O tráfego entre os clientes e a instância de banco de dados ou o tráfego para o subsistema de armazenamento pode ser muito pesado para a largura de banda da rede.

Pressão da CPU

Uma workload pesada pode estar impedindo que o daemon de armazenamento do Aurora obtenha tempo suficiente de CPU.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Monitorar seus recursos
- Aumentar a escala da CPU verticalmente
- Aumentar a largura de banda da rede
- Reduza o número de confirmações

Monitorar seus recursos

Para determinar a causa do aumento de eventos `IO:XactSync`, verifique as seguintes métricas:

- `WriteThroughput` e `CommitThroughput`: mudanças na taxa de transferência de gravação ou na taxa de transferência de confirmação podem indicar um aumento na workload.
- `WriteLatency` e `CommitLatency`: mudanças na latência de gravação ou latência de confirmação podem indicar que o subsistema de armazenamento está sendo solicitado a trabalhar mais.
- `CPUUtilization`: se a utilização da CPU da instância estiver acima de 90%, o daemon de armazenamento do Aurora talvez não esteja obtendo tempo suficiente na CPU. Nesse caso, a performance da E/S diminuirá.

Para obter informações sobre essas métricas, consulte Métricas no nível da instância do Amazon Aurora.

Aumentar a escala da CPU verticalmente

Para solucionar problemas de falta de CPU, considere mudar para um tipo de instância com mais capacidade de CPU. Para obter informações sobre a capacidade de CPU de uma classe de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

Aumentar a largura de banda da rede

Para determinar se a instância está atingindo seus limites de largura de banda de rede, verifique os seguintes eventos de espera:

- `IO:DataFileRead`, `IO:BufferRead`, `IO:BufferWrite` e `IO:XactWrite`: consultas que utilizam grandes quantidades de E/S podem gerar mais desses eventos de espera.
- `Client:ClientRead` e `Client:ClientWrite`: consultas com grandes quantidades de comunicação com o cliente podem gerar mais desses eventos de espera.

Se a largura de banda da rede for um problema, considere mudar para um tipo de instância com maior largura de banda de rede. Para obter informações sobre a performance de rede de uma classe de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

Reduza o número de confirmações

Para reduzir o número de confirmações, combine instruções em blocos de transações.

IPC:DamRecordTxAck

O evento `IPC:DamRecordTxAck` ocorre quando o Aurora PostgreSQL em uma sessão que utiliza fluxos de atividades do banco de dados gera um evento de fluxo de atividades e espera que o evento se torne durável.

Tópicos

- Versões de mecanismos relevantes
- Contexto
- Causas
- Ações

Versões de mecanismos relevantes

Essas informações de evento de espera são relevantes para o Aurora PostgreSQL 10.7 e todas as versões 10 superiores, a versão 11.4 e todas as versões 11 e todas as versões 12 e 13.

Contexto

No modo síncrono, a durabilidade dos eventos de fluxos de atividades é favorecida em relação à performance do banco de dados. Ao aguardar uma gravação duradoura do evento, a sessão bloqueia outras atividades do banco de dados, causando o evento de espera `IPC:DamRecordTxAck`.

Causas

A causa mais comum do surgimento do evento `IPC:DamRecordTxAck` nas principais esperas é que o recurso Database Activity Streams (DAS) é uma auditoria holística. A atividade SQL mais elevada gera eventos de fluxo de atividades que precisam ser registrados.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera:

- Reduza o número de instruções SQL ou desative fluxos de atividades de banco de dados. Fazer isso reduz o número de eventos que exigem gravações duráveis.
- Mude para o modo assíncrono. Isso ajuda a reduzir a contenção no evento de espera `IPC:DamRecordTxAck`.

  No entanto, o recurso DAS não pode garantir a durabilidade de cada um dos eventos no modo assíncrono.

IPC: eventos de espera paralelos

Os seguintes `IPC:parallel wait events` indicam que uma sessão está aguardando a comunicação entre processos relacionada às operações de execução paralela de consultas.

- `IPC:BgWorkerStartup`: um processo está aguardando um processo de operador concluir sua sequência de inicialização. Isso acontece ao inicializar operadores para execução paralela de consultas.
- `IPC:BgWorkerShutdown`: um processo está aguardando um processo de operador concluir sua sequência de shutdown. Isso ocorre durante a fase de limpeza da execução paralela da consulta.
- `IPC:ExecuteGather`: um processo está aguardando para receber dados de processos de operador paralelos durante a execução da consulta. Isso ocorre quando o processo líder precisa coletar resultados de seus operadores.
- `IPC:ParallelFinish`: um processo está aguardando operadores paralelos terminarem sua execução e relatarem seus resultados finais. Isso acontece durante a fase de conclusão da execução paralela da consulta.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

A execução paralela de consultas no PostgreSQL envolve vários processos trabalhando juntos para processar uma única consulta. Quando uma consulta é determinada como adequada para paralelização, um processo líder se coordena com um ou mais processos de operador paralelos com base na configuração de parâmetros `max_parallel_workers_per_gather`. O processo líder divide o trabalho entre os operadores, cada um processa sua parte dos dados de forma independente e os resultados são reunidos de volta ao processo líder.

> **Note**
>
> Cada operador paralelo opera como um processo separado com requisitos de recursos semelhantes a uma sessão completa de usuário. Isso significa que uma consulta paralela com 4 operadores pode consumir até 5 vezes mais recursos (CPU, memória, largura de banda de E/S) em comparação com uma consulta não paralela, pois tanto o processo líder quanto cada processo de operador mantêm suas próprias alocações de recursos. Por exemplo, configurações como `work_mem` são aplicadas individualmente a cada operador, potencialmente multiplicando o uso total de memória em todos os processos.

A arquitetura de consulta paralela consiste em três componentes principais:

- Processo líder: o processo principal que inicia a operação paralela, divide a workload e coordena com os processos de operador.
- Processos de operador: processos em segundo plano que executam partes da consulta em paralelo.
- Fusão Gather/Gather: operações que combinam resultados de vários processos de operador de volta ao líder

Durante a execução paralela, os processos precisam se comunicar entre si por meio de mecanismos de comunicação entre processos (IPC). Esses eventos de espera do IPC ocorrem durante diferentes fases:

- Inicialização do operador: quando operadores paralelos estão sendo inicializados

- Troca de dados: quando os operadores estão processando dados e enviando resultados para o líder
- Encerramento do operador: quando a execução paralela é concluída e os operadores estão sendo encerrados
- Pontos de sincronização: quando os processos precisam se coordenar ou esperar que outros processos concluam suas tarefas

Compreender esses eventos de espera é crucial para diagnosticar problemas de desempenho relacionados à execução paralela de consultas, especialmente em ambientes de alta concorrência, nos quais várias consultas paralelas podem ser executadas simultaneamente.

Possíveis causas do maior número de esperas

Vários fatores podem contribuir para um aumento nos eventos de espera do IPC relacionados ao paralelismo:

Alta concorrência de consultas paralelas

Quando muitas consultas paralelas são executadas simultaneamente, isso pode levar à contenção de recursos e ao aumento do tempo de espera para operações de IPC. Isso é particularmente comum em sistemas com altos volumes de transações ou workloads analíticas.

Planos de consulta paralela subótimos

Se o planejador de consultas escolher planos paralelos ineficientes, isso poderá resultar em paralelização desnecessária ou má distribuição do trabalho entre os operadores. Isso pode levar ao aumento das esperas de IPC, especialmente para eventos IPC:ExecuteGather e IPC:ParallelFinish. Esses problemas de planejamento geralmente resultam de estatísticas desatualizadas e do inchaço da tabela/índice.

Inicialização e encerramento frequentes de operadores paralelos

Consultas de curta duração que frequentemente iniciam e encerram operadores paralelos podem causar um aumento nos eventos `IPC:BgWorkerStartup` e `IPC:BgWorkerShutdown`. Isso geralmente é visto em workloads OLTP com muitas consultas pequenas e paralelizáveis.

Restrições de recursos

A capacidade limitada de CPU, memória ou E/S pode causar gargalos na execução paralela, levando ao aumento do tempo de espera em todos os eventos de IPC. Por exemplo, se a CPU

estiver saturada, os processos de operador podem levar mais tempo para iniciar ou processar sua parte do trabalho.

Estruturas complexas de consulta

Consultas com vários níveis de paralelismo (por exemplo, junções paralelas seguidas por agregações paralelas) podem levar a padrões de IPC mais complexos e a tempos de espera potencialmente maiores, especialmente para eventos `IPC:ExecuteGather`.

Conjuntos de resultados grandes

Consultas que produzem grandes conjuntos de resultados podem causar maiores tempos de espera `IPC:ExecuteGather`, pois o processo líder passa mais tempo coletando e processando resultados dos processos de operador.

Entender esses fatores pode ajudar a diagnosticar e resolver problemas de desempenho relacionados à execução paralela de consultas no Aurora PostgreSQL.

Ações

Quando você vê esperas relacionadas à consulta paralela, isso normalmente significa que um processo de backend está coordenando ou aguardando processos de operador paralelos. Essas esperas são comuns durante a execução de planos paralelos. Você pode investigar e mitigar o impacto dessas esperas monitorando o uso de operadores paralelos, revisando as configurações de parâmetros e ajustando a execução da consulta e a alocação de recursos.

Tópicos

- Analisar planos de consulta para paralelismo ineficiente
- Monitorar o uso de consultas paralelas
- Revisar e ajustar as configurações de consulta paralela
- Otimizar a alocação de recursos
- Investigar o gerenciamento de conexão
- Revisar e otimizar as operações de manutenção
- Utilize o gerenciamento de planos de consultas (QPM)

Analisar planos de consulta para paralelismo ineficiente

A execução paralela de consultas pode frequentemente levar à instabilidade do sistema, picos de CPU e variações imprevisíveis no desempenho da consulta. É crucial analisar minuciosamente se o paralelismo realmente melhora sua workload específica. Use EXPLAIN ANALYZE para revisar os planos de execução de consultas paralelas.

Desative temporariamente o paralelismo no nível da sessão para comparar a eficiência do plano:

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

Reative o paralelismo e compare:

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

Se a desativação do paralelismo produzir resultados melhores ou mais consistentes, desative-o para consultas específicas no nível da sessão usando os comandos SET. Para um impacto mais amplo, você pode querer desabilitar o paralelismo no nível da instância ajustando os parâmetros relevantes em seu grupo de parâmetros do cluster ou instância. Para obter mais informações, consulte Amazon Aurora PostgreSQL parameters.

Monitorar o uso de consultas paralelas

Use as consultas a seguir para obter visibilidade da atividade e da capacidade de consultas paralelas:

Verifique os processos de operador paralelos ativos:

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

Esta consulta mostra o número de processos de operador paralelos ativos. Um valor alto pode indicar que seu `max_parallel_workers` está configurado com um valor alto e você pode querer reduzi-lo.

Verifique as consultas paralelas simultâneas:

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

Essa consulta retorna o número de processos líderes distintos que iniciaram consultas paralelas. Um número alto aqui indica que várias sessões estão executando consultas paralelas simultaneamente, o que pode aumentar a demanda de CPU e memória.

Revisar e ajustar as configurações de consulta paralela

Analise os seguintes parâmetros para garantir que estejam alinhados à sua workload:

- `max_parallel_workers`: número total de workers paralelos em todas as sessões.
- `max_parallel_workers_per_gather`: máximo de workers por consulta.

Para workloads OLAP, aumentar esses valores pode melhorar o desempenho. Para workloads OLTP, valores mais baixos geralmente são preferidos.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

Otimizar a alocação de recursos

Monitore a utilização da CPU e ajuste o número de vCPUs se for consistentemente alto e se seu aplicativo se beneficiar de consultas paralelas. Verifique se há memória adequada disponível para operações paralelas.

- Use as métricas do Performance Insights para determinar se o sistema está limitado pela CPU.
- Cada operador paralelo usa o seu próprio `work_mem`. Verifique se o uso total da memória está dentro dos limites da instância.

As consultas paralelas podem consumir substancialmente mais recursos do que as consultas não paralelas, porque cada processo de operador é completamente separado que tem aproximadamente o mesmo impacto no sistema que uma sessão de usuário adicional. Isso deve ser levado em consideração ao escolher um valor para essa configuração, bem como ao definir outras configurações que controlam a utilização de recursos, como `work_mem`. Para obter mais informações, consulte a Documentação do PostgreSQL. Limites de recursos, como `work_mem`, são aplicados individualmente a cada operador, o que significa que a utilização total pode ser muito maior em todos os processos do que normalmente seria em qualquer processo único.

Aumente as vCPUs ou ajuste os parâmetros de memória se sua workload estiver altamente paralelizada.

Investigar o gerenciamento de conexão

Se estiver experimentando esgotamento de conexões, revise as estratégias de pool de conexões do aplicativo. Implemente o pool de conexões no nível do aplicativo, se ainda não estiver em uso.

Revisar e otimizar as operações de manutenção

Coordene a criação de índices e outras tarefas de manutenção para evitar a contenção de recursos. Agende essas operações fora do horário de pico. Evite agendar manutenções pesadas (por exemplo, construções paralelas de índices) durante períodos de alta carga de consultas do usuário. Essas operações podem consumir workers paralelos e impactar o desempenho de consultas regulares.

Utilize o gerenciamento de planos de consultas (QPM)

No Aurora PostgreSQL, o atributo de gerenciamento de planos de consulta (QPM) foi projetado para garantir a adaptabilidade e a estabilidade do plano, independentemente das alterações no ambiente de banco de dados que poderiam causar a regressão do plano de consulta. Para obter mais informações, consulte Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL. O QPM fornece algum controle sobre o otimizador. Revise os planos aprovados no QPM para garantir que estejam alinhados com as configurações atuais de paralelismo. Atualize ou remova planos desatualizados que possam estar forçando uma execução paralela subótima.

Você também pode corrigir os planos usando pg_hint_plan. Para obter mais informações, consulte Corrigir planos usando pg_hint_plan. Você pode usar a dica chamada `Parallel` para forçar a execução paralela. Para obter mais informações, consulte Dicas para planos paralelos.

IPC:ProcArrayGroupUpdate

O evento `IPC:ProcArrayGroupUpdate` ocorre quando uma sessão está aguardando o líder do grupo atualizar o status da transação ao final dessa operação. Embora o PostgreSQL geralmente associe eventos de espera do tipo IPC a operações de consulta paralela, esse evento de espera determinado não é específico de consultas paralelas.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

As informações sobre esse evento de espera são aceitas em todas as versões do Aurora PostgreSQL.

Contexto

Noções básicas sobre a matriz do processo: a matriz do processo (proc) é uma estrutura de memória compartilhada no PostgreSQL. Ela contém informações sobre todos os processos em execução, incluindo detalhes da transação. Durante a conclusão da transação (`COMMIT` ou `ROLLBACK`), o ProcArray precisa ser atualizado para refletir a alteração e limpar o ID da transação da matriz. A sessão que tenta concluir sua transação deve adquirir um bloqueio exclusivo no ProcArray. Isso impede que outros processos tenham bloqueios compartilhados ou exclusivos.

Mecanismo de atualização de grupo: ao executar um COMMIT ou ROLLBACK, se um processo de backend não conseguir recuperar um ProcArrayLock no modo exclusivo, ele atualizará um campo especial chamado ProcArrayGroupMember. Isso adiciona a transação à lista de sessões que pretendem terminar. Esse processo de backend então é suspenso, e o tempo de suspensão é instrumentado como o evento de espera ProcArrayGroupUpdate. O primeiro processo no ProcArray com o procArrayGroupMember, conhecido como processo líder, adquire o ProcArrayLock no modo

exclusivo. Depois, ele limpa a lista de processos que aguardam a limpeza do transactionID do grupo. Quando isso é concluído, o líder libera o ProcArrayLock e, depois, ativa todos os processos dessa lista, notificando-os de que a transação foi concluída.

Possíveis causas do maior número de esperas

Quanto mais processos estiverem em execução, mais tempo um líder manterá um procArrayLock no modo exclusivo. Consequentemente, quanto mais transações de gravação ocorrerem, maior a chance de entrarem em um cenário de atualização em grupo, causando um possível acúmulo de processos aguardando o evento de espera `ProcArrayGroupUpdate`. Na visualização Top SQL do Database Insights, você verá que COMMIT é a declaração com a maior parte desse evento de espera. Isso é esperado, mas exigirá uma investigação mais profunda do SQL de gravação específico que está sendo executado para determinar qual ação apropriada realizar.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera. Identifique eventos `IPC:ProcArrayGroupUpdate` utilizando o Insights de Performance do Amazon RDS ou consultando a visualização `pg_stat_activity` do sistema PostgreSQL.

Tópicos

- Monitorar operações de confirmação e reversão de transações
- Reduzir a simultaneidade
- Implementar o agrupamento de conexões

Monitorar operações de confirmação e reversão de transações

Monitorar confirmações e reversões: um número maior de confirmações e reversões pode aumentar a pressão sobre o ProcArray. Por exemplo, se uma instrução SQL começar a falhar devido ao aumento das violações de chaves duplicadas, você poderá observar um aumento nas reversões, o que pode aumentar a contenção do ProcArray e o inchaço da tabela.

O Amazon RDS Database Insights fornece as métricas `xact_commit` e `xact_rollback` do PostgreSQL e relata o número de confirmações e reversões por segundo.

Reduzir a simultaneidade

Transações em lote: sempre que possível, agrupe operações em transações únicas para reduzir as operações de confirmação/reversão.

Limitar a simultaneidade: reduza o número de transações ativas simultaneamente para aliviar a contenção de bloqueios no ProcArray. Embora isso exija alguns testes, reduzir o número total de conexões simultâneas pode reduzir a contenção e manter o throughput.

Implementar o agrupamento de conexões

Soluções de agrupamento de conexões: use o grupo de conexões para gerenciar conexões de banco de dados com eficiência, reduzindo o número total de backends e, portanto, a workload no ProcArray. Embora isso exija alguns testes, reduzir o número total de conexões simultâneas pode reduzir a contenção e manter o throughput.

Para acessar mais informações, consulte Connection pooling for Aurora PostgreSQL.

Reduzir os surtos de conexão: da mesma forma, um padrão de criação e encerramento frequentes de conexões causa pressão adicional no ProcArray. Ao reduzir esse padrão, a contenção geral é reduzida.

Lock:advisory

O evento `Lock:advisory` ocorre quando uma aplicação PostgreSQL utiliza um bloqueio para coordenar as atividades em várias sessões.

Tópicos

- Versões de mecanismos relevantes
- Contexto
- Causas
- Ações

Versões de mecanismos relevantes

As informações sobre eventos de espera são relevantes para o Aurora PostgreSQL versão 9.6 e versões superiores.

Contexto

Bloqueios consultivos do PostgreSQL são bloqueios cooperativos em nível de aplicação, feitos explicitamente e desfeitos pelo código da aplicação do usuário. Uma aplicação pode utilizar bloqueios consultivos do PostgreSQL para coordenar atividades em várias sessões. Ao contrário

de bloqueios regulares, ou em nível de objeto ou linha, a aplicação tem controle total ao longo da vida útil do bloqueio. Para obter mais informações, consulte o tópico sobre Bloqueios consultivos na documentação do PostgreSQL.

Bloqueios consultivos podem ser liberados antes que uma transação termine ou podem ser mantidos por uma sessão em todas as transações. Isso não é válido para bloqueios implícitos aplicados pelo sistema, como um bloqueio exclusivo de acesso em uma tabela adquirida por uma instrução `CREATE INDEX`.

Para obter uma descrição das funções utilizadas para adquirir (bloquear) e liberar (desbloquear) bloqueios consultivos, consulte o tópico sobre Funções de bloqueios consultivos, na documentação do PostgreSQL.

Bloqueios consultivos são implementados sobre o sistema de bloqueio regular do PostgreSQL e ficam visíveis na visualização do sistema `pg_locks`.

Causas

Esse tipo de bloqueio é controlado exclusivamente por uma aplicação que o utiliza explicitamente. Bloqueios consultivos que são adquiridos para cada linha como parte de uma consulta podem causar um aumento nos bloqueios ou um acúmulo a longo prazo.

Esses efeitos acontecem quando a consulta é executada de uma maneira que adquire bloqueios em mais linhas do que as retornadas pela consulta. A aplicação deve eventualmente liberar todos os bloqueios, mas, se eles forem adquiridos em linhas que não são retornadas, a aplicação não poderá localizar todos os bloqueios.

O exemplo a seguir foi extraído do tópico Bloqueios consultivos na documentação do PostgreSQL.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

Nesse exemplo, a cláusula `LIMIT` apenas pode interromper a saída da consulta depois que as linhas já foram selecionadas internamente e seus valores de ID estão bloqueados. Isso pode acontecer repentinamente quando um volume de dados crescente faz com que o planejador escolha um plano de execução diferente que não foi testado durante o desenvolvimento. O acúmulo nesse caso acontece porque a aplicação chama explicitamente `pg_advisory_unlock` para cada valor de ID bloqueado. No entanto, nesse caso, não é possível encontrar o conjunto de bloqueios adquiridos em linhas que não foram retornadas. Como os bloqueios são adquiridos em nível de sessão, eles não são liberados automaticamente no final da transação.

Outra possível causa para picos em tentativas de bloqueio bloqueadas são conflitos não intencionais. Nesses conflitos, partes não relacionadas da aplicação compartilham o mesmo espaço de ID de bloqueio por engano.

Ações

Revise o uso da aplicação de bloqueios consultivos e detalhe onde e quando no fluxo de aplicação cada tipo de bloqueio consultivo é adquirido e liberado.

Determine se uma sessão está adquirindo muitos bloqueios ou se uma sessão de longa execução não está liberando bloqueios cedo o suficiente, resultando em um acúmulo lento de bloqueios. Você pode corrigir um acúmulo lento de bloqueios em nível de sessão encerrando a sessão com `pg_terminate_backend(pid)`.

Um cliente que aguarda um bloqueio de consultoria aparece em `pg_stat_activity` com `wait_event_type=Lock` e `wait_event=advisory`. É possível obter valores de bloqueio específicos consultando a visualização do sistema `pg_locks` em busca do mesmo `pid`, procurando `locktype=advisory` e `granted=f`.

Em seguida, identifique a sessão de bloqueio consultando `pg_locks` em busca do mesmo bloqueio consultivo que possui `granted=t`, conforme mostrado no exemplo a seguir.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
```

```
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

Todas as funções de API de bloqueio consultivo têm dois conjuntos de argumentos, um argumento `bigint` ou dois argumentos `integer`:

- Para as funções de API com um único argumento `bigint`, os 32 bits superiores estão em `pg_locks.classid` e os 32 bits inferiores estão em `pg_locks.objid`.
- Para as funções da API com dois argumentos `integer`, o primeiro argumento é `pg_locks.classid` e o segundo é `pg_locks.objid`.

O valor `pg_locks.objsubid` indica qual formato de API foi utilizado: 1 significa um argumento `bigint`; 2significa dois argumentos `integer`.

Lock:extend

O evento `Lock:extend` ocorre quando um processo de backend está aguardando para bloquear uma relação com o objetivo de a estender, enquanto outro processo tem um bloqueio nessa relação para o mesmo objetivo.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

O evento `Lock:extend` indica que um processo de backend está aguardando para estender uma relação na qual outro processo de backend mantém um bloqueio enquanto estende essa relação. Como apenas um processo por vez pode estender uma relação, o sistema gera um evento de espera `Lock:extend`. Operações `INSERT`, `COPY` e `UPDATE` podem gerar esse evento.

Possíveis causas do maior número de esperas

Quando o evento `Lock:extend` aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Surto de inserções simultâneas ou atualizações na mesma tabela

Pode haver um aumento no número de sessões simultâneas com consultas que inserem ou atualizam a mesma tabela.

Largura de banda da rede insuficiente

É possível que a largura de banda da rede na instância de banco de dados para as necessidades de comunicação de armazenamento da workload atual. Isso pode contribuir para a latência de armazenamento que causa um aumento em eventos `Lock:extend`.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Reduzir inserções e atualizações simultâneas para a mesma relação
- Aumentar a largura de banda da rede

Reduzir inserções e atualizações simultâneas para a mesma relação

Primeiro, determine se há um aumento nas métricas `tup_inserted` e `tup_updated` e um aumento acompanhante nesse evento de espera. Em caso afirmativo, verifique quais relações

estão em alta disputa por operações de inserção e atualização. Para determinar isso, consulte a visualização `pg_stat_all_tables` para os valores nos campos `n_tup_ins` e `n_tup_upd`. Para obter mais informações sobre a visualização `pg_stat_all_tables`, consulte pg_stat_all_tables na documentação do PostgreSQL.

Para obter mais informações sobre bloqueio e consultas bloqueadas, consulte `pg_stat_activity` como no exemplo a seguir:

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+---------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Depois de identificar relações que contribuem para o aumento dos eventos `Lock:extend`, utilize as seguintes técnicas para reduzir a contenção:

- Descubra se é possível utilizar o particionamento para reduzir a contenção para a mesma tabela. Separar tuplas inseridas ou atualizadas em partições diferentes pode reduzir a contenção. Para obter informações sobre particionamento, consulte Gerenciar partições do PostgreSQL com a extensão pg_partman.
- Se o evento de espera for principalmente devido a atividades de atualização, reduza o valor do fator de preenchimento da relação. Isso pode reduzir as solicitações de novos bloqueios durante a atualização. O fator de preenchimento é um parâmetro de armazenamento de uma tabela que

determina o espaço máximo para empacotar uma página de tabela. Ele é expresso como uma porcentagem do espaço total de uma página. Para obter mais informações sobre o parâmetro de fator de preenchimento, consulte CREATE TABLE na documentação do PostgreSQL.

> **Important**
>
> É altamente recomendável testar seu sistema se você alterar o fator de preenchimento, pois isso pode afetar negativamente a performance dependendo da workload.

Aumentar a largura de banda da rede

Para ver se há um aumento na latência de gravação, verifique a métrica `WriteLatency` no CloudWatch. Se houver, use as métricas `WriteThroughput` e `ReadThroughput` do Amazon CloudWatch para monitorar o tráfego relacionado ao armazenamento no cluster de banco de dados. Essas métricas podem ajudar a determinar se a largura de banda da rede é suficiente para a atividade de armazenamento da sua workload.

Se a largura de banda da rede não suficiente, aumente-a. Se a sua instância de banco de dados estiver atingindo os limites de largura de banda da rede, a única maneira de aumentar a largura de banda será ampliar o tamanho da instância de banco de dados.

Para obter mais informações sobre métricas do CloudWatch, consulte Métricas do Amazon CloudWatch para o Amazon Aurora. Para obter informações sobre a performance de rede de cada classe de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

Lock:Relation

O evento `Lock:Relation` ocorre quando uma consulta está aguardando para adquirir um bloqueio em uma tabela ou visualização (relação) que está atualmente bloqueada por outra transação.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

A maioria dos comandos PostgreSQL utiliza bloqueios implicitamente para controlar o acesso simultâneo aos dados em tabelas. Também é possível utilizar esses bloqueios explicitamente no código da aplicação com o comando `LOCK`. Vários modos de bloqueio não são compatíveis entre si e podem bloquear transações quando tentam acessar o mesmo objeto. Quando isso acontece, o Aurora PostgreSQL gera um evento `Lock:Relation`. Veja a seguir alguns exemplos comuns:

- Bloqueios exclusivos, como `ACCESS EXCLUSIVE`, podem bloquear todo o acesso simultâneo. Operações de linguagem de definição de dados (DDL), como `DROP TABLE`, `TRUNCATE`, `VACUUM FULL` e `CLUSTER`, adquirem bloqueios `ACCESS EXCLUSIVE` implicitamente. `ACCESS EXCLUSIVE` também é o modo de bloqueio padrão para instruções `LOCK TABLE` que não especificam um modo explicitamente.
- Usar `CREATE INDEX (without CONCURRENT)` em uma tabela gera conflito com instruções de linguagem de manipulação de dados (DML) `UPDATE`, `DELETE` e `INSERT`, que adquirem bloqueios `ROW EXCLUSIVE`.

Para obter mais informações sobre bloqueios em nível da tabela e modos de bloqueio conflitantes, consulte o tópico sobre Bloqueio explícito na documentação do PostgreSQL.

Normalmente, o bloqueio de consultas e transações é liberado de uma das seguintes maneiras:

- Consulta de bloqueio: a aplicação pode cancelar a consulta ou o usuário pode encerrar o processo. O mecanismo também pode forçar a finalização da consulta devido a um tempo limite de declaração de uma sessão ou um mecanismo de detecção de deadlock.
- Transação de bloqueio: uma transação para de bloquear quando executa uma instrução `ROLLBACK` ou `COMMIT`. Reversões também acontecem automaticamente quando as sessões são desconectadas por um cliente ou por problemas de rede, ou quando são encerradas. As sessões podem ser encerradas quando o mecanismo de banco de dados é desligado, quando o sistema está sem memória e assim por diante.

Possíveis causas do maior número de esperas

Quando o evento `Lock:Relation` ocorre com maior frequência do que o normal, pode indicar um problema de performance. As causas típicas incluem:

Maior número de sessões simultâneas com bloqueios de tabela conflitantes

Pode haver um aumento no número de sessões simultâneas com consultas que bloqueiam a mesma tabela com modos de bloqueio conflitantes.

Operações de manutenção

Operações de manutenção de integridade, como `VACUUM` e `ANALYZE`, podem aumentar significativamente o número de bloqueios conflitantes. `VACUUM FULL` adquire um bloqueio `ACCESS EXCLUSIVE` e `ANALYZE` adquire um bloqueio `SHARE UPDATE EXCLUSIVE`. Ambos os tipos de bloqueios podem causar um evento de espera `Lock:Relation`. Operações de manutenção de dados de aplicações, como atualizar uma visualização materializada, também podem aumentar as consultas e transações bloqueadas.

Bloqueios em instâncias de leitor

Pode haver um conflito entre os bloqueios de relação mantidos pelo gravador e os leitores. No momento, só bloqueios de relação do `ACCESS EXCLUSIVE` são replicados para instâncias do leitor. No entanto, o bloqueio de relação do `ACCESS EXCLUSIVE` entrará em conflito com qualquer bloqueio de relação do `ACCESS SHARE` mantido pelo leitor. Isso pode causar um aumento nos eventos de espera de relação de bloqueio no leitor.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Reduzir o impacto do bloqueio de instruções SQL
- Minimizar o efeito de operações de manutenção
- Verificar se há bloqueios de leitor

Reduzir o impacto do bloqueio de instruções SQL

Para reduzir o impacto do bloqueio de instruções SQL, modifique o código da aplicação sempre que possível. Veja a seguir duas técnicas comuns para reduzir bloqueios:

- Usar a opção `NOWAIT`: alguns comandos SQL, como instruções `SELECT` e `LOCK`, oferecem suporte a essa opção. A diretiva `NOWAIT` cancela a consulta de solicitação de bloqueio quando o bloqueio não pode ser adquirido imediatamente. Essa técnica pode evitar que uma sessão de bloqueio cause um empilhamento de sessões bloqueadas por detrás dela.

  Por exemplo: suponha que a transação A esteja aguardando um bloqueio mantido pela transação B. Se B solicitar um bloqueio em uma tabela bloqueada pela transação C, a transação A poderá ser bloqueada até a conclusão da transação C. Porém, se a transação B utilizar um `NOWAIT` quando solicitar o bloqueio em C, ela pode falhar rapidamente e garantir que a transação A não precise aguardar indefinidamente.
- Usar `SET lock_timeout`: defina um valor de `lock_timeout` para limitar o tempo de espera de uma instrução SQL para adquirir um bloqueio em uma relação. Se o bloqueio não for obtido dentro do tempo limite definido, a transação que o está solicitando será cancelada. Defina o valor no nível da sessão.

Minimizar o efeito de operações de manutenção

Operações de manutenção, como `VACUUM` e `ANALYZE`, são importantes. Recomendamos que você não as desative ao encontrar eventos de espera `Lock:Relation` relacionados a essas operações de manutenção. As abordagens a seguir podem minimizar o efeito dessas operações:

- Execute operações de manutenção manualmente fora do horário de pico.
- Para reduzir esperas `Lock:Relation` causadas por tarefas de autovacuum, realize qualquer ajuste de autovacuum necessário. Para obter mais informações sobre ajuste de autovacuum, consulte Trabalhar com o autovacuum do PostgreSQL no Amazon RDS, no Guia do usuário do Amazon RDS.

Verificar se há bloqueios de leitor

É possível ver como as sessões simultâneas em um gravador e leitores podem estar mantendo bloqueios que são aplicáveis uns aos outros. Uma maneira de fazer isso é executando consultas que retornem o tipo de bloqueio e a relação. Na tabela, você pode encontrar uma sequência de consultas para duas dessas sessões simultâneas, uma de gravador e uma de leitor.

O processo de repetição aguarda a duração de `max_standby_streaming_delay` antes de cancelar a consulta do leitor. Como mostrado no exemplo, o tempo limite de bloqueio de 100 ms está

bem abaixo do `max_standby_streaming_delay` padrão de 30 segundos. O bloqueio esgota-se antes que seja um problema.

Evento de sequência	Sessão	Comando ou saída

Define uma variável de ambiente chamada READER com o valor especificado e tenta se conectar à instância de banco de dados com esse endpoint.	Sessão de leitor	Comando da CLI:
		`export READER=aurorapg2.1234567890.us-west-1.rds.amazonaws.com`
		`psql -h $READER`
		Resultado:
		`psql (15devel, server 10.14)`
		`Type "help" for help.`

Define uma variável de ambiente chamada WRITER e tenta se conectar à instância de banco de dados com esse endpoint.	Sessão de gravador	Comando da CLI:
		`export WRITER=aurorapg1.1234567890.us-west-1.rds.amazonaws.com`
		`psql -h $WRITER`
		Resultado:
		`psql (15devel, server 10.14)`
		`Type "help" for help.`

A sessão do gravador cria a tabela t1 na instância de gravador.	Sessão de gravador	Consulta do PostgreSQL:
		`postgres=> CREATE TABLE t1(b integer);`
		`CREATE TABLE`

Evento de sequência	Sessão	Comando ou saída
Se não houver consultas conflitantes no gravador, o bloqueio ACCESS EXCLUSIVE é obtido no gravador imediatamente.	Sessão de gravador	Bloqueio `ACCESS EXCLUSIVE` habilitado

A sessão do leitor especific a um intervalo de tempo limite de bloqueio igual a 100 milissegundos.	Sessão de leitor	Consulta do PostgreSQL:
		`postgres=> SET lock_timeout=100;`
		`SET`

A sessão do leitor tenta ler dados da tabela t1 na instância de leitor.	Sessão de leitor	Consulta do PostgreSQL:
		`postgres=> SELECT * FROM t1;`
		Exemplo de resultado:
		`b`
		`---`
		`(0 rows)`

A sessão do gravador descarta t1.	Sessão de gravador	Consulta do PostgreSQL:
		`postgres=> BEGIN;`
		`BEGIN`
		`postgres=> DROP TABLE t1;`
		`DROP TABLE`
		`postgres=>`

Evento de sequência	Sessão	Comando ou saída

A consulta atinge o tempo limite e é cancelada no leitor.	Sessão de leitor	Consulta do PostgreSQL:
		`postgres=>  SELECT * FROM t1;`
		Exemplo de resultado:
		`ERROR:  canceling statement due to lock timeout`
		`LINE 1: SELECT * FROM t1;`
		`^`

Para determinar a causa do erro, a sessão do leitor consulta `pg_locks` e `pg_stat_activity`	Sessão de leitor	Consulta do PostgreSQL:
		`postgres=>  SELECT locktype, relation, mode, backend_type`
		`postgres=>  FROM pg_locks l, pg_stat_activity t1`
		`postgres=>  WHERE l.pid=t1.pid AND relation = 't1'::regclass::oid;`

Evento de sequência	Sessão	Comando ou saída

O resultado indica que o processo `aurora wal replay` está mantendo um bloqueio `ACCESS EXCLUSIVE` na tabela t1.	Sessão de leitor	Resultado da consulta:
		`locktype | relation | mode | backend_type`
		`----------+----------+------------------+---------------`
		`relation | 68628525 | AccessExclusiveLock | aurora wal replay`
		`(1 row)`

Lock:transactionid

O evento `Lock:transactionid` ocorre quando uma transação está aguardando um bloqueio em nível de linha.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

O evento `Lock:transactionid` ocorre quando uma transação está tentando adquirir um bloqueio em nível de linha já concedido a uma transação que está sendo executada simultaneamente. A sessão que mostra o evento de espera `Lock:transactionid` está bloqueada devido a esse

bloqueio. Depois que a transação de bloqueio terminar em uma instrução `COMMIT` ou `ROLLBACK`, a transação bloqueada pode continuar.

A semântica de controle de simultaneidade de várias versões do Aurora PostgreSQL garante que os leitores não bloqueiem gravadores e que os gravadores não bloqueiem os leitores. Para que ocorram conflitos em nível de linha, transações de bloqueio e bloqueadas devem emitir instruções conflitantes dos seguintes tipos:

- `UPDATE`
- `SELECT … FOR UPDATE`
- `SELECT … FOR KEY SHARE`

A instrução `SELECT … FOR KEY SHARE` é um caso especial. O banco de dados usa a cláusula `FOR KEY SHARE` para otimizar a performance da integridade referencial. Um bloqueio em nível de linha em uma linha pode bloquear comandos `INSERT`, `UPDATE` e `DELETE` em outras tabelas que fazem referência à linha.

Possíveis causas do maior número de esperas

Quando esse evento aparece mais do que o normal, a causa geralmente envolve instruções `UPDATE`, `SELECT … FOR UPDATE` ou `SELECT … FOR KEY SHARE` combinadas com as seguintes condições.

Tópicos

- Alta simultaneidade
- Ocioso na transação
- Transações de longa execução

Alta simultaneidade

O Aurora PostgreSQL pode utilizar semântica de bloqueio granular em nível de linha. A probabilidade de conflitos em nível de linha aumenta quando as condições a seguir são atendidas:

- Uma workload altamente simultânea controla as mesmas linhas.
- A simultaneidade aumenta.

Ocioso na transação

Às vezes, a coluna `pg_stat_activity.state` mostra o valor `idle in transaction`. Esse valor aparece para sessões que iniciaram uma transação, mas ainda não emitiram um `COMMIT` ou `ROLLBACK`. Se o valor de `pg_stat_activity.state` não for `active`, a consulta mostrada em `pg_stat_activity` será a mais recente a terminar a execução. A sessão de bloqueio não está processando uma consulta ativamente porque uma transação aberta está mantendo um bloqueio.

Se uma transação ociosa adquiriu um bloqueio em nível de linha, talvez ela esteja impedindo que outras sessões o adquiram. Essa condição leva à ocorrência frequente do evento de espera `Lock:transactionid`. Para diagnosticar o problema, examine a saída proveniente de `pg_stat_activity` e `pg_locks`.

Transações de longa execução

Transações executadas por um longo tempo recebem bloqueios por um longo tempo. Esses bloqueios de longa duração podem impedir que outras transações sejam executadas.

Ações

O bloqueio de linhas é um conflito entre instruções `UPDATE`, `SELECT … FOR UPDATE` ou `SELECT … FOR KEY SHARE`. Antes de tentar uma solução, descubra quando essas instruções estão sendo executadas na mesma linha. Use essas informações para escolher uma estratégia descrita nas seguintes seções.

Tópicos

- Responder a alta simultaneidade
- Responder a transações ociosas
- Responder a transações de longa duração

Responder a alta simultaneidade

Se a simultaneidade for o problema, tente uma das seguintes técnicas:

- Reduza a simultaneidade na aplicação. Por exemplo, reduza o número de sessões ativas.
- Implemente um pool de conexões. Para saber como agrupar conexões com o RDS Proxy, consulte Amazon RDS Proxypara Aurora.

- Projete a aplicação ou o modelo de dados para evitar a contenção de instruções `UPDATE` e `SELECT … FOR UPDATE`. Também é possível diminuir o número de chaves estrangeiras acessadas por instruções `SELECT … FOR KEY SHARE`.

Responder a transações ociosas

Se `pg_stat_activity.state` mostrar `idle in transaction`, utilize as seguintes estratégias:

- Ative a confirmação automática sempre que possível. Essa abordagem impede que transações bloqueiem outras transações enquanto aguardam `COMMIT` ou `ROLLBACK`.
- Procure caminhos de código que não contenham `COMMIT`, `ROLLBACK` ou `END`.
- Assegure-se de que a lógica de tratamento de exceções na seu aplicação sempre tenha um caminho para um válido `end of transaction`.
- Assegure-se de que a sua aplicação processe os resultados da consulta depois de encerrar a transação com `COMMIT` ou `ROLLBACK`.

Responder a transações de longa duração

Se transações de longa duração estiverem causando a ocorrência frequente de `Lock:transactionid`, tente as estratégias a seguir:

- Mantenha os bloqueios de linha fora de transações de longa execução.
- Limite o comprimento das consultas implementando a confirmação automática sempre que possível.

Lock:tuple

O evento `Lock:tuple` ocorre quando um processo de backend está aguardando para adquirir um bloqueio em uma tupla.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

O evento `Lock:tuple` indica que um backend está aguardando para adquirir um bloqueio em uma tupla enquanto outro backend mantém um bloqueio conflitante na mesma tupla. A tabela a seguir ilustra um cenário em que as sessões geram o evento `Lock:tuple`.

Temp	Sessão 1	Sessão 2	Sessão 3
t1	Inicia uma transação.		
t2	Atualiza a linha 1.		
t3		Atualiza a linha 1. A sessão adquire um bloqueio exclusivo na tupla e aguarda a sessão 1 liberar esse bloqueio via confirmação ou reversão.	
t4			Atualiza a linha 1. A sessão aguarda a sessão 2 liberar o bloqueio exclusivo na tupla.

Outra alternativa é simular esse evento de espera utilizando a ferramenta de benchmarking `pgbench`. Configure um número elevado de sessões simultâneas para atualizar a mesma linha em uma tabela com um arquivo SQL personalizado.

Para saber mais sobre modos de bloqueio conflitantes, consulte o tópico sobre Bloqueio explícito, na documentação do PostgreSQL. Para saber mais sobre `pgbench`, consulte pgbench na documentação do PostgreSQL.

Possíveis causas do maior número de esperas

Quando esse evento aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

- Um número elevado de sessões simultâneas está tentando adquirir um bloqueio conflitante para a mesma tupla executando instruções `UPDATE` ou `DELETE`.
- Sessões altamente simultâneas estão executando uma instrução `SELECT` utilizando os modos de bloqueio `FOR UPDATE` ou `FOR NO KEY UPDATE`.
- Diversos fatores fazem com que grupos de aplicações ou conexões abram mais sessões para executar as mesmas operações. À medida que novas sessões estão tentando modificar as mesmas linhas, a carga de banco de dados pode atingir picos e `Lock:tuple` pode surgir.

Para obter mais informações, consulte o tópico sobre Bloqueios em nível de linha, na documentação do PostgreSQL.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Investigar a lógica da sua aplicação
- Localizar a sessão bloqueadora
- Reduzir a simultaneidade quando ela estiver elevada
- Solucionar problemas de gargalos

Investigar a lógica da sua aplicação

Descubra se uma sessão bloqueadora está no estado `idle in transaction` por muito tempo. Em caso positivo, considere encerrar a sessão bloqueadora como uma solução de curto prazo. Também é possível utilizar a função `pg_terminate_backend`. Para obter mais informações sobre essa função, consulte Funções de sinalização de servidor, na documentação do PostgreSQL.

Para uma solução de longo prazo, faça o seguinte:

- Ajuste a lógica da aplicação.
- Use o parâmetro `idle_in_transaction_session_timeout`. Esse parâmetro encerra qualquer sessão com uma transação aberta que tenha ficado ociosa por um tempo maior que o especificado. Para obter mais informações, consulte o tópico sobre Padrões de conexão de clientes, na documentação do PostgreSQL.
- Use a confirmação automática o máximo possível. Para obter mais informações, consulte SET AUTOCOMMIT, na documentação do PostgreSQL.

Localizar a sessão bloqueadora

Enquanto o evento de espera `Lock:tuple` está ocorrendo, identifique a sessão bloqueadora e a sessão bloqueada descobrindo quais bloqueios dependem um do outro. Para obter mais informações, consulte o tópico sobre Informações de dependências de bloqueios, na wiki do PostgreSQL. Para analisar eventos `Lock:tuple` passados, utilize a função do Aurora `aurora_stat_backend_waits`.

O exemplo a seguir consulta todas as sessões, filtrando em `tuple` e ordenando por `wait_time`.

```
--AURORA_STAT_BACKEND_WAITS
      SELECT a.pid,
             a.usename,
             a.app_name,
             a.current_query,
             a.current_wait_type,
             a.current_wait_event,
             a.current_state,
             wt.type_name AS wait_type,
             we.event_name AS wait_event,
             a.waits,
             a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
                     state AS current_state,
                     left(query,80) as current_query,
                     (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND usename<>'rdsadmin') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we
WHERE we.event_name = 'tuple'
    ORDER BY a.wait_time;

  pid  | usename | app_name |                 current_query                  |
 current_wait_type | current_wait_event | current_state | wait_type | wait_event |
 waits | wait_time
```

```
-------+---------+----------+-------------------------------------------------
+------------------+--------------------+--------------+----------+-----------
+-------+-----------
 32136 | sys     | psql     | /*session3*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000018
 11999 | sys     | psql     | /*session4*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000024
```

Reduzir a simultaneidade quando ela estiver elevada

O evento `Lock:tuple` pode ocorrer constantemente, especialmente em um tempo de workload ocupado. Nessa situação, considere reduzir a simultaneidade elevada para linhas muito ocupadas. Várias vezes, apenas algumas linhas controlam uma fila ou a lógica booleana, o que torna essas linhas muito ocupadas.

Você pode reduzir a simultaneidade utilizando diferentes abordagens com base no seu requisito de negócios, na lógica da aplicação e no tipo de workload. Por exemplo, você pode fazer o seguinte:

- Reformule sua tabela e a lógica de dados para reduzir a simultaneidade elevada.
- Altere a lógica da aplicação para reduzir a simultaneidade elevada no nível da linha.
- Aproveite e reformule consultas com bloqueios em nível de linha.
- Use a cláusula `NOWAIT` com operações de novas tentativas.
- Considere utilizar o controle de simultaneidade lógico otimista e bloqueio híbrido.
- Considere modificar o nível de isolamento do banco de dados.

Solucionar problemas de gargalos

O `Lock:tuple` pode ocorrer com gargalos, como falta de CPU ou uso máximo da largura de banda do Amazon EBS. Para diminuir gargalos, considere as seguintes abordagens:

- Aumente a escala do seu tipo de classe de instância na vertical.
- Otimize consultas que consomem muitos recursos.
- Altere a lógica da aplicação.
- Arquive dados acessados raramente.

LWLock:buffer_content (BufferContent)

O evento `LWLock:buffer_content` ocorre quando uma sessão aguarda para ler ou gravar uma página de dados na memória enquanto outra sessão fica com a página bloqueada para gravação. No Aurora PostgreSQL 13 e versões superiores, esse evento de espera é chamado de `BufferContent`.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

Para ler ou manipular dados, o PostgreSQL os acessa por meio de buffers de memória compartilhada. Para ler a partir do buffer, um processo obtém um bloqueio leve (LWLock) no conteúdo do buffer no modo compartilhado. Para gravar no buffer, ele adquire esse bloqueio no modo exclusivo. Bloqueios compartilhados permitem que outros processos adquiram bloqueios compartilhados simultaneamente nesse conteúdo. Bloqueios exclusivos impedem que outros processos obtenham qualquer tipo de bloqueio nele.

O evento `LWLock:buffer_content` (`BufferContent`) indica que vários processos estão tentando obter bloqueios leves (LWLocks) no conteúdo de um buffer específico.

Possíveis causas do maior número de esperas

Quando o evento `LWLock:buffer_content` (`BufferContent`) aparece mais que o normal, possivelmente indicando um problema de performance, as causas típicas incluem:

Maior número de atualizações simultâneas para os mesmos dados

> Pode haver um aumento no número de sessões simultâneas com consultas que atualizam o mesmo conteúdo do buffer. Essa contenção pode ser mais evidente em tabelas com vários índices.

Os dados da workload não estão na memória

Quando os dados que a workload ativa está processando não estão na memória, esses eventos de espera podem aumentar. Esse efeito ocorre porque os processos que mantêm bloqueios podem fazer isso por mais tempo enquanto executam operações de E/S de disco.

Uso excessivo de restrições de chaves externas

Restrições de chave externas podem aumentar o tempo durante o qual um processo mantém um bloqueio de conteúdo de buffer. Esse efeito ocorre porque operações de leitura exigem um bloqueio de conteúdo de buffer compartilhado na chave referenciada enquanto esta está sendo atualizada.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera. Você pode identificar eventos `LWLock:buffer_content` (`BufferContent`) utilizando o Amazon RDS Performance Insights ou consultando a visualização `pg_stat_activity`.

Tópicos

- Melhorar a eficiência na memória
- Reduzir o uso de restrições de chaves externas
- Remover índices não utilizados
- Remover índices duplicados
- Excluir ou REINDEXAR índices inválidos
- Usar índices parciais
- Remover o inchaço de tabelas e índices

Melhorar a eficiência na memória

Para aumentar as chances de que os dados da workload ativa estejam na memória, particione tabelas ou aumente a escala da sua classe de instância na vertical. Para obter informações sobre classes de instância de banco de dados, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Monitore a métrica `BufferCacheHitRatio`, que mede a porcentagem de solicitações atendidas pelo cache do buffer de uma instância de banco de dados em seu cluster de banco de dados. Essa

métrica fornece insights sobre a quantidade de dados que estão sendo obtidos da memória. Uma alta taxa de acerto indica que a instância de banco de dados tem memória suficiente disponível para seu conjunto de dados de trabalho, enquanto uma taxa baixa sugere que suas consultas acessam dados do armazenamento com frequência.

O acerto de leitura do cache por tabela e o acerto de leitura do cache por índice na seção de configuração de memória do relatório do PG Collector podem fornecer insights sobre a taxa de acerto do cache de tabelas e índices.

Reduzir o uso de restrições de chaves externas

Investigue workloads com um número elevado de eventos de espera `LWLock:buffer_content` (`BufferContent`) quanto ao uso de restrições de chaves externas. Remova restrições desnecessárias de chaves externas.

Remover índices não utilizados

Para workloads com um número elevado de eventos de espera `LWLock:buffer_content` (`BufferContent`), identifique índices não utilizados e remova-os.

A seção de índices não utilizados do relatório do PG Collector pode fornecer insights sobre os índices não utilizados no banco de dados.

Remover índices duplicados

Identifique índices duplicados e remova-os.

A seção de índices duplicados do relatório do PG Collector pode fornecer insights sobre os índices duplicados no banco de dados.

Excluir ou REINDEXAR índices inválidos

Os índices inválidos geralmente se apresentam ao usar `CREATE INDEX CONCURRENTLY` ou `REINDEX CONCURRENTLY` e quando o comando falha ou é cancelado.

Embora esses índices não possam ser usados para consultas, eles serão atualizados e ocuparão espaço em disco.

A seção de índices inválidos do relatório do PG Collector pode fornecer insights sobre os índices inválidos no banco de dados.

Usar índices parciais

Os índices parciais podem ser aproveitados para aprimorar o desempenho da consulta e reduzir o tamanho do índice. Um índice parcial é um índice criado sobre um subconjunto de uma tabela definido por uma expressão condicional. Conforme detalhado na documentação do índice parcial, os índices parciais podem reduzir os custos indiretos de manutenção de índices, pois o PostgreSQL não precisa atualizar o índice em todos os casos.

Remover o inchaço de tabelas e índices

O inchaço de tabelas e índices pode afetar negativamente o desempenho do banco de dados. Tabelas e índices grandes demais aumentam o tamanho do conjunto de trabalho ativo, degradando a eficiência na memória. Além disso, o inchaço aumenta os custos de armazenamento e causa lentidão na execução de consultas. Para diagnosticar inchaços, consulte Diagnosticar a sobrecarga na tabela e no índice. Além disso, a seção de fragmentação (inchaço) do relatório do PG Collector pode fornecer insights sobre o inchaço de tabelas e índices.

Para lidar com o inchaço de tabelas e índices, há algumas opções:

VACUUM FULL

> `VACUUM FULL` cria uma cópia da tabela, copiando somente as tuplas ativas e, em seguida, substitui a tabela antiga pela nova enquanto mantém um bloqueio `ACCESS EXCLUSIVE`. Isso evita qualquer leitura ou gravação na tabela, o que pode causar uma interrupção. Além disso, `VACUUM FULL` levará mais tempo se a tabela for grande.

pg_repack

> O `pg_repack` é útil em situações em que `VACUUM FULL` talvez não seja adequado. Ele cria uma tabela que contém os dados da tabela inchada, monitora as alterações na tabela original e, em seguida, substitui a tabela original pela nova. Ele não bloqueia a tabela original para operações de leitura ou gravação enquanto está criando a outra tabela. Para ter mais informações sobre como usar `pg_repack`, consulte Removing bloat with pg_repack e pg_repack.

REINDEX

> O comando `REINDEX` pode ser usado para lidar com o inchaço do índice. `REINDEX` grava uma nova versão do índice sem as páginas inativas ou as páginas vazias ou quase vazias, reduzindo assim o consumo de espaço do índice. Para ter informações detalhadas sobre o comando `REINDEX`, consulte a documentação do REINDEX.

Depois de remover o inchaço das tabelas e índices, pode ser necessário aumentar a frequência de autovacuum nessas tabelas. A implementação de configurações agressivas de autovacuum em nível de tabela pode ajudar a evitar a ocorrência de futuros inchaços. Para ter mais informações, consulte a documentação em `Vacuuming and analyzing tables automatically`.

LWLock:buffer_mapping

Esse evento ocorre quando uma sessão está aguardando para associar um bloco de dados a um buffer no grupo de buffer compartilhado.

> **Note**
>
> Esse evento aparece como `LWLock:buffer_mapping` no Aurora PostgreSQL versão 12 e inferiores e como `LWLock:BufferMapping` na versão 13 e superiores.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Causas
- Ações

Versões compatíveis do mecanismo

As informações sobre eventos de espera são relevantes para o Aurora PostgreSQL versão 9.6 e versões superiores.

Contexto

O grupo de buffer compartilhado é uma área de memória do Aurora PostgreSQL que contém todas as páginas que estão ou estavam sendo utilizadas por processos. Quando um processo precisa de uma página, ele lê a página no grupo de buffer compartilhado. O parâmetro `shared_buffers` define o tamanho do buffer compartilhado e reserva uma área de memória para armazenar a tabela e as páginas de índice. Se você alterar esse parâmetro, assegure-se de reiniciar o banco de dados. Para obter mais informações, consulte Buffers compartilhados.

O evento de espera `LWLock:buffer_mapping` ocorre nos seguintes cenários:

- Um processo pesquisa a tabela de buffer em busca de uma página e adquire um bloqueio de mapeamento de buffer compartilhado.
- Um processo carrega uma página no grupo de buffer e adquire um bloqueio exclusivo de mapeamento de buffer.
- Um processo remove uma página do grupo e adquire um bloqueio exclusivo de mapeamento de buffer.

Causas

Quando esse evento aparece mais do que o normal, possivelmente indicando um problema de performance, o banco de dados está paginando dentro e fora do grupo de buffer compartilhado. As causas típicas incluem:

- Consultas grandes
- Índices e tabelas inchados
- Varreduras completas de tabelas
- Um tamanho de grupo compartilhado menor que o conjunto de trabalho

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Monitorar métricas relacionadas ao buffer
- Avaliar sua estratégia de indexação
- Diminuir o número de buffers que devem ser alocados rapidamente

Monitorar métricas relacionadas ao buffer

Quando as esperas `LWLock:buffer_mapping` atingirem picos, investigue a taxa de acertos de buffer. É possível utilizar essas métricas para entender melhor o que está acontecendo no cache de buffer. Examine as métricas a seguir:

`BufferCacheHitRatio`

Essa métrica do Amazon CloudWatch mede a porcentagem de solicitações atendidas pelo cache de buffer de uma instância de banco de dados no seu cluster de banco de dados. Você pode ver essa métrica diminuir na preparação para o evento de espera `LWLock:buffer_mapping`.

`blks_hit`

Essa métrica de contador do Performance Insights indica o número de bloqueios que foram recuperados do grupo de buffer compartilhado. Após o surgimento do evento de espera `LWLock:buffer_mapping`, é possível observar um pico em `blks_hit`.

`blks_read`

Essa métrica de contador do Performance Insights indica o número de bloqueios que exigiram a leitura da E/S no grupo de buffer compartilhado. Você pode observar um pico em `blks_read` em preparação para o evento de espera `LWLock:buffer_mapping`.

Avaliar sua estratégia de indexação

Para confirmar que sua estratégia de indexação não está diminuindo a performance, verifique o seguinte:

Inchaço de índice

Assegure-se de que o índice e o inchaço da tabela não estejam fazendo com que páginas desnecessárias sejam lidas no buffer compartilhado. Se as suas tabelas contiverem linhas não utilizadas, considere arquivar os dados e remover as linhas das tabelas. Em seguida, você poderá reconstruir os índices para as tabelas redimensionadas.

Índices para consultas utilizadas com frequência

Para determinar se você tem os índices ideais, monitore as métricas do mecanismo de banco de dados no Performance Insights. A métrica `tup_returned` mostra o número de linhas lidas. A métrica `tup_fetched` mostra o número de linhas retornadas ao cliente. Se `tup_returned` for significativamente maior que `tup_fetched`, talvez os dados não estejam devidamente indexados. Além disso, as estatísticas da tabela podem não estar atualizadas.

Diminuir o número de buffers que devem ser alocados rapidamente

Para diminuir os eventos de espera `LWLock:buffer_mapping`, tente reduzir o número de buffers que devem ser alocados rapidamente. Uma estratégia é realizar operações em lote menores. Talvez seja possível obter lotes menores particionando tabelas.

LWLock:BufferIO (IPC:BufferIO)

O evento `LWLock:BufferIO` ocorre quando o Aurora PostgreSQL ou o RDS for PostgreSQL aguarda outros processos terminarem suas operações de entrada/saída (E/S) ao tentarem acessar simultaneamente uma página. Sua finalidade é fazer com que a mesma página seja lida no buffer compartilhado.

Tópicos

- Versões de mecanismos relevantes
- Contexto
- Causas
- Ações

Versões de mecanismos relevantes

Essas informações de evento de espera são relevantes para todas as versões do Aurora PostgreSQL. Para o Aurora PostgreSQL 12 e versões anteriores, esse evento de espera é denominado lwlock:buffer_io, ao passo que, no Aurora PostgreSQL versão 13, é denominado lwlock:bufferio. A partir do Aurora PostgreSQL versão 14, o evento de espera BufferIO foi movido do tipo de evento de espera `LWLock` para `IPC` (IPC:BufferIO).

Contexto

Cada buffer compartilhado tem um bloqueio de E/S associado ao evento de espera `LWLock:BufferIO`, todas as vezes que um bloqueio (ou uma página) precisa ser recuperado fora do grupo de buffer compartilhado.

Esse bloqueio é utilizado para lidar com várias sessões que requerem acesso ao mesmo bloco. Esse bloco precisa ser lido de fora do grupo de buffer compartilhado, definido pelo parâmetro `shared_buffers`.

Assim que a página for lida dentro do grupo de buffer compartilhado, o bloqueio de `LWLock:BufferIO` será liberado.

> **Note**
>
> O evento de espera `LWLock:BufferIO` precede o evento de espera IO:DataFileRead. O evento de espera `IO:DataFileRead` ocorre enquanto os dados estão sendo lidos do armazenamento.

Para obter mais informações sobre bloqueios leves, consulte Visão geral de bloqueios.

Causas

Causas comuns do surgimento do evento `LWLock:BufferIO` nas principais esperas incluem:

- Vários backends ou conexões tentando acessar a mesma página para a qual também há uma operação de E/S pendente
- A proporção entre o tamanho do grupo de buffer compartilhado (definido pelo parâmetro `shared_buffers`) e o número de buffers necessários para a workload atual
- O tamanho do grupo de buffer compartilhado não está bem equilibrado com o número de páginas que estão sendo despejadas por outras operações
- Índices grandes ou inchados que exigem que o mecanismo leia mais páginas que o necessário no grupo de buffer compartilhado
- Ausência índices, o que força o mecanismo de banco de dados a ler mais páginas das tabelas que o necessário
- Picos repentinos de conexões de banco de dados tentando realizar operações na mesma página

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera:

- Observe as métricas do Amazon CloudWatch para encontrar uma correlação entre reduções acentuadas nos eventos de espera `BufferCacheHitRatio` e `LWLock:BufferIO`. Esse efeito pode significar que existe uma pequena configuração de buffers compartilhados. Talvez seja necessário aumentá-lo ou aumentar a escala da classe de instância de banco de dados na vertical. Você pode dividir sua workload em mais nós de leitura.
- Verifique se existem índices não utilizados e remova-os.
- Use tabelas particionadas (que também tenham índices particionados). Fazer isso ajuda a manter a reordenação do índice baixo e reduz seu impacto.

- Evite indexar colunas desnecessariamente.
- Evite picos repentinos de conexão de banco de dados utilizando um grupo de conexões.
- Restrinja o número máximo de conexões com o banco de dados como prática recomendada.

LWLock:lock_manager

Esse evento ocorre quando o mecanismo Aurora PostgreSQL mantém a área de memória do bloqueio compartilhado para alocar, verificar e desalocar um bloqueio nos casos em que um bloqueio de caminho rápido não é possível.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

As informações sobre eventos de espera são relevantes para o Aurora PostgreSQL versão 9.6 e versões superiores.

Contexto

Quando você emite uma instrução SQL, o Aurora PostgreSQL registra bloqueios para proteger a estrutura, os dados e a integridade do banco de dados durante operações simultâneas. O mecanismo pode atingir esse objetivo utilizando um bloqueio de caminho rápido ou um bloqueio de caminho que não é rápido. Um bloqueio de caminho que não é rápido é mais caro e gera mais sobrecarga do que um bloqueio de caminho rápido.

Bloqueio de caminho rápido

Para reduzir a sobrecarga de bloqueios que são retirados e liberados com frequência, mas que raramente entram em conflito, os processos de backend podem utilizar o bloqueio de caminho rápido. O banco de dados usa esse mecanismo para bloqueios que atendem aos seguintes critérios:

- Usam o método de bloqueio DEFAULT.

- Representam um bloqueio em uma relação de banco de dados em vez de uma relação compartilhada.
- São bloqueios fracos que provavelmente não entrarão em conflito.
- O mecanismo é capaz de verificar rapidamente se nenhum bloqueio conflitante pode existir.

O mecanismo não pode utilizar o bloqueio rápido de caminho quando uma das seguintes condições é verdadeira:

- O bloqueio não atende aos critérios anteriores.
- Não há mais slots disponíveis para o processo de backend.

Para obter mais informações sobre o bloqueio de caminho rápido, consulte o tópico sobre caminho rápido, no README do gerenciador de bloqueios do PostgreSQL epg-locks na documentação do PostgreSQL.

Exemplo de um problema de escalabilidade para o gerenciador de bloqueios

Neste exemplo, uma tabela chamada `purchases` armazena cinco anos de dados, particionados por dia. Cada partição possui dois índices. A seguinte sequência de eventos ocorre:

1. Você consulta muitos dias de dados, o que exige que o banco de dados leia várias partições.
2. O banco de dados cria uma entrada de bloqueio para cada partição. Se os índices de partição fizerem parte do caminho de acesso do otimizador, o banco de dados também criará uma entrada de bloqueio para eles.
3. Quando o número de entradas de bloqueios solicitadas para o mesmo processo de backend for maior que 16, que é o valor de `FP_LOCK_SLOTS_PER_BACKEND`, o gerenciador de bloqueio usará o método de bloqueio de caminho não rápido.

Aplicações modernas podem ter centenas de sessões. Se sessões simultâneas estiverem consultando o pai sem a devida limpeza da partição, o banco de dados poderá criar centenas ou até milhares de bloqueios de caminho não rápidos. Em geral, quando essa simultaneidade é maior que o número de vCPUs, o evento de espera `LWLock:lock_manager` é exibido.

> **Note**
>
> O evento de espera `LWLock:lock_manager` não está relacionado ao número de partições ou índices em um esquema de banco de dados e sim ao número de bloqueios de caminho não rápidos que o banco de dados deve controlar.

Possíveis causas do maior número de esperas

Quando o evento de espera `LWLock:lock_manager` ocorre mais do que o normal, possivelmente indicando um problema de performance, as causas mais prováveis de picos súbitos são:

- Sessões ativas simultâneas estão executando consultas que não utilizam bloqueios de caminho rápido. Essas sessões também excedem a vCPU máxima.
- Um número elevado de sessões ativas simultâneas está acessando uma tabela fortemente particionada. Cada partição possui vários índices.
- O banco de dados está passando por uma tempestade de conexões. Por padrão, algumas aplicações e softwares de grupo de conexões criam mais conexões quando o banco de dados está lento. Essa prática piora o problema. Ajuste o software do grupo de conexões para que tempestades de conexões não ocorram.
- Um número elevado de sessões consulta uma tabela pai sem separar partições.
- Um comando de linguagem de definição de dados (DDL), linguagem de manipulação de dados (DML) ou manutenção bloqueia exclusivamente uma relação ocupada ou tuplas que são frequentemente acessadas ou modificadas.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

Tópicos

- Usar a limpeza de partição
- Remover índices desnecessários
- Ajustar suas consultas para bloqueio de caminho rápido
- Fazer ajustes com base em eventos de espera
- Reduzir gargalos de hardware
- Usar um agrupador de conexões

- Fazer upgrade da versão do Aurora PostgreSQL

Usar a limpeza de partição

Limpeza de partição é uma estratégia de otimização de consultas que exclui partições desnecessárias das varreduras de tabelas, melhorando assim a performance. Por padrão, a remoção de partição está ativada. Se ela estiver desativada, ative-a da seguinte maneira.

```
SET enable_partition_pruning = on;
```

As consultas podem tirar proveito da limpeza de partição quando suas cláusula `WHERE` contêm a coluna utilizada para o particionamento. Para obter mais informações, consulte o tópico sobre Limpeza de partição, na documentação do PostgreSQL.

Remover índices desnecessários

Seu banco de dados pode conter índices não utilizados ou raramente utilizados. Se esse for o caso, considere excluí-los. Realize um dos procedimentos a seguir:

- Saiba mais sobre como encontrar índices desnecessários lendo o tópico sobre Índices não utilizados na wiki do PostgreSQL.
- Execute o PG Collector. Esse script SQL reúne informações do banco de dados e as apresenta em um relatório HTML consolidado. Confira a seção “Índices não utilizados”. Para obter mais informações, consulte pg-collector no Repositório AWS Labs GitHub.

Ajustar suas consultas para bloqueio de caminho rápido

Para descobrir se as suas consultas usam o bloqueio de caminho rápido, consulte a coluna `fastpath` na tabela `pg_locks`. Se as suas consultas não estiverem utilizando o bloqueio de caminho rápido, tente reduzir o número de relações por consulta para menos de 16.

Fazer ajustes com base em eventos de espera

Se `LWLock:lock_manager` for o primeiro ou o segundo na lista de principais esperas, verifique se os seguintes eventos de espera também aparecem na lista:

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

Se os eventos anteriores aparecerem em posição elevada na lista, considere ajustar esses eventos de espera primeiro. Esses eventos podem ser um fator impulsionador para `LWLock:lock_manager`.

Reduzir gargalos de hardware

É possível que haja um gargalo de hardware, como falta de CPU ou uso máximo da largura de banda do Amazon EBS. Nesses casos, considere reduzir gargalos de hardware. Considere as ações a seguir:

- Aumente a escala da sua classe de instância na vertical.
- Otimize consultas que consomem grandes quantidades de CPU e memória.
- Altere a lógica da aplicação.
- Arquive os dados.

Para obter mais informações sobre CPU, memória e largura de banda da rede do EBS, consulte Tipos de instâncias do Amazon RDS.

Usar um agrupador de conexões

Se o número total de conexões ativas exceder o máximo de vCPU, mais processos do SO exigirão CPU do que o tipo de instância pode suportar. Nesse caso, considere utilizar ou ajustar um grupo de conexões. Para obter mais informações sobre as vCPUs para o seu tipo de instância, consulte o tópico sobre Tipos de instância do Amazon RDS.

Para obter mais informações sobre agrupamento de conexões, consulte os seguintes recursos:

- Amazon RDS Proxypara Aurora
- pgbouncer
- Grupos conexões e fontes de dados, na Documentação do PostgreSQL

Fazer upgrade da versão do Aurora PostgreSQL

Se a versão atual do Aurora PostgreSQL for menor que 12, atualize para a versão 12 ou superior. As versões 12 e 13 do PostgreSQL possuem um mecanismo de partição aprimorado. Para obter mais informações sobre a versão 12, consulte Notas de release do PostgreSQL 12.0. Para obter mais informações sobre como fazer upgrade do Aurora PostgreSQL, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL.

LWLock:MultiXact

Os eventos de espera `LWLock:MultiXactMemberBuffer`, `LWLock:MultiXactOffsetBuffer`, `LWLock:MultiXactMemberSLRU` e `LWLock:MultiXactOffsetSLRU` indicam que uma sessão está aguardando para recuperar uma lista de transações que modificam a mesma linha em uma tabela específica.

- `LWLock:MultiXactMemberBuffer`: um processo está aguardando a E/S em um buffer utilizado com menos frequência (SLRU) para um membro multixact.
- `LWLock:MultiXactMemberSLRU`: um processo está aguardando para acessar o cache utilizado com menos frequência (SLRU) para um membro multixact.
- `LWLock:MultiXactOffsetBuffer`: um processo está aguardando a E/S em um buffer utilizado com menos frequência (SLRU) para um desvio multixact.
- `LWLock:MultiXactOffsetSLRU`: um processo está aguardando para acessar o cache utilizado com menos frequência (SLRU) para um desvio multixact.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Contexto

Multixact é uma estrutura de dados que armazena uma lista de IDs de transação (XIDs) que modificam a mesma linha da tabela. Quando uma única transação faz referência a uma linha em uma tabela, o ID da transação é armazenado na linha de cabeçalho da tabela. Quando várias transações fazem referência à mesma linha em uma tabela, a lista de IDs de transação é armazenada na estrutura de dados multixact. Os eventos de espera multixact indicam que uma sessão está recuperando da estrutura de dados a lista de transações que se referem a uma determinada linha em uma tabela.

Possíveis causas do maior número de esperas

Estas são três causas comuns para uso do multixact:

- Subtransações de pontos de salvamento explícitos: a criação explícita de um ponto de salvamento nas transações gera novas transações para a mesma linha. Por exemplo, usando `SELECT FOR UPDATE`, `SAVEPOINT` e, então `UPDATE`.

  Alguns drivers, mapeamentos de objetos relacionais (ORMs) e camadas de abstração têm opções de configuração para envolver automaticamente todas as operações com pontos de salvamento. Isso pode gerar muitos eventos de espera multiexact em algumas workloads. A opção `autosave` do driver JDBC do PostgreSQL é um exemplo disso. Para obter mais informações, consulte pgJDBC na documentação do JDBC do PostgreSQL. Outro exemplo é o driver ODBC do PostgreSQL e sua opção `protocol`. Para obter mais informações, consulte psqlODBC Configuration Options (Opções de configuração do psqlODBC) na documentação do driver ODBC do PostgreSQL.
- Subtransações de cláusulas EXCEPTION de PL/pgSQL: cada cláusula `EXCEPTION` que você escreve em funções ou procedimentos de PL/pgSQL cria um `SAVEPOINT` interno.
- Chaves externas: várias transações adquirem um bloqueio compartilhado no registro pai.

Quando uma determinada linha é incluída em uma operação de várias transações, o processamento da linha exige a recuperação de IDs de transação das listas `multixact`. Se as pesquisas não conseguirem obter o multixact do cache de memória, a estrutura de dados deverá ser lida da camada de armazenamento do Aurora. Essa E/S do armazenamento significa que as consultas de SQL podem demorar mais. As falhas no cache de memória podem começar a ocorrer com o uso intenso devido a um grande número de transações múltiplas. Todos esses fatores contribuem para o aumento desse evento de espera.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera. Algumas dessas ações podem ajudar na redução imediata dos eventos de espera. Porém, outras podem precisar de investigação e correção para escalar a workload.

Tópicos

- Realizar a operação vacuum freeze nas tabelas com este evento de espera
- Aumentar a frequência da limpeza automática das tabelas com esse evento de espera

- Aumentar os parâmetros de memória
- Reduzir transações de longa execução
- Ações de longo prazo

Realizar a operação vacuum freeze nas tabelas com este evento de espera

Se esse evento de espera aumentar repentinamente e afetar o ambiente de produção, você poderá usar qualquer um dos métodos temporários a seguir para reduzir a contagem.

- Use VACUUM FREEZE na tabela ou na partição da tabela afetada para resolver o problema imediatamente. Para ter mais informações, consulte VACUUM.
- Use a cláusula VACUUM (FREEZE, INDEX_CLEANUP FALSE) para realizar uma limpeza rápida ignorando os índices. Para ter mais informações, consulte Aspirar uma tabela o mais rápido possível.

Aumentar a frequência da limpeza automática das tabelas com esse evento de espera

Depois de verificar todas as tabelas em todos os bancos de dados, a operação VACUUM removerá multixacts, e os valores multixact mais antigos avançarão. Para ter mais informações, consulte Multixacts e Wraparound. Para reduzir ao mínimo os eventos de espera LWLock:MultiXact, é necessário executar a operação VACUUM sempre que necessário. Para fazer isso, garanta que o VACUUM no cluster de banco de dados do Aurora PostgreSQL esteja configurado de forma ideal.

Se o uso de VACUUM FREEZE na tabela ou na partição da tabela afetada resolver o problema do evento de espera, recomendamos usar um programador, como o `pg_cron`, para realizar o VACUUM em vez de ajustar o autovacuum na instância.

Para que o autovacuum ocorra com maior frequência, é possível reduzir o valor do parâmetro de armazenamento `autovacuum_multixact_freeze_max_age` na tabela afetada. Para ter mais informações, consulte autovacuum_multixact_freeze_max_age.

Aumentar os parâmetros de memória

Você pode otimizar o uso da memória para caches multixact ajustando os seguintes parâmetros. Essas configurações controlam a quantidade de memória reservada para esses caches, o que pode ajudar a reduzir os eventos de espera multixact na workload. Recomendamos começar com os valores a seguir:

Para o Aurora PostgreSQL 17 e posterior:

- `multixact_offset_buffers` = 128
- `multixact_member_buffers` = 256

Para o Aurora PostgreSQL 16 e posterior:

- `multixact_offsets_cache_size` = 128
- `multixact_members_cache_size` = 256

> **Note**
>
> No Aurora PostgreSQL 17, os nomes dos parâmetros foram alterados de `multixact_offsets_cache_size` para `multixact_offset_buffers` e de `multixact_members_cache_size` para `multixact_member_buffers` para se alinharem à comunidade do PostgreSQL 17.

É possível definir esses parâmetros no nível do cluster para que todas as instâncias do cluster permaneçam consistentes. Recomendamos que você teste e ajuste os valores para melhor atender aos requisitos específicos da workload e à classe de instância. É necessário reinicializar a instância do gravador para que a alteração do parâmetro tenha efeito.

Os parâmetros são expressos em termos de entradas de cache multixact. Cada entrada de cache usa `8 KB` de memória. Para calcular a memória total reservada, multiplique o valor de cada parâmetro por `8 KB`. Por exemplo, se você definir um parâmetro como 128, o total de memória reservada será `128 * 8 KB = 1 MB`.

Reduzir transações de longa execução

Transações de longa duração fazem com que o vácuo retenha as informações até que a transação seja confirmada ou até que a transação somente leitura seja fechada. Recomendamos que você monitore e gerencie transações de longa duração de maneira proativa. Para obter mais informações, consulte O banco de dados está inativo há muito tempo na conexão da transação. Tente modificar a aplicação para evitar ou minimizar o uso de transações de longa duração.

Ações de longo prazo

Examine a workload para descobrir a causa do transbordamento de multixact. É necessário corrigir o problema para escalar a workload e reduzir o evento de espera.

- É necessário analisar a DDL (linguagem de definição de dados) usada para criar tabelas. Verifique se as estruturas e os índices das tabelas estão bem projetados.
- Quando as tabelas afetadas tiverem chaves externas, determine se elas são necessárias ou se há outra forma de aplicar a integridade referencial.
- Quando uma tabela tem grandes índices não utilizados, isso pode fazer com que o autovacuum não se ajuste à workload e pode impedir a execução. Para evitar isso, verifique se há índices não utilizados e remova-os completamente. Para ter mais informações, consulte Gerenciar o autovacuum com grandes índices.
- Reduza o uso de pontos de salvamento nas transações.

LWLock:pg_stat_statements

O evento de espera LWLock:pg_stat_statements ocorre quando a extensão `pg_stat_statements` obtém um bloqueio exclusivo na tabela de hash que rastreia instruções SQL. Isso acontece nos seguintes cenários:

- Quando o número de instruções rastreadas atinge o valor configurado do parâmetro `pg_stat_statements.max` e é necessário abrir espaço para mais entradas, a extensão executa uma classificação no número de chamadas, remove 5% das instruções menos executadas e preenche novamente o hash com as entradas restantes.
- Quando `pg_stat_statements` executa uma operação `garbage collection` no arquivo `pgss_query_texts.stat` no disco e reescreve o arquivo.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

As informações sobre esse evento de espera são aceitas em todas as versões do Aurora PostgreSQL.

Contexto

Conceitos básicos sobre a extensão pg_stat_statements: essa extensão rastreia as estatísticas de execução de instruções SQL em uma tabela de hash. Ela rastreia as instruções SQL até o limite definido pelo parâmetro `pg_stat_statements.max`. Esse parâmetro determina o número máximo de instruções que podem ser rastreadas, o qual corresponde ao número máximo de linhas na visualização pg_stat_statements.

Persistência das estatísticas de instrução: a extensão mantém as estatísticas da instrução em todas as reinicializações da instância:

- Gravando dados em um arquivo chamado pg_stat_statements.stat.
- Usando o parâmetro pg_stat_statements.save para controlar o comportamento de persistência.

Quando pg_stat_statements.save é definido como:

- ativado (padrão): as estatísticas são salvas no desligamento e recarregadas na inicialização do servidor.
- desativado: as estatísticas não são salvas no desligamento nem recarregadas na inicialização do servidor.

Armazenamento de texto de consulta: a extensão armazena o texto das consultas rastreadas em um arquivo chamado `pgss_query_texts.stat`. Esse arquivo pode aumentar até o dobro do tamanho médio de todas as instruções SQL rastreadas antes que a coleta de resíduos ocorra. A extensão requer um bloqueio exclusivo na tabela de hash durante as operações de limpeza e regravação do arquivo `pgss_query_texts.stat`.

Processo de desalocação de instruções: quando o número de instruções rastreadas atinge o limite `pg_stat_statements.max` e novas instruções precisam ser rastreadas, a extensão:

- Obtém um bloqueio exclusivo (LWLock:pg_stat_statements) na tabela de hash.
- Carrega os dados existentes na memória local.
- Executa uma classificação rápida com base no número de chamadas.
- Remove as instruções menos chamadas (5% inferiores).
- Preenche novamente a tabela de hash com as entradas restantes.

Monitoramento da desalocação de instruções: no PostgreSQL 14 e versões posteriores, você pode monitorar a desalocação de instruções usando a visualização pg_stat_statements_info. Essa visualização inclui uma coluna de desalocação que mostra quantas vezes as instruções foram desalocadas para abrir espaço para novas.

Se a desalocação de instruções ocorrer com frequência, haverá uma coleta de resíduos mais frequente do arquivo `pgss_query_texts.stat` no disco.

Possíveis causas do maior número de esperas

As causas típicas do aumento de esperas `LWLock:pg_stat_statements` incluem:

- Um aumento no número de consultas exclusivas usadas pela aplicação.
- O valor do parâmetro `pg_stat_statements.max` é pequeno em comparação ao número de consultas exclusivas que estão sendo usadas.

Ações

Recomenda-se ações distintas, dependendo dos motivos do evento de espera. Você pode identificar eventos `LWLock:pg_stat_statements` utilizando o Insights de Performance do Amazon RDS ou consultando a visualização `pg_stat_activity`.

Ajuste os parâmetros `pg_stat_statements` a seguir para controlar o comportamento de rastreamento e reduzir os eventos de espera das instruções LWLock:pg_stat_.

Tópicos

- Desabilitar o parâmetro pg_stat_statements.track
- Aumentar o parâmetro pg_stat_statements.max
- Desabilitar o parâmetro pg_stat_statements.track_utility

Desabilitar o parâmetro pg_stat_statements.track

Se o evento de espera LWLock:pg_stat_statements estiver afetando negativamente o desempenho do banco de dados e for necessária uma solução rápida antes de uma análise mais aprofundada da visualização `pg_stat_statements` para identificar a causa-raiz, o parâmetro `pg_stat_statements.track` poderá ser desabilitado configurando-o como `none`. Isso desabilitará a coleta de estatísticas de instrução.

Aumentar o parâmetro pg_stat_statements.max

Para reduzir a desalocação e minimizar a coleta de resíduos do arquivo `pgss_query_texts.stat` no disco, aumente o valor do parâmetro `pg_stat_statements.max`. O valor padrão é `5,000`.

> **Note**
>
> O parâmetro `pg_stat_statements.max` é estático. É necessário reinicializar a instância de banco de dados para aplicar qualquer alteração nesse parâmetro.

Desabilitar o parâmetro pg_stat_statements.track_utility

Você pode analisar a visualização pg_stat_statements para determinar quais comandos do utilitário estão consumindo mais recursos rastreados por `pg_stat_statements`.

O parâmetro `pg_stat_statements.track_utility` controla se o módulo rastreia os comandos do utilitário, que incluem todos os comandos, exceto SELECT, INSERT, UPDATE, DELETE e MERGE. Esse parâmetro é definido como por padrão on.

Por exemplo, quando sua aplicação usa muitas consultas de ponto de salvamento, que são inerentemente exclusivas, ela pode aumentar a desalocação de instruções. Para resolver isso, você pode desabilitar o parâmetro `pg_stat_statements.track_utility` para impedir que `pg_stat_statements` rastreie consultas de pontos de salvamento.

> **Note**
>
> O parâmetro `pg_stat_statements.track_utility` é dinâmico. Você pode alterar o respectivo valor sem reiniciar a instância de banco de dados.

Example Exemplo de consultas exclusivas de ponto de salvamento em pg_stat_statements

```
                    query                         |       queryid
--------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                  | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                    | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                   |  54791337410474486
```

```
SAVEPOINT JDBC_SAVEPOINT_1294466                | 8170064357463507593
ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016      | -33608214779996400
SAVEPOINT JDBC_SAVEPOINT_14185                  | -2175035613806809562
SAVEPOINT JDBC_SAVEPOINT_45837                  | -6201592986750645383
SAVEPOINT JDBC_SAVEPOINT_1324                   | 6388797791882029332
```

O PostgreSQL 17 inclui vários aprimoramentos para o rastreamento de comandos de utilitário:

- Os nomes de ponto de salvamento agora são exibidos como constantes.
- Os IDs de transação global (GIDs) dos comandos de confirmação em duas fases agora são exibidos como constantes.
- Os nomes das instruções DEALLOCATE são mostrados como constantes.
- Os parâmetros CALL agora são exibidos como constantes.

Tempo limite:PgSleep

O evento `Timeout:PgSleep` ocorre quando um processo do servidor chama a função `pg_sleep` e está aguardando o tempo limite de suspensão expirar.

Tópicos

- Versões compatíveis do mecanismo
- Possíveis causas do maior número de esperas
- Ações

Versões compatíveis do mecanismo

Essas informações de eventos de espera têm suporte para todas as versões do Aurora PostgreSQL.

Possíveis causas do maior número de esperas

Esse evento de espera ocorre quando uma aplicação, uma função armazenada ou um usuário emite uma instrução SQL que chama uma das seguintes funções:

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

As funções anteriores atrasarão a execução até que o número especificado de segundos tenha decorrido. Por exemplo, `SELECT pg_sleep(1)` pausa por 1 segundo. Para obter mais informações, consulte Atrasar a execução na documentação do PostgreSQL.

Ações

Identifique a instrução que estava executando a função `pg_sleep`. Determine se o uso da função é apropriado.

Ajustar o Aurora PostgreSQL com insights proativos do Amazon DevOps Guru

Os insights proativos do DevOps Guru detectam condições em clusters de banco de dados do Aurora PostgreSQL que podem causar problemas, permitindo que você as conheça antes que ocorram. Os insights proativos podem alertar quando ocorre um período longo de inatividade na conexão da transação. Para ter mais informações sobre como solucionar períodos longos de inatividade nas conexões das transações, consulte O banco de dados está inativo há muito tempo na conexão da transação.

O DevOps Guru pode fazer o seguinte:

- Evitar muitos problemas comuns de bancos de dados ao comparar a configuração do seu banco de dados com as configurações comuns recomendadas.
- Alertar sobre problemas críticos em sua frota que, se não forem controlados, poderão causar problemas maiores no futuro.
- Alertar sobre problemas recém-descobertos.

Cada insight proativo contém uma análise da causa do problema e recomendações de ações corretivas.

Para ter informações sobre o Amazon DevOps Guru para Amazon RDS, consulte Analisar anomalias de desempenho do Aurora com o Amazon DevOps Guru para Amazon RDS.

O banco de dados está inativo há muito tempo na conexão da transação

A conexão com o banco de dados está no estado `idle in transaction` há mais de 1.800 segundos.

Tópicos

- Versões compatíveis do mecanismo
- Contexto
- Causas prováveis desse problema
- Ações
- Métricas relevantes

Versões compatíveis do mecanismo

Essas informações de insights são aceitas em todas as versões do Aurora PostgreSQL.

Contexto

Uma transação no estado `idle in transaction` pode conter bloqueios que impedem outras consultas. Também pode impedir que `VACUUM` (incluindo o autovacuum) limpe linhas mortas, causando inchaço no índice ou na tabela ou recorrência de IDs de transação.

Causas prováveis desse problema

Uma transação iniciada em uma sessão interativa com BEGIN ou START TRANSACTION não terminou usando um comando COMMIT, ROLLBACK ou END. Isso faz com que a transação passe para o estado `idle in transaction`.

Ações

Você pode encontrar transações inativas consultando `pg_stat_activity`.

Em seu cliente SQL, execute a seguinte consulta para listar todas as conexões em estado `idle in transaction` e ordená-las por duração:

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

Recomendamos ações distintas dependendo dos motivos do insight.

Tópicos

- Encerrar transação
- Encerrar a conexão
- Configurar o parâmetro idle_in_transaction_session_timeout
- Verificar o status de AUTOCOMMIT
- Verificar a lógica da transação no código da aplicação

Encerrar transação

Quando você inicia uma transação em uma sessão interativa com BEGIN ou START TRANSACTION, ela passa para o estado `idle in transaction`. Ela permanecerá nesse estado até você encerrar a transação emitindo um comando COMMIT, ROLLBACK ou END, ou até desfazer completamente a conexão para reverter a transação.

Encerrar a conexão

Encerre a conexão com uma transação inativa usando a seguinte consulta:

```
SELECT pg_terminate_backend(pid);
```

pid é o ID de processo da conexão.

Configurar o parâmetro idle_in_transaction_session_timeout

Configure o parâmetro `idle_in_transaction_session_timeout` no grupo de parâmetros. A vantagem de configurar esse parâmetro é que ele não requer uma intervenção manual para encerrar transações que estão inativas há muito tempo. Para obter mais informações sobre esse parâmetro, consulte a documentação do PostgreSQL.

A mensagem a seguir será relatada no arquivo de log do PostgreSQL depois do encerramento da conexão quando uma conexão permanecer no estado idle_in_transaction por mais tempo do que o tempo especificado.

```
FATAL: terminating connection due to idle in transaction timeout
```

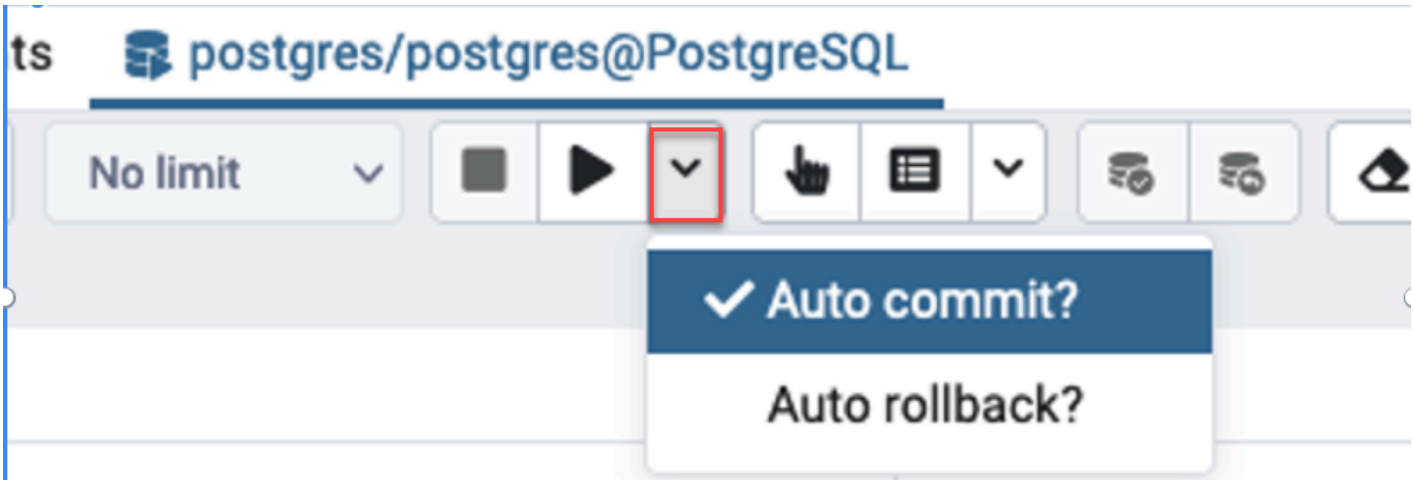
Verificar o status de AUTOCOMMIT

A opção AUTOCOMMIT está ativada por padrão. Se for acidentalmente desativada no cliente, ative-a novamente.

- No cliente psql, execute o seguinte comando:

```
postgres=> \set AUTOCOMMIT on
```

- Em pgadmin, ative-a escolhendo a opção AUTOCOMMIT na seta para baixo.



Verificar a lógica da transação no código da aplicação

Investigue a lógica da aplicação em busca de possíveis problemas. Considere as ações a seguir:

- Verifique se a confirmação automática do JDBC está definida como true na aplicação. Além disso, considere usar comandos `COMMIT` explícitos em seu código.
- Verifique sua lógica de tratamento de erros para ver se ela fecha uma transação depois de um erro.
- Verifique se a aplicação está demorando para processar as linhas retornadas por uma consulta enquanto a transação está aberta. Nesse caso, considere programa a aplicação para fechar a transação antes de processar as linhas.
- Verifique se uma transação contém muitas operações de longa duração. Em caso afirmativo, divida uma única transação em várias transações.

Métricas relevantes

As seguintes métricas de PI estão relacionadas a esse insight:

- idle_in_transaction_count: número de sessões no estado `idle in transaction`.
- idle_in_transaction_max_time: a duração da transação que passou mais tempo no estado `idle in transaction`.

Práticas recomendadas do Amazon Aurora PostgreSQL

A seguir, é possível encontrar várias práticas recomendadas para gerenciar seu cluster de banco de dados do Amazon Aurora PostgreSQL. Lembre-se de rever as tarefas básicas de manutenção também. Para obter mais informações, consulte Performance e escalabilidade do Amazon Aurora PostgreSQL.

Tópicos

- Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora PostgreSQL
- Diagnosticar a sobrecarga na tabela e no índice
- Gerenciar contagens altas de objeto no Amazon Aurora PostgreSQL
- Gerenciamento aprimorado de memória no Aurora PostgreSQL
- Failover rápido com o Amazon Aurora PostgreSQL
- Recuperação rápida após failover com o gerenciamento de cache do cluster para o Aurora PostgreSQL
- Gerenciar a rotatividade de conexão do Aurora PostgreSQL com agrupamento
- Tratamento de conexões inativas no PostgreSQL
- Ajustar parâmetros de memória para o Aurora PostgreSQL
- Usar métricas do Amazon CloudWatch para analisar o uso de recursos do Aurora PostgreSQL
- Usar replicação lógica para realizar uma atualização de versão principal do Aurora PostgreSQL
- Gerenciar conversões personalizadas no Aurora PostgreSQL
- Práticas recomendadas para consultas paralelas no Aurora PostgreSQL
- Solução de problemas de armazenamento no Aurora PostgreSQL

Evitar baixa performance, reinicialização automática e failover para instâncias de banco de dados do Aurora PostgreSQL

Se você estiver executando uma workload pesada ou workloads que ultrapassam os recursos alocados de sua instância de banco de dados, você pode esgotar os recursos nos quais está executando sua aplicação e o banco de dados do Aurora. Para obter métricas em sua instância de banco de dados, como utilização de CPU, uso de memória e número de conexões de banco de dados utilizadas, você pode consultar as métricas fornecidas pelo Amazon CloudWatch,

Performance Insights e Enhanced Monitoring. Para obter informações sobre como monitorar a instância de banco de dados, consulte Monitorar métricas em um cluster do Amazon Aurora.

Se sua workload esgotar os recursos que você está utilizando, sua instância de banco de dados poderá ficar lenta, ser reiniciada ou até mesmo realizar o failover para outra instância de banco de dados. Para evitar isso, monitore a utilização dos recursos, examine a workload em execução em sua instância de banco de dados e faça otimizações quando necessário. Se as otimizações não melhorarem as métricas da instância e mitigarem a exaustão de recursos, considere aumentar a escala verticalmente de sua instância de banco de dados antes de atingir seus limites. Para obter mais informações sobre as classes de instância de banco de dados disponíveis e suas especificações, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Diagnosticar a sobrecarga na tabela e no índice

Você pode usar o controle de simultaneidade de várias versões (MVCC) do PostgreSQL para ajudar a preservar a integridade dos dados. O PostgreSQL MVCC funciona salvando uma cópia interna das linhas atualizadas ou excluídas (também chamadas de tuplas) até que uma transação seja confirmada ou revertida. Essa cópia interna salva é invisível para os usuários. No entanto, pode ocorrer sobrecarga da tabela quando essas cópias invisíveis não são limpas regularmente pelos utilitários VACUUM ou AUTOVACUUM. Se não houver controle, a sobrecarga da tabela pode acarretar maiores custos de armazenamento e diminuir a velocidade de processamento.

Em muitos casos, as configurações padrão para VACUUM ou AUTOVACUUM no Aurora são suficientes para lidar com a sobrecarga indesejada da tabela. No entanto, convém conferir se há sobrecarga se sua aplicação estiver enfrentando as seguintes condições:

- Processa um grande número de transações em um tempo relativamente curto entre os processos de VACUUM.
- Funciona mal e fica sem espaço de armazenamento.

Para começar, reúna as informações mais precisas sobre quanto espaço é usado por tuplas mortas e quanto você pode recuperar limpando a sobrecarga de tabela e índice. Para fazer isso, use a extensão `pgstattuple` para coletar estatísticas sobre o cluster do Aurora. Para ter mais informações, consulte pgstattuple. Os privilégios para usar a extensão `pgstattuple` são limitados aos superusuários do banco de dados e perfil `pg_stat_scan_tables`.

Para criar a extensão `pgstattuple` no Aurora, conecte uma sessão do cliente ao cluster, por exemplo, psql ou pgAdmin, e use o seguinte comando:

```
CREATE EXTENSION pgstattuple;
```

Crie a extensão em cada banco de dados para o qual você deseja criar o perfil. Depois de criar a extensão, use a interface de linha de comando (CLI) para medir a quantidade de espaço inutilizável que você pode recuperar. Antes de coletar estatísticas, modifique o grupo de parâmetros do cluster definindo AUTOVACUUM como 0. Uma configuração de 0 impede que o Aurora limpe automaticamente quaisquer tuplas mortas deixadas pela aplicação, o que pode afetar a precisão dos resultados. Use o seguinte comando para criar uma tabela simples:

```
postgres=> CREATE TABLE lab AS SELECT generate_series (0,100000);
SELECT 100001
```

No exemplo a seguir, executamos a consulta com o AUTOVACUUM ativado para o cluster de banco de dados. O `dead_tuple_count` é 0, o que indica que o AUTOVACUUM removeu dados obsoletos ou tuplas do banco de dados PostgreSQL.

Para usar `pgstattuple` para coletar informações sobre a tabela, especifique o nome de uma tabela ou um identificador de objeto (OID) na consulta:

```
postgres=> SELECT * FROM pgstattuple('lab');
```

```
table_len  | tuple_count | tuple_len | tuple_percent | dead_tuple_count |
 dead_tuple_len | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056    | 100001      | 2800028   | 77.16         | 0                | 0
  | 0                  | 16616      | 0.46
(1 row)
```

Na consulta a seguir, desativamos o AUTOVACUUM e usamos um comando que exclui 25 mil linhas da tabela. Como resultado, a `dead_tuple_count` aumenta para 25 mil.

```
postgres=> DELETE FROM lab WHERE generate_series < 25000;
```

```
DELETE 25000
```

```
SELECT * FROM pgstattuple('lab');
```

```
table_len | tuple_count | tuple_len | tuple_percent | dead_tuple_count | dead_tuple_len
 | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056 | 75001 | 2100028 | 57.87 | 25000 | 700000 | 19.29 | 16616 | 0.46
(1 row)
```

Para recuperar essas tuplas mortas, inicie um processo do VACUUM.

Observar a sobrecarga sem interromper sua aplicação

As configurações em um cluster do Aurora são otimizadas para oferecer as práticas recomendadas para a maioria das workloads. No entanto, talvez você queira otimizar um cluster para melhor atender às suas aplicações e padrões de uso. Nesse caso, você pode usar a extensão `pgstattuple` sem interromper uma aplicação ocupada. Para fazer isso, execute as seguintes etapas:

1. Clone sua instância do Aurora.
2. Modifique o arquivo de parâmetros para desativar o AUTOVACUUM no clone.
3. Execute uma consulta `pgstattuple` ao testar o clone com uma workload de exemplo ou com o pgbench, que é um programa para executar testes de benchmark no PostgreSQL. Para ter mais informações, consulte pgbench.

Depois de executar suas aplicações e visualizar o resultado, use pg_repack ou VACUUM FULL na cópia restaurada e compare as diferenças. Se você observar uma queda significativa no dead_tuple_count, dead_tuple_len ou dead_tuple_percent, ajuste o cronograma de vácuo em seu cluster de produção para minimizar a sobrecarga.

Evitar a sobrecarga em tabelas temporárias

Se a aplicação criar tabelas temporárias, garanta que a aplicação as remova quando elas não forem mais necessárias. Os processos do Autovacuum não localizam tabelas temporárias. Se não forem conferidas, as tabelas temporárias poderão criar sobrecarga rapidamente no banco de dados. Além disso, a sobrecarga pode se estender às tabelas do sistema, que são as tabelas internas que rastreiam objetos e atributos do PostgreSQL, como pg_attribute e pg_depend.

Quando uma tabela temporária não é mais necessária, você pode usar uma instrução TRUNCATE para esvaziá-la e liberar espaço. Depois, limpe manualmente as tabelas pg_attribute e pg_depend. A limpeza dessas tabelas garante que as operações de criar e truncar/excluir tabelas temporárias não esteja adicionando tuplas continuamente e contribuindo para a sobrecarga do sistema.

Você pode evitar esse problema ao criar uma tabela temporária incluindo a seguinte sintaxe, que exclui as novas linhas quando o conteúdo é confirmado:

```
CREATE TEMP TABLE IF NOT EXISTS table_name(table_description) ON COMMIT DELETE ROWS;
```

A cláusula `ON COMMIT DELETE ROWS` trunca a tabela temporária quando a transação é confirmada.

Evitar a sobrecarga nos índices

Quando você altera um campo indexado em uma tabela, a atualização do índice ocasiona uma ou mais tuplas mortas nesse índice. Por padrão, o processo de limpeza automática elimina a sobrecarga nos índices, mas essa limpeza consome uma quantidade significativa de tempo e recursos. Para especificar as preferências de limpeza do índice ao criar uma tabela, inclua a cláusula vacuum_index_cleanup. Por padrão, no momento da criação da tabela, a cláusula é definida como AUTO, o que significa que o servidor decide se seu índice precisa ser limpo ao limpar a tabela. Você pode definir a cláusula como ATIVADA, para ativar a limpeza do índice para uma tabela específica, ou DESATIVADA, para desativar a limpeza do índice dessa tabela. Lembre-se de que desativar a limpeza do índice pode economizar tempo, mas ocasionar um índice sobrecarregado.

Você pode controlar manualmente a limpeza do índice ao limpar uma tabela na linha de comando. Para limpar uma tabela e remover tuplas mortas dos índices, inclua a cláusula INDEX_CLEANUP com um valor ATIVADO e o nome da tabela:

```
acctg=> VACUUM (INDEX_CLEANUP ON) receivables;
```

```
INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Para limpar uma tabela sem limpar os índices, especifique um valor DESATIVADO:

```
acctg=> VACUUM (INDEX_CLEANUP OFF) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Gerenciar contagens altas de objeto no Amazon Aurora PostgreSQL

Embora as limitações do PostgreSQL sejam teóricas, ter contagens de objetos extremamente altas em um banco de dados causará um impacto perceptível no desempenho de várias operações. Esta documentação aborda vários tipos comuns de objeto que podem causar vários impactos possíveis quando a contagem total é alta.

A seguinte tabela apresenta um resumo dos tipos de objeto e dos possíveis impactos.

Tipos de objeto e possíveis impactos

Tipo de objeto	Autovacuum	Replicação lógica	Atualização de versão principal	pg_dump/pg_restore	Desempenho geral	Reinicialização da instância
Relações	x		x	x	x	
Tabelas temporárias	x				x	
Tabelas não registradas		x				x
Partições					x	
Arquivos temporários					x	
Sequências		x				
Objetos grandes		x	x			

Relações

Não há um limite rígido específico em relação ao número de tabelas em um banco de dados PostgreSQL. O limite teórico é extremamente alto, mas há outros limites práticos que precisam ser considerados na fase de design do banco de dados.

Impacto: atraso do autovacuum

O autovacuum pode ter dificuldade de acompanhar o aumento de IDs de transação ou o inchaço de tabela devido à falta de operadores em comparação à quantidade de trabalho.

Ação recomendada: há vários fatores que permitem ajustar o autovacuum para acompanhar adequadamente determinado número de tabelas e determinada workload. Consulte Práticas recomendadas para trabalhar com o autovacuum do PostgreSQL para ver sugestões sobre como determinar as configurações apropriadas do autovacuum. Use o postgres_get_av_diag utility para monitorar problemas de aumento de IDs de transação.

Impacto: atualização da versão principal/pg_dump e pg_restore

O Amazon RDS usa a opção “--link” durante a execução do pg_upgrade para evitar a necessidade de fazer cópias dos arquivos de dados. Os metadados do esquema ainda assim precisam ser restaurados na nova versão do banco de dados. Mesmo com o pg_restore paralelo, se houver um número significativo de relações, isso aumentará o tempo de inatividade.

Impacto: degradação geral do desempenho

Degradação geral do desempenho devido ao tamanho do catálogo. Cada tabela e as colunas correspondentes aumentarão as tabelas `pg_attribute`, `pg_class` e `pg_depend` que são frequentemente usadas em operações normais de banco de dados. Não haverá um evento de espera específico visível, mas a eficiência do buffer compartilhado será afetada.

Ação recomendada: verifique regularmente o inchaço dessas tabelas específicas e, ocasionalmente, execute `VACUUM FULL` nessas tabelas. Lembre-se de que `VACUUM FULL` nas tabelas de catálogo exigem um bloqueio `ACCESS EXCLUSIVE`, o que significa que nenhuma outra consulta poderá acessá-las enquanto a operação não for concluída.

Limite aproximado: milhões.

Tabelas temporárias

O uso de tabelas temporárias, que é útil para dados de teste ou resultados intermediários, é um padrão comum visto em muitos mecanismos de banco de dados. É necessário entender as implicações do uso intenso no PostgreSQL para evitar algumas das armadilhas. Cada tabela temporária criada e descartada adicionará linhas às tabelas de catálogo do sistema, o que causará problemas gerais de desempenho quando essas tabelas ficarem inchadas.

Impacto: atraso do autovacuum

As tabelas temporárias não são limpas pelo autovacuum. Elas ficarão atreladas aos IDs de transação enquanto existirem e poderão provocar wraparound se não forem removidas.

Ação recomendada: manter as tabelas temporárias durante a sessão que as criou ou eliminá-las manualmente. Evitar transações de longa duração com tabelas temporárias é uma prática recomendada que impedirá que essas tabelas contribuam para o aumento máximo de IDs de transação usados.

Impacto: degradação geral do desempenho

Degradação geral do desempenho devido ao tamanho do catálogo. Quando as sessões criam e eliminam continuamente tabelas temporárias, elas aumentam as tabelas `pg_attribute`, `pg_class` e `pg_depend` tabelas que são frequentemente usadas em operações normais de banco de dados. Não haverá um evento de espera específico visível, mas a eficiência do buffer compartilhado será afetada.

Ação recomendada:

- Verifique regularmente o inchaço dessas tabelas específicas e, ocasionalmente, execute `VACUUM FULL` nessas tabelas. Lembre-se de que `VACUUM FULL` nas tabelas de catálogo exigem um bloqueio `ACCESS EXCLUSIVE`, o que significa que nenhuma outra consulta poderá acessá-las enquanto a operação não for concluída.
- Se forem usadas muitas tabelas temporárias, é altamente recomendável executar `VACUUM FULL` nessas tabelas de catálogo específicas antes de uma atualização de versão principal para reduzir o tempo de inatividade.

Práticas recomendadas gerais:

- Reduza o uso de tabelas temporárias utilizando expressões de tabela comuns para produzir resultados intermediários. Às vezes, isso pode complicar as consultas necessárias, mas eliminará os impactos listados acima.
- Reutilize as tabelas temporárias usando o comando `TRUNCATE` para limpar o conteúdo em vez de executar as etapas de descartar/criar. Isso também eliminará o problema de aumento de IDs de transação provocado por tabelas temporárias.

Limite aproximado: dezenas de milhares.

Tabelas não registradas

Tabelas não registradas em log podem oferecer ganhos de desempenho, pois não geram nenhuma informação de registro em log com gravação antecipada (WAL). Visto que elas não oferecerão durabilidade durante a recuperação de falhas no banco de dados porque estarão truncadas, é necessário usá-las com cautela. Essa é uma operação cara no PostgreSQL, pois cada tabela não registrada e log é truncada em série. Embora essa operação seja rápida para um pequeno número de tabelas não registradas em log, quando o número atinge a casa de milhares, isso pode começar a causar um atraso notável durante a inicialização.

Impacto: replicação lógica

> As tabelas não registradas em log geralmente não são incluídas na replicação lógica, inclusive em implantações azuis/verde, porque a replicação lógica depende do WAL para capturar e transferir alterações. Leia mais sobre como configurar o Aurora PostgreSQL para replicar tabelas não registradas em log.

Impacto: nós de leitor

> É possível acessar tabelas não registradas em log somente por meio do nó de gravador no cluster de banco de dados do Aurora. Consulte Trabalhar com tabelas não registradas em log no Aurora PostgreSQL para obter detalhes.

Práticas recomendadas gerais:

- Como as tabelas não registradas em log oferecem benefícios de desempenho limitados no Aurora PostgreSQL em comparação com o PostgreSQL padrão, avalie se as tabelas normais são mais apropriadas.

Limite aproximado: milhares.

Partições

O particionamento pode aumentar o desempenho da consulta e oferecer uma organização lógica dos dados. Em cenários ideais, o particionamento é organizado para que a remoção de partições possa ser usada durante o planejamento e a execução da consulta. Usar muitas partições pode provocar impactos negativos no desempenho das consultas e na manutenção do banco de dados. A escolha de como particionar uma tabela deve ser feita com cuidado, pois o desempenho do planejamento e da execução da consulta pode ser afetado negativamente por um design ineficiente. Consulte a documentação do PostgreSQL para obter detalhes sobre particionamento.

Impacto: degradação geral do desempenho

Algumas vezes os custos indiretos do tempo de planejamento aumentarão e a explicação dos planos referentes às consultas se tornarão mais complexa, dificultando a identificação de oportunidades de ajuste. Em versões do PostgreSQL anteriores à 18, muitas partições com alta workload podem provocar esperas `LWLock:LockManager`.

Ação recomendada: determine um número mínimo de partições que permita concluir a organização dos dados e, ao mesmo tempo, oferecer uma execução de consulta eficiente.

Impacto: manutenção complexa

Um número muito alto de partições provocará dificuldades de manutenção, como predefinição e remoção. O autovacuum tratará as partições como relações normais e deverá realizar uma limpeza regular, exigindo, portanto, operadores suficientes para concluir a tarefa.

Ação recomendada:

- Predefina as partições para que a workload não seja bloqueada quando uma nova partição for necessária (por exemplo, partições mensais) e partições antigas forem removidas.
- Garanta que haja operadores de autovacuum suficientes para realizar a manutenção de limpeza normal de todas as partições.

Limite aproximado:: centenas.

Arquivos temporários

Diferentemente das tabelas temporárias mencionadas acima, o PostgreSQL cria arquivos temporários na eventualidade de uma consulta complexa executar várias operações de classificação

ou de hash simultaneamente e de cada uma dessas operações utilizar a memória da instância para armazenar resultados até o valor especificado no parâmetro `work_mem`. Quando a memória da instância não é suficiente, arquivos temporários são criados para armazenar os resultados. Consulte Gerenciar arquivos temporários para obter mais detalhes sobre arquivos temporários. Se a workload gerar um grande número desses arquivos, poderá haver vários impactos.

Impacto: consumo de FreeLocalStorage

> Os arquivos temporários são uma parte necessária do PostgreSQL quando os resultados da consulta não correspondem ao work_mem. Geralmente, isso é aceitável. No entanto, quando a workload gera arquivos temporários com frequência, isso indica que estão sendo executadas grandes consultas continuamente, caso em que o FreelLocalStorage diminuirá. Consulte Gerenciar arquivos temporários para obter mais detalhes.

Práticas recomendadas gerais:

- Monitore o uso de arquivos temporários com o Insights de Performance.
- Ajuste as consultas que estão gerando arquivos temporários significativos para ver se é possível reduzir o número total de arquivos temporários.

Limite aproximado: milhares.

Sequências

As sequências são o objeto subjacente usado para acrescentar colunas automaticamente no PostgreSQL e oferecem exclusividade e uma chave para os dados. Elas podem ser usadas em tabelas individuais sem problemas durante operações normais, exceto na replicação lógica.

No momento, no PostgreSQL, a replicação lógica não replica o valor atual de uma sequência para nenhum assinante. Para saber mais, consulte a página Restrictions na documentação do PostgreSQL.

Impacto: tempo de transição estendido

> Se você planeja usar implantações azuis/verdes do Amazon RDS para qualquer tipo de alteração ou atualização de configuração, é importante entender o impacto de ter um grande número de sequências na transição. Uma das fases finais da transição sincronizará o valor atual das sequências e, se houver vários milhares, isso aumentará o tempo geral de transição.

Ação recomendada: se a workload do banco de dados permitir o uso de um UUID compartilhado em vez de uma abordagem de sequência por tabela, isso poderá reduzir a etapa de sincronização durante a transição.

Limite aproximado: milhares.

Objetos grandes

Objetos grandes são armazenados em uma única tabela do sistema chamada pg_largeobject. Cada objeto grande também tem uma entrada na tabela do sistema pg_largeobject_metadata. O processo de criação, modificação e limpeza desses objetos é bem diferente do utilizado nas relações padrão. Os objetos grandes não são processados pelo autovacuum e devem ser limpos periodicamente por meio de um processo separado, chamado vacuumlo. Consulte “Gerenciar objetos grandes com o módulo lo” para ver exemplos sobre como gerenciar objetos grandes.

Impacto: replicação lógica

No momento, não é possível replicar objetos grandes no PostgreSQL durante a replicação lógica. Para saber mais, consulte a página Restrictions na documentação do PostgreSQL. Em uma configuração azul/verde, isso significa que objetos grandes no ambiente azul não são replicados no ambiente verde.

Impacto: atualização de versão principal

Uma atualização poderá ficar sem memória e falhar se houver milhões de objetos grandes, e a instância não puder manipulá-los durante uma atualização. O processo de atualização da versão principal do PostgreSQL compreende duas fases amplas: despejar o esquema via pg_dump e restaurá-lo por meio de pg_restore. Se seu banco de dados tiver milhões de objetos grandes, você precisará garantir que sua instância tenha memória suficiente para lidar com o pg_dump e o pg_restore durante uma atualização e escalá-la para um tipo de instância maior.

Práticas recomendadas gerais:

- Use regularmente o utilitário vacuumlo para remover qualquer objeto grande órfão que você possa ter.
- Considere a possibilidade de usar o tipo de dados BYTEA para armazenar objetos grandes no banco de dados.

Limite aproximado: milhões.

Limites aproximados

Os limites aproximados mencionados neste tópico são usados somente para oferecer uma estimativa de até que ponto determinado recurso pode ser escalado. Eles representam o âmbito geral em que os impactos descritos se tornam mais prováveis, mas o comportamento real depende de fatores específicos, como workload, tamanho da instância e configuração. Embora seja possível exceder essas estimativas, é necessário seguir as operações de cuidado e manutenção para evitar os impactos listados.

Gerenciamento aprimorado de memória no Aurora PostgreSQL

O Aurora PostgreSQL agora inclui recursos avançados de gerenciamento de memória para otimizar o desempenho e a resiliência do banco de dados sob workloads variadas. Essas melhorias ajudam o Aurora PostgreSQL a manter disponibilidade e capacidade de resposta consistentes, mesmo durante períodos de alta demanda de memória.

Esse recurso está disponível e habilitado por padrão nas seguintes versões do Aurora PostgreSQL para instâncias provisionadas:

- 15.3 e todas versões secundárias posteriores
- 14.8 e versões secundárias posteriores
- 13.11 e versões secundárias posteriores
- 12.15 e versões secundárias posteriores
- 11.20 e versões secundárias posteriores

Esse recurso está disponível e habilitado por padrão para as seguintes versões do Aurora PostgreSQL para instâncias do Aurora Serverless:

- 16.3 e todas versões secundárias posteriores
- 15.7 e todas versões secundárias posteriores
- 14.12 e versões secundárias posteriores
- 13.15 e versões secundárias posteriores

Quando as workloads do cliente usam toda a memória livre disponível, o sistema operacional pode reiniciar o banco de dados para proteger os recursos, levando à indisponibilidade temporária. As

novas melhorias no gerenciamento de memória no Aurora PostgreSQL cancelam proativamente determinadas transações quando o sistema sofre alta pressão de memória, ajudando a manter a estabilidade do banco de dados.

As principais características do gerenciamento aprimorado de memória são as seguintes:

- Cancela transações de banco de dados que solicitam mais memória quando o sistema está se aproximando de uma demanda crítica de memória.
- Diz-se que o sistema está com demanda crítica de memória quando esgota toda a memória física e está prestes a esgotar a memória swap. Nessas circunstâncias, qualquer transação que solicite memória será cancelada na tentativa de reduzir imediatamente a demanda de memória na instância de banco de dados.
- Os inicializadores essenciais do PostgreSQL e os agentes em segundo plano, como os agentes de autovacuum, estão sempre protegidos.

Lidar com parâmetros de gerenciamento de memória

Como ativar o gerenciamento de memória

Esse recurso está ativado por padrão. Uma mensagem de erro é exibida quando uma transação é cancelada devido à falta de memória, conforme mostrado no exemplo a seguir:

```
ERROR: out of memory Detail: Failed on request of size 16777216.
```

Como desativar o gerenciamento de memória

Para desativar esse recurso, conecte-se ao cluster de banco de dados do Aurora PostgreSQL com psql e use a declaração SET para os valores dos parâmetros, conforme mencionado abaixo.

> **Note**
>
> Recomendamos que você mantenha o gerenciamento de memória habilitado. Isso ajuda a evitar possíveis erros de falta de memória que podem levar a reinicializações do banco de dados induzidas pela workload devido ao esgotamento da memória.

A tabela a seguir mostra como desativar o recurso de gerenciamento de memória das diferentes versões do Aurora PostgreSQL:

Versões do Aurora PostgreSQL	Parâmetro	Padrão	Comando para desativar o gerenciamento de memória no nível da sessão
11.20, 11.21, 12.15, 12.16, 13.11, 13.12, 14.8, 14.9, 15.3, 15.4	`rds.memory_allocation_guard`	`false`	`SET rds.memory_allocation_guard = true;`
12.17, 13.13, 14.10, 15.5 e versões posteriores	`rds.enable_memory_management`	`true`	`SET rds.enable_memory_management = false;`

> **Note**
>
> O parâmetro `rds.memory_allocation_guard` foi descontinuado no Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 e versões posteriores.

Definir os valores desses parâmetros no grupo de parâmetros do cluster de banco de dados evita que as consultas sejam canceladas. Para ter mais informações sobre o grupo de parâmetros do cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Limitação

- Esse recurso não é compatível com a classe de instância db.t3.medium.

Failover rápido com o Amazon Aurora PostgreSQL

A seguir, você vai aprender a garantir que o failover ocorra o mais rápido possível. Para se recuperar rapidamente após o failover, você pode usar o gerenciamento de cache de cluster para o cluster de banco de dados do Aurora PostgreSQL. Para obter mais informações, consulte Recuperação rápida após failover com o gerenciamento de cache do cluster para o Aurora PostgreSQL.

Veja algumas das etapas que podem ser adotadas para que o failover seja executado rapidamente:

- Definir os keepalives do Transmission Control Protocol (TCP) com períodos curtos para interromper a execução de consultas por mais tempo antes que o tempo limite de leitura expire em caso de falha.
- Definir tempos limites agressivamente para o armazenamento em cache do Sistema de Nomes de Domínio (DNS) em Java. Isso ajuda a garantir que o endpoint somente leitura do Aurora possa percorrer corretamente os nós somente leitura em tentativas de conexão posteriores.
- Definir as variáveis de tempo limite usadas na string de conexão JDBC com o valor mais baixo possível. Usar objetos de conexão separados para consultas de curta e longa execução.
- Usar os endpoints do Aurora de leitura e gravação fornecidos para conectar com o cluster.
- Usar as operações da API do RDS para testar a resposta da aplicação em caso de falhas do lado do servidor. Além disso, usar uma ferramenta de descarte de pacotes para testar a resposta da aplicação para falhas do lado do cliente.
- Use o driver JDBC da AWS para aproveitar ao máximo os recursos de failover do Aurora PostgreSQL. Consulte mais informações sobre o driver JDBC da AWS e siga as instruções para usá-lo em Amazon Web Services (AWS) JDBC Driver GitHub repository.

Isso é abordado em mais detalhes a seguir.

Tópicos

- Definir parâmetros de keepalives de TCP
- Configurar a aplicação para failover rápido
- Testar o failover
- Exemplo de failover rápido em Java

Definir parâmetros de keepalives de TCP

Quando você configura uma conexão TCP, um conjunto de timers é associado à conexão. Quando o timer de keepalive chegar a zero, um pacote de sondagem keepalive é enviado ao endpoint de conexão. Se receber uma resposta, você poderá assumir que a conexão ainda está ativa.

Ativar parâmetros de keepalive de TCP e defini-los de forma agressiva garante que, se o seu cliente não conseguir se conectar ao banco de dados, qualquer conexão ativa será rapidamente encerrada. Depois, a aplicação pode se conectar a um novo endpoint.

Defina os seguintes parâmetros de keepalive de TCP:

- `tcp_keepalives_idle` controla o tempo, em segundos, após o qual um pacote de keepalive é enviado quando nenhum dado é enviado pelo soquete. ACKs não são considerados dados. Recomendamos a seguinte configuração:

  ```
  tcp_keepalives_idle = 1
  ```

- `tcp_keepalives_interval` controla o tempo, em segundos, entre o envio de pacotes de keepalive subsequentes depois que o pacote inicial é enviado. Defina esse tempo usando o parâmetro `tcp_keepalives_idle`. Recomendamos a seguinte configuração:

  ```
  tcp_keepalives_interval = 1
  ```

- `tcp_keepalives_count` é o número de sondagens de keepalive não confirmadas que ocorrem antes de a aplicação ser notificada. Recomendamos a seguinte configuração:

  ```
  tcp_keepalives_count = 5
  ```

Essas configurações devem notificar a aplicação em até cinco segundos depois que o banco de dados para de responder. Se os pacotes de keepalive forem frequentemente descartados dentro na rede da aplicação, você poderá definir um valor de `tcp_keepalives_count` maior. Isso permite mais buffer em redes menos confiáveis, embora aumente o tempo necessário para detectar uma falha real.

Como definir parâmetros de keepalive de TCP no Linux

1. Teste como configurar seus parâmetros de keepalive de TCP.

   Recomendamos fazer isso usando a linha de comando com os comandos a seguir. Essa configuração sugerida abrange todo o sistema. Em outras palavras, ela também afeta todas as outras aplicações que criam soquetes com a opção `SO_KEEPALIVE` ativada.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   sudo sysctl net.ipv4.tcp_keepalive_probes=5
   ```

2. Depois de encontrar uma configuração que funcione para a sua aplicação, mantenha essas configurações adicionando as seguintes linhas a `/etc/sysctl.conf`, incluindo as alterações feitas:

   ```
   tcp_keepalive_time = 1
   tcp_keepalive_intvl = 1
   ```

```
tcp_keepalive_probes = 5
```

Configurar a aplicação para failover rápido

A seguir, você encontrará uma discussão sobre as várias alterações de configuração possíveis do Aurora PostgreSQL para obter um failover rápido. Para saber mais sobre a instalação e configuração do driver JDBC PostgreSQL, consulte a documentação do Driver JDBC PostgreSQL.

Tópicos

- Reduzir tempos limite de cache do DNS
- Definir uma string de conexão do Aurora PostgreSQL para failover rápido
- Outras opções para obter a string do host

Reduzir tempos limite de cache do DNS

Quando a aplicação tentar estabelecer uma conexão após um failover, o novo gravador do Aurora PostgreSQL será um leitor anterior. Você pode encontrá-lo usando o endpoint somente leitura do Aurora antes que as atualizações de DNS sejam totalmente propagadas. Definir a vida útil (TTL) do DNS em Java com um valor baixo, como inferior a 30 segundos, ajuda a percorrer os nós de leitor em tentativas de conexão posteriores.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Definir uma string de conexão do Aurora PostgreSQL para failover rápido

Para usar o failover rápido do Aurora PostgreSQL, verifique se a string de conexão da aplicação tem uma lista de hosts, em vez de apenas um host. Veja a seguir um exemplo de string de conexão que você pode usar para se conectar a um cluster do Aurora PostgreSQL. Neste exemplo, os hosts estão em negrito.

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<primaryuser>&password=<primarypw>&loginTimeout=2
```

```
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=primary
```

Para obter a melhor disponibilidade e evitar uma dependência para com a API do RDS, recomendamos manter um arquivo para se conectar com ele. Esse arquivo contém uma string de host que é lida pela aplicação quando você estabelece uma conexão com o banco de dados. Essa string de host tem todos os endpoints do Aurora disponíveis para o cluster. Para obter mais informações sobre endpoints do Aurora, consulte Conexões de endpoints do Amazon Aurora.

Por exemplo, você pode armazenar os endpoints em um arquivo local, conforme mostrado a seguir.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

A aplicação faz a leitura desse arquivo para preencher a seção de host da string de conexão JDBC. Renomear o cluster de banco de dados faz com que esses endpoints sejam alterados. Verifique se a aplicação lidará com esse evento se ele ocorrer.

Outra opção é usar uma lista de nós de instância de banco de dados da seguinte maneira:

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

A vantagem dessa abordagem é que o driver da conexão JDBC PostgreSQL faz um loop em todos os nós dessa lista para encontrar uma conexão válida. Entretanto, quando você usa os endpoints do Aurora, apenas dois nós são testados em cada tentativa de conexão. Mas há uma desvantagem em usar nós de instância de banco de dados. Se você adicionar ou remover nós do cluster e a lista de endpoints de instância ficar obsoleta, o driver de conexão talvez nunca encontre o host correto com o qual se conectar.

Para ajudar a garantir que a aplicação não espere muito tempo para se conectar a qualquer host, defina os seguintes parâmetros de forma agressiva:

- `targetServerType`: controla se o driver se conecta a um nó de gravação ou leitura. Para garantir que as aplicações se reconectem somente a um nó de gravação, defina o valor `targetServerType` como `primary`.

Os valores para o parâmetro `targetServerType` incluem `primary`, `secondary`, `any` e `preferSecondary`. O valor `preferSecondary` primeiro tenta estabelecer uma conexão com um leitor. Ele se conectará ao gravador se nenhuma conexão com o leitor puder ser estabelecida.

- `loginTimeout`: controla o tempo de espera para a aplicação acessar o banco de dados depois de estabelecida uma conexão de soquete.
- `connectTimeout`: controla o tempo de espera do soquete para estabelecer uma conexão com o banco de dados.

Você pode modificar outros parâmetros de aplicações para acelerar o processo de conexão, dependendo do nível de agressividade que você deseja que a aplicação tenha:

- `cancelSignalTimeout`: em algumas aplicações ativas, convém enviar um sinal de cancelamento de “melhor esforço” em uma consulta que atingiu o tempo limite. Se esse sinal de cancelamento estiver no seu caminho de failover, considere uma definição agressiva para evitar enviar esse sinal a um host inativo.
- `socketTimeout`: esse parâmetro controla por quanto tempo o soquete aguarda operações de leitura. Esse parâmetro pode ser usado como um "tempo limite de consulta" global para garantir que nenhuma consulta espere mais do que esse valor. É recomendável ter dois manipuladores de conexão. Um que execute consultas de curta duração e defina um valor mais baixo. Outro que execute consultas de longa duração e defina um valor bem mais alto. Com essa abordagem, você pode depender de parâmetros de keepalive de TCP para interromper consultas de longa duração caso o servidor fique inoperante.
- `tcpKeepAlive`: ative esse parâmetro para garantir que os parâmetros de keepalive de TCP que você definir sejam respeitados.
- `loadBalanceHosts`: quando definido como `true`, esse parâmetro faz com que a aplicação se conecte a um host aleatório escolhido de uma lista de hosts candidatos.

Outras opções para obter a string do host

Você pode obter a string de host de várias origens, incluindo a função `aurora_replica_status` e usando a API do Amazon RDS.

Em muitos casos, você precisa determinar quem é o gravador do cluster ou localizar outros nós de leitor no cluster. Para fazer isso, sua aplicação pode se conectar a qualquer instância de banco de dados no cluster de banco de dados e consultar a função `aurora_replica_status`. Você pode

usar essa função para reduzir o tempo necessário para localizar um host ao qual se conectar. No entanto, em determinados cenários de falha de rede, a função `aurora_replica_status` pode exibir informações desatualizadas ou incompletas.

Uma boa maneira de garantir que a aplicação possa encontrar um nó ao qual se conectar é tentar se conectar ao endpoint do gravador do cluster e depois ao endpoint do leitor do cluster. Você deve fazer isso até estabelecer uma conexão legível. Esses endpoints não mudam, a menos que você renomeie o cluster de banco de dados. Por isso, você geralmente pode deixá-los como membros estáticos da aplicação ou armazená-los em um arquivo de recursos do qual a aplicação lê conteúdo.

Depois de estabelecer uma conexão usando um desses endpoints, você poderá obter informações sobre o restante do cluster. Para fazer isso, chame a função `aurora_replica_status`. Por exemplo, o comando a seguir recupera informações com `aurora_replica_status`.

```
postgres=> SELECT server_id, session_id, highest_lsn_rcvd, cur_replay_latency_in_usec,
 now(), last_update_timestamp
FROM aurora_replica_status();

server_id | session_id | highest_lsn_rcvd | cur_replay_latency_in_usec | now |
 last_update_timestamp
-----------+--------------------------------------+-----------------
+---------------------------+-------------------------------+-----------------------
mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca | 594221001 | 201421 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a | 594221001 | 201350 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-3 | MASTER_SESSION_ID | | | 2017-03-07 19:50:24.695322+00 | 2017-03-07
 19:50:23+00
(3 rows)
```

Por exemplo, a seção de hosts da string de conexão pode começar com os endpoints de cluster de gravador e leitor, conforme mostrado a seguir.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Nesse cenário, a aplicação tenta estabelecer uma conexão com qualquer tipo de nó, primário ou secundário. Quando a aplicação estiver conectada, é recomendável primeiro examinar o status de leitura/gravação do nó. Para fazer isso, consulte o resultado do comando `SHOW transaction_read_only`.

Se o valor de retorno da consulta for `OFF`, você já se conectou com êxito ao nó primário. No entanto, vamos supor que o valor de retorno seja `ON` e sua aplicação exija uma conexão de leitura/gravação. Nesse caso, você pode chamar a função `aurora_replica_status` para determinar o `server_id` que tem `session_id='MASTER_SESSION_ID'`. Essa função fornece o nome do nó primário. Você pode usá-la com o `endpointPostfix` descrito a seguir.

Tenha atenção quando você se conectar a uma réplica que tem dados obsoletos. Quando isso acontece, a função `aurora_replica_status` pode mostrar informações desatualizadas. Você pode definir um limite de desatualização no nível da aplicação. Para verificar isso, você pode observar a diferença entre a hora do servidor e o valor `last_update_timestamp`. Em geral, a aplicação deve evitar a mudança entre dois hosts devido a informações conflitantes retornadas pela função `aurora_replica_status`. Sua aplicação deve tentar todos os hosts conhecidos primeiro, em vez de seguir os dados retornados por `aurora_replica_status`.

Listar instâncias usando a operação de API DescribeDBCluster, exemplo em Java

Você pode localizar programaticamente a lista de instâncias usando AWS SDK para Java, especificamente a operação de API DescribeDBClusters.

Veja a seguir um pequeno exemplo de como você pode fazer isso no Java 8.

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

Aqui, `pgJDBCEndpointStr` contém uma lista formatada de endpoints, conforme mostrado a seguir.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

A variável `endpointPostfix` pode ser uma constante definida pela sua aplicação. Ou sua aplicação pode obtê-la consultando a operação de API `DescribeDBInstances` para uma única instância em seu cluster. Esse valor permanece constante em uma Região da AWS e para um cliente individual. Por isso, ele salva uma chamada de API para simplesmente manter isso constante em um arquivo de recursos do qual sua aplicação lê. No exemplo anterior, isso é definido como o seguinte.

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

Para fins de disponibilidade, é recomendável usar como padrão os endpoints do Aurora do seu cluster de banco de dados, se a API não estiver respondendo ou se estiver demorando muito para responder. Existe a garantia de que os endpoints permanecerão atualizados durante o tempo necessário para atualizar o registro DNS. A atualização do registro de DNS com um endpoint normalmente leva menos de 30 segundos. Você pode armazenar o endpoint em um arquivo de recursos consumido pela aplicação.

Testar o failover

Em todos os casos, você deve ter um cluster de banco de dados que contenha duas ou mais instâncias de banco de dados.

Do lado do servidor, certas operações de API podem causar uma interrupção que pode ser usada para testar como as aplicações respondem:

- FailoverDBCluster: essa operação tenta promover uma nova instância de banco de dados a gravador no cluster de banco de dados.

  O exemplo de código a seguir mostra como você pode usar o `failoverDBCluster` para causar uma interrupção. Para obter mais detalhes sobre a configuração de um cliente Amazon RDS, consulte Usar o AWS SDK for Java.

  ```
  public void causeFailover() {

      final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

      FailoverDBClusterRequest request = new FailoverDBClusterRequest();
      request.setDBClusterIdentifier("cluster-identifier");

      rdsClient.failoverDBCluster(request);
  }
  ```

- RebootDBInstance: o failover não é garantido com essa operação de API. No entanto, ele desativa o banco de dados no gravador. Você pode usá-lo para testar como sua aplicação responde a uma queda de conexões. O parâmetro `ForceFailover` não se aplica a mecanismos do Aurora. Em vez disso, use a operação de API `FailoverDBCluster`.
- ModifyDBCluster: modificar o parâmetro `Port` causará uma interrupção quando os nós no cluster começarem a escutar em uma porta nova. Em geral, sua aplicação pode responder a essa falha primeiro garantindo que somente ela controle as alterações de porta. Além disso, verifique se ela consegue atualizar adequadamente os endpoints dos quais depende. Para isso, você pode especificar que alguém atualize manualmente a porta ao fazer modificações no nível da API. Ou você pode fazer isso usando a API do RDS na aplicação para determinar se a porta foi alterada.
- ModifyDBInstance: modificar o parâmetro `DBInstanceClass` causa uma interrupção.
- DeleteDBInstance: excluir o primário (gravador) faz com que uma nova instância de banco de dados seja promovida a gravador no cluster de banco de dados.

Do lado da aplicação ou do cliente, se usar o Linux, você poderá testar como a aplicação responde a perdas súbitas de pacotes. Você pode fazer isso com base na porta, no host ou em se os pacotes de keepalive de TCP são enviados ou recebidos usando o comando iptables.

Exemplo de failover rápido em Java

O exemplo de código a seguir mostra como uma aplicação pode configurar um gerenciador de driver do Aurora PostgreSQL.

A aplicação chama `getConnection` quando precisa de uma conexão. Uma chamada para `getConnection` pode não conseguir encontrar um host válido. Um exemplo é quando nenhum gravador é encontrado, exceto o parâmetro `targetServerType` definido como `primary`. Nesse caso, a aplicação da chamada deve simplesmente tentar chamar a função novamente.

Para evitar enviar o comportamento de repetição à aplicação, você pode compactar essa nova tentativa em um agrupador de conexões. Com a maioria dos agrupadores de conexão, você pode especificar uma cadeia de conexão JDBC. Assim, sua aplicação pode chamar em `getJdbcConnectionString` e passá-lo para o agrupador de conexões. Isso significa que você pode usar um failover mais rápido com o Aurora PostgreSQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
   private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

   public FastFailoverDriverManager() {
       try {
            Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException e) {
            e.printStackTrace();
       }

       /*
         * RO endpoint has a TTL of 1s, we should honor that here. Setting this
 aggressively makes sure that when
         * the PG JDBC driver creates a new connection, it will resolve a new different
 RO endpoint on subsequent attempts
         * (assuming there is > 1 read node in your cluster)
         */
        java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
       // If the lookup fails, default to something like small to retry
       java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
   }

   public Connection getConnection(String targetServerType) throws SQLException {
       return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
   }

   public Connection getConnection(String targetServerType, Duration queryTimeout)
 throws SQLException {
        Connection conn =
 DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

       /*
         * A good practice is to set socket and statement timeout to be the same thing
 since both
```

```
        * the client AND server will stop the query at the same time, leaving no
running queries
        * on the backend
        */
       Statement st = conn.createStatement();
       st.execute("set statement_timeout to " + queryTimeout.getMillis());
       st.close();

      return conn;
  }

  private static String urlFormat = "jdbc:postgresql://%s"
          + "/postgres"
          + "?user=%s"
          + "&password=%s"
          + "&loginTimeout=%d"
          + "&connectTimeout=%d"
          + "&cancelSignalTimeout=%d"
          + "&socketTimeout=%d"
          + "&targetServerType=%s"
          + "&tcpKeepAlive=true"
          + "&ssl=true"
          + "&loadBalanceHosts=true";
  public String getJdbcConnectionString(String targetServerType, Duration
queryTimeout) {
      return String.format(urlFormat,
               getFormattedEndpointList(getLocalEndpointList()),
               CredentialManager.getUsername(),
               CredentialManager.getPassword(),
               LOGIN_TIMEOUT.getStandardSeconds(),
               CONNECT_TIMEOUT.getStandardSeconds(),
               CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
               queryTimeout.getStandardSeconds(),
               targetServerType
      );
  }

  private List<String> getLocalEndpointList() {
      /*
        * As mentioned in the best practices doc, a good idea is to read a local
resource file and parse the cluster endpoints.
        * For illustration purposes, the endpoint list is hardcoded here
        */
       List<String> newEndpointList = new ArrayList<>();
```

```
        newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
        newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

        return newEndpointList;
    }

    private static String getFormattedEndpointList(List<String> endpoints) {
        return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
    }
}
```

Recuperação rápida após failover com o gerenciamento de cache do cluster para o Aurora PostgreSQL

Para a recuperação rápida da instância de banco de dados do gravador em seus clusters do Aurora PostgreSQL, se houver um failover, use o gerenciamento de cache do cluster para o Amazon Aurora PostgreSQL. O gerenciamento de cache do cluster garante que a performance da aplicação seja mantida se houver um failover.

Em uma situação de failover típica, você pode ver uma degradação de performance temporária, mas grande, depois do failover. Essa degradação ocorre porque quando a instância de banco de dados de failover é iniciada, o cache do buffer está vazio. Um cache vazio também é conhecido como um cache frio. Um cache frio degrada a performance porque a instância de banco de dados precisa ler no disco, que é mais lento, em vez de aproveitar os valores armazenados no cache do buffer.

Com o gerenciamento de cache do cluster, você define uma instância de banco de dados do leitor específica como o destino do failover. O gerenciamento de cache do cluster garante que os dados no cache do leitor designado sejam mantidos sincronizados com os dados no cache da instância de banco de dados do gravador. O cache do leitor designado com valores pré-preenchidos é conhecido como um cache quente. Se ocorrer um failover, o leitor designado usará valores de seu cache quente imediatamente quando for promovido para a nova instância de banco de dados do gravador. Essa abordagem fornece uma performance de recuperação muito melhor à sua aplicação.

O gerenciamento de cache do cluster requer que a instância do leitor designada tenha o mesmo tipo de classe de instância e tamanho (`db.r5.2xlarge` ou `db.r5.xlarge`, por exemplo) que o gravador. Tenha isso em mente ao criar seus clusters de banco de dados do Aurora PostgreSQL

para que seu cluster possa se recuperar durante um failover. Para obter uma lista de tipos e tamanhos de classe de instância, consulte Especificações de hardware para classes de instância de banco de dados para o Aurora.

> **Note**
>
> Não há suporte ao gerenciamento de cache de cluster para um cluster de banco de dados secundário do Aurora PostgreSQL que faz parte de bancos de dados globais do Aurora. Com relação a clusters primários nos quais é possível usar esse recurso, é recomendável não executar nenhuma workload no leitor de nível 0 designado.

Sumário

- Configuração do gerenciamento de cache do cluster
  - Habilitar o gerenciamento de cache do cluster
  - Definir a prioridade da camada de promoção para a instância de banco de dados do gravador
  - Definir a prioridade da camada de promoção para a instância de banco de dados do leitor
- Monitorar o cache do buffer
- Solução de problemas de configuração do CCM

Configuração do gerenciamento de cache do cluster

Para configurar o gerenciamento de cache do cluster, realize os processos a seguir na ordem especificada.

Tópicos

- Habilitar o gerenciamento de cache do cluster
- Definir a prioridade da camada de promoção para a instância de banco de dados do gravador
- Definir a prioridade da camada de promoção para a instância de banco de dados do leitor

> **Note**
>
> Aguarde pelo menos 1 minuto depois de concluir estas etapas para que o gerenciamento de cache de cluster esteja totalmente operacional.

Habilitar o gerenciamento de cache do cluster

Para habilitar o gerenciamento de cache do cluster, siga as etapas descritas a seguir.

Console

Para habilitar o gerenciamento de cache do cluster

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros para seu cluster de banco de dados Aurora PostgreSQL.

   O cluster de banco de dados deve usar um grupo de parâmetros diferente do padrão, porque não é possível alterar os valores em um grupo de parâmetros padrão.
4. Em Parameter group actions (Ações do grupo de parâmetros), escolha Edit (Editar).
5. Defina o valor do parâmetro do cluster `apg_ccm_enabled` como 1.
6. Selecione Save changes.

AWS CLI

Para habilitar o gerenciamento de cache do cluster para um cluster de banco de dados do Aurora PostgreSQL, use o comando da AWS CLI `modify-db-cluster-parameter-group` com os seguintes parâmetros obrigatórios:

- `--db-cluster-parameter-group-name`
- `--parameters`

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my-db-cluster-parameter-group \
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my-db-cluster-parameter-group ^
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Definir a prioridade da camada de promoção para a instância de banco de dados do gravador

Para o gerenciamento de cache do cluster, verifique se a prioridade de promoção é tier-0 para a instância de banco de dados do gravador do cluster de banco de dados do Aurora PostgreSQL. A prioridade da camada de promoção é um valor que especifica a ordem em que um leitor do Aurora é promovido para a instância de banco de dados do leitor depois de uma falha. Os valores válidos são 0 – 15, em que 0 é a primeira prioridade, e 15 é a última prioridade. Para obter mais informações sobre a camada de promoção, consulte Tolerância a falhas para um cluster de banco de dados do Aurora.

Console

Para definir a prioridade da camada de promoção da instância de banco de dados do gravador como tier-0.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha a instância de banco de dados do Writer (Gravador) do cluster de banco de dados Aurora PostgreSQL.
4. Selecione Modify. A página Modify DB Instance (Modificar instância de banco de dados) é exibida.
5. No painel Additional configuration (Configuração adicional), escolha a tier-0 (camada-0) para a Failover priority (Prioridade de failover).
6. Selecione Continue (Continuar) e verifique o resumo de modificações.
7. Para aplicar as alterações imediatamente depois de salvá-las, escolha Apply immediately (Aplicar imediatamente).
8. Selecione Modify DB Instance (Modificar instância de banco de dados) para salvar suas alterações.

AWS CLI

Para definir a prioridade da camada de promoção como 0 para a instância de banco de dados do gravador usando a AWS CLI, chame o comando modify-db-instance com os seguintes parâmetros obrigatórios:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier writer-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier writer-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Definir a prioridade da camada de promoção para a instância de banco de dados do leitor

É necessário definir somente uma instância de banco de dados de leitor para o gerenciamento de cache do cluster. Para isso, escolha um leitor no cluster do Aurora PostgreSQL que seja da mesma classe e tamanho de instância que a instância de banco de dados do gravador. Por exemplo, se o gravador usar `db.r5.xlarge`, escolha um leitor que use esse mesmo tipo e tamanho de classe de instância. Depois, defina a prioridade de sua camada de promoção como 0.

A prioridade da camada de promoção é um valor que especifica a ordem em que um leitor do Aurora é promovido para a instância de banco de dados do leitor depois de uma falha. Os valores válidos são 0 – 15, em que 0 é a primeira prioridade, e 15 é a última prioridade.

Console

Para definir a prioridade de promoção da instância de banco de dados do leitor como tier-0

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha uma instância de banco de dados do Reader (Leitor) do cluster de banco de dados do Aurora PostgreSQL que seja da mesma classe de instância que a instância de banco de dados do gravador.
4. Selecione Modify. A página Modify DB Instance (Modificar instância de banco de dados) é exibida.
5. No painel Additional configuration (Configuração adicional), escolha a tier-0 (camada-0) para a Failover priority (Prioridade de failover).
6. Selecione Continue (Continuar) e verifique o resumo de modificações.
7. Para aplicar as alterações imediatamente depois de salvá-las, escolha Apply immediately (Aplicar imediatamente).
8. Selecione Modify DB Instance (Modificar instância de banco de dados) para salvar suas alterações.

AWS CLI

Para definir a prioridade da camada de promoção como 0 para a instância de banco de dados do gravador usando a AWS CLI, chame o comando modify-db-instance com os seguintes parâmetros obrigatórios:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier reader-db-instance \
```

```
    --promotion-tier 0 \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier reader-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Monitorar o cache do buffer

Depois de configurar o gerenciamento de cache de cluster, você pode monitorar o estado de sincronização entre o cache de buffer da instância de banco de dados do gravador e o cache de buffer a quente do leitor designado. Para examinar o conteúdo do cache do buffer nas instâncias de banco de dados do gravador e do leitor, use o módulo `pg_buffercache` do PostgreSQL. Para obter mais informações, consulte a seção Documentação `pg_buffercache` do PostgreSQL.

Como usar a função **`aurora_ccm_status`**

O gerenciamento de cache do cluster também fornece a função `aurora_ccm_status`. Use a função `aurora_ccm_status` na instância de banco de dados do gravador para obter as informações a seguir sobre o andamento do aquecimento do cache no leitor designado:

- `buffers_sent_last_minute` – quantos buffers foram enviados ao leitor designado no último minuto.
- `buffers_found_last_minute`: o número de buffers acessados com frequência que foram identificados durante o último minuto.
- `buffers_sent_last_scan` – quantos buffers foram enviados ao leitor designado durante a última verificação completa do cache do buffer.
- `buffers_found_last_scan` – quantos buffers foram identificados como acessados com frequência e precisaram ser enviados durante a última verificação completa do cache do buffer. Os buffers que já estão em cache no leitor designado não são enviados.
- `buffers_sent_current_scan` – quantos buffers foram enviados até agora durante a verificação atual.
- `buffers_found_current_scan` – quantos buffers forma identificados como acessados com frequência na verificação atual.

- `current_scan_progress` – quantos buffers foram visitados até agora durante a verificação atual.

O exemplo a seguir mostra como usar a função `aurora_ccm_status` para converter parte de sua saída em uma taxa quente e em uma porcentagem quente.

```
SELECT buffers_sent_last_minute*8/60 AS warm_rate_kbps,
   100*(1.0-buffers_sent_last_scan::float/buffers_found_last_scan) AS warm_percent
   FROM aurora_ccm_status();
```

Solução de problemas de configuração do CCM

Ao habilitar o parâmetro de cluster `apg_ccm_enabled`, o gerenciamento do cache do cluster é ativado automaticamente em nível de instância na instância de banco de dados de gravador e em uma instância de banco de dados de leitor no cluster de banco de dados do Aurora PostgreSQL. As instâncias de gravador e de leitor devem usar o mesmo tipo e tamanho de classe de instância. A prioridade do nível de promoção deles é definida como `0`. Outras instâncias de leitor no cluster de banco de dados devem ter um nível de promoção diferente de zero e o gerenciamento do cache do cluster está desabilitado para essas instâncias.

Os motivos a seguir podem causar problemas na configuração e desabilitar o gerenciamento do cache do cluster:

- Quando não há uma única instância de banco de dados de leitor definida como nível de promoção 0.
- Quando a instância de banco de dados de leitor não está definida como nível de promoção 0.
- Quando mais de uma instância de banco de dados de leitor está definida como nível de promoção 0.
- Quando as instâncias de gravador e uma instância de banco de dados de leitor com nível de promoção 0 não têm o mesmo tamanho de instância.

Gerenciar a rotatividade de conexão do Aurora PostgreSQL com agrupamento

Quando as aplicações cliente se conectam e desconectam com uma frequência tal que o tempo de resposta do cluster de banco de dados do Aurora PostgreSQL diminui, diz-se que o cluster

está passando por rotatividade de conexão. Cada nova conexão com o endpoint do cluster de banco de dados do Aurora PostgreSQL consome recursos, reduzindo assim os recursos que podem ser usados para processar a workload em si. A rotatividade de conexão é um problema que recomendamos gerenciar com algumas das práticas recomendadas descritas a seguir.

Para começar, você pode melhorar os tempos de resposta nos clusters de banco de dados do Aurora PostgreSQL que têm altas taxas de rotatividade de conexão. Por exemplo, você pode usar um agrupador de conexões, como o RDS Proxy. O agrupador de conexões fornece aos clientes um cache de conexões prontas para uso. Quase todas as versões do Aurora PostgreSQL comportam o RDS Proxy. Para obter mais informações, consulte Proxy do Amazon RDS com Aurora PostgreSQL.

Se a sua versão específica do Aurora PostgreSQL não for compatível com o RDS Proxy, você poderá usar outro agrupador de conexões compatível com o PostgreSQL, como o PgBouncer. Para saber mais, consulte o site do PgBouncer.

Para ver se o cluster de banco de dados do Aurora PostgreSQL pode se beneficiar do agrupamento de conexões, confira o arquivo `postgresql.log` para consultar sobre conexões e desconexões. Você também pode usar o Performance Insights para descobrir o nível de rotatividade de conexão que o cluster de banco de dados do Aurora PostgreSQL está enfrentando. A seguir, você encontrará informações sobre os dois tópicos.

Registrar em log conexões e desconexões

O `log_connections` PostgreSQL e os parâmetros do `log_disconnections` podem capturar conexões e desconexões com a instância do gravador do cluster de banco de dados do Aurora PostgreSQL. Por padrão, esse parâmetro é desativado. Para ativar esses parâmetros, use um grupo de parâmetros personalizado e altere o valor para 1. Para obter mais informações sobre grupos de parâmetros personalizados, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora. Para verificar as configurações, conecte-se ao endpoint do cluster de banco de dados para o Aurora PostgreSQL usando psql e consulta da maneira a seguir.

```
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_connections';
 setting
---------
on
(1 row)
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_disconnections';
```

```
setting
---------
on
(1 row)
```

Com esses dois parâmetros ativados, o log captura todas as novas conexões e desconexões. Você vê o usuário e o banco de dados de cada nova conexão autorizada. No momento da desconexão, a duração da sessão também é registrada em log, como mostrado no exemplo a seguir.

```
2022-03-07 21:44:53.978 UTC [16641] LOG: connection authorized: user=labtek
 database=labdb application_name=psql
2022-03-07 21:44:55.718 UTC [16641] LOG: disconnection: session time: 0:00:01.740
 user=labtek database=labdb host=[local]
```

Para verificar se há rotatividade de conexão em sua aplicação, ative esses parâmetros se eles ainda não estiverem ativados. Depois disso, reúna dados no log do PostgreSQL para análise executando sua aplicação com uma workload e um período realistas. Você pode visualizar o arquivo de log no console do RDS. Escolha a instância do gravador do seu cluster de banco de dados do Aurora PostgreSQL e selecione a guia Logs & events (Logs e eventos). Para obter mais informações, consulte Como visualizar e listar arquivos de log do banco de dados.

Ou você pode baixar o arquivo de log por meio do console e usar a sequência de comandos a seguir. Essa sequência encontra o número total de conexões autorizadas e descartadas por minuto.

```
grep "connection authorized\|disconnection: session time:"
 postgresql.log.2022-03-21-16|\
awk {'print $1,$2}' |\
sort |\
uniq -c |\
sort -n -k1
```

No exemplo de saída, você pode ver um pico nas conexões autorizadas seguido de desconexões a partir de 16:12:10.

```
.....
,......
.........
5 2022-03-21 16:11:55 connection authorized:
9 2022-03-21 16:11:55 disconnection: session
5 2022-03-21 16:11:56 connection authorized:
5 2022-03-21 16:11:57 connection authorized:
```

```
5 2022-03-21 16:11:57 disconnection: session
32 2022-03-21 16:12:10 connection authorized:
30 2022-03-21 16:12:10 disconnection: session
31 2022-03-21 16:12:11 connection authorized:
27 2022-03-21 16:12:11 disconnection: session
27 2022-03-21 16:12:12 connection authorized:
27 2022-03-21 16:12:12 disconnection: session
41 2022-03-21 16:12:13 connection authorized:
47 2022-03-21 16:12:13 disconnection: session
46 2022-03-21 16:12:14 connection authorized:
41 2022-03-21 16:12:14 disconnection: session
24 2022-03-21 16:12:15 connection authorized:
29 2022-03-21 16:12:15 disconnection: session
28 2022-03-21 16:12:16 connection authorized:
24 2022-03-21 16:12:16 disconnection: session
40 2022-03-21 16:12:17 connection authorized:
42 2022-03-21 16:12:17 disconnection: session
40 2022-03-21 16:12:18 connection authorized:
40 2022-03-21 16:12:18 disconnection: session
.....
,.......
.........
1 2022-03-21 16:14:10 connection authorized:
1 2022-03-21 16:14:10 disconnection: session
1 2022-03-21 16:15:00 connection authorized:
1 2022-03-21 16:16:00 connection authorized:
```

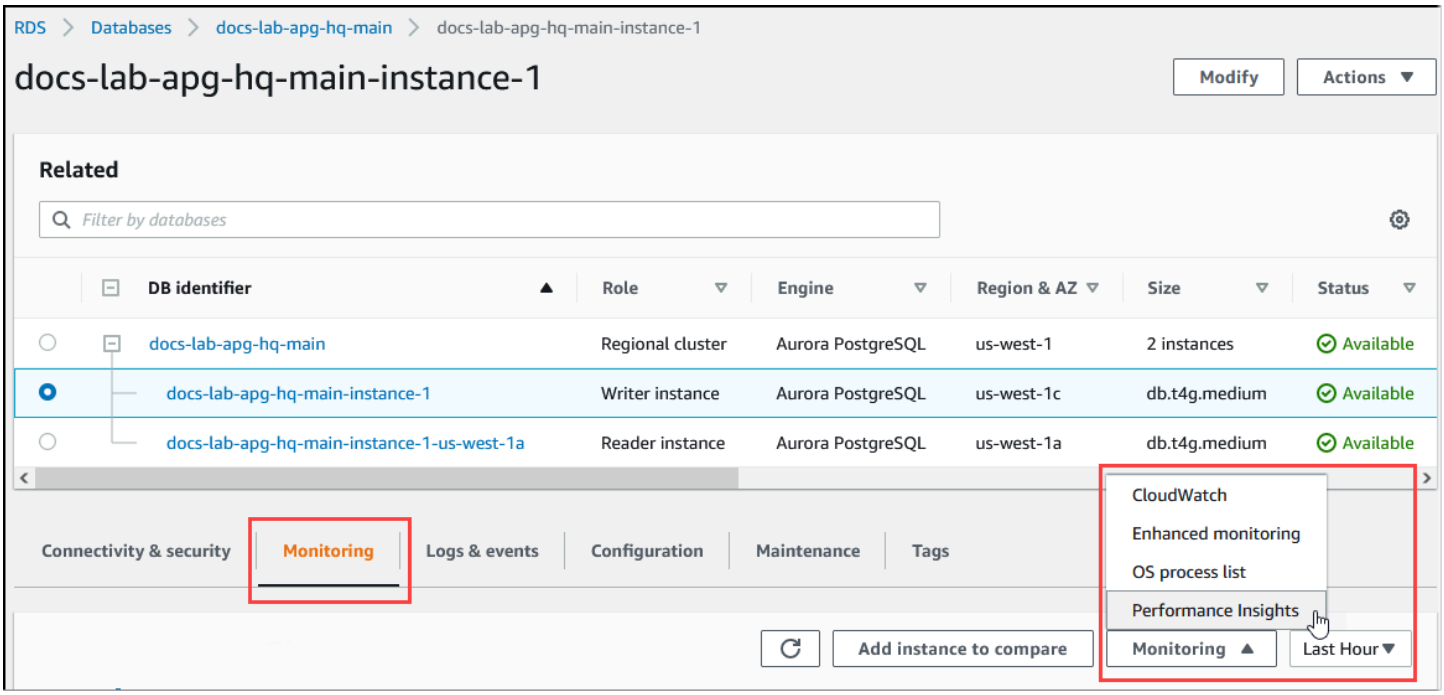
Com essas informações, você pode decidir se a workload pode se beneficiar de um agrupador de conexões. Para obter uma análise mais detalhada, você pode usar o Performance Insights.

Detectar a rotatividade de conexão com o Performance Insights

Você pode usar o Performance Insights para avaliar a quantidade de rotatividade de conexão no cluster de banco de dados da edição compatível com Aurora PostgreSQL. Ao criar um cluster de banco de dados do Aurora PostgreSQL, a configuração do Performance Insights é ativada por padrão. Se você desmarcou essa opção ao criar o cluster de banco de dados, modifique o cluster para ativar o recurso. Para obter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.

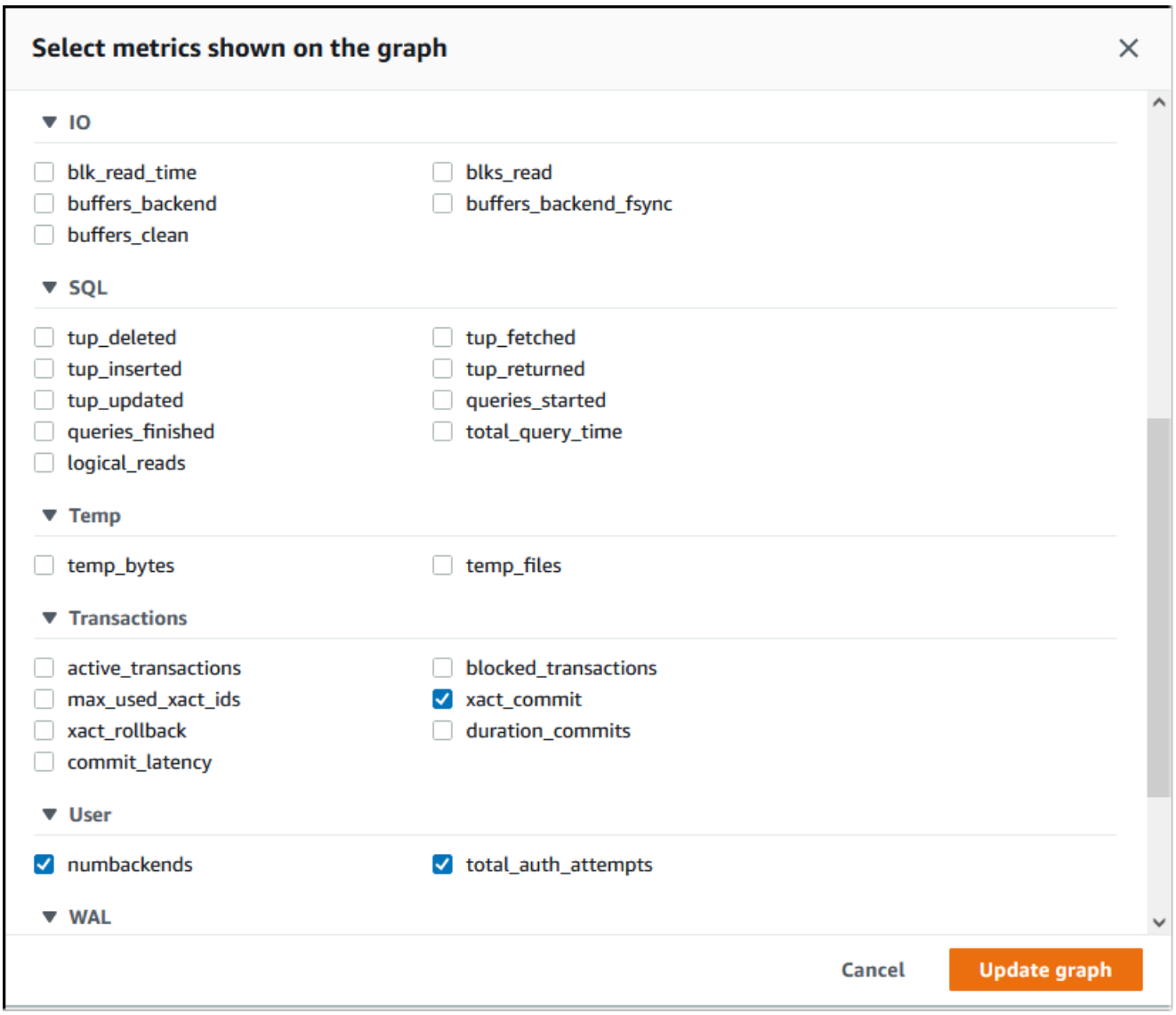
Com o Performance Insights em execução no cluster de banco de dados do Aurora PostgreSQL, você pode escolher as métricas que deseja monitorar. Você pode acessar o Performance Insights no painel de navegação do console. Você também pode acessar o Performance Insights com a guia

Monitoring (Monitoramento) da instância do gravador para o cluster de banco de dados do Aurora PostgreSQL, conforme mostrado na imagem a seguir.



No console do Performance Insights, escolha Manage metrics (Gerenciar métricas). Para analisar a atividade de conexão e desconexão do cluster de banco de dados do Aurora PostgreSQL, escolha as métricas a seguir. Estas métricas são todas do PostgreSQL.

- `xact_commit`: o número de transações confirmadas.
- `total_auth_attempts`: o número de tentativas de conexões de usuários autenticados por minuto.
- `numbackends`: o número de back-ends conectados ao banco de dados no momento.

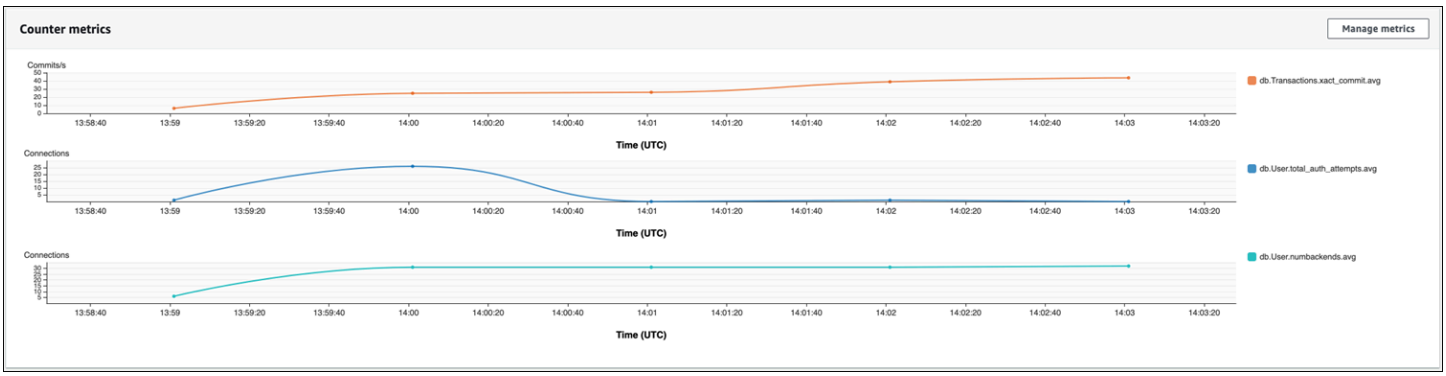


Para salvar as configurações e exibir a atividade de conexão, escolha Update graph (Atualizar grafo).

Na imagem a seguir, você pode ver o impacto da execução do pgbench com 100 usuários. A linha que mostra as conexões é consistentemente ascendente. Para saber mais sobre o pgbench e como usá-lo, consulte pgbench na documentação do PostgreSQL.

A imagem mostra que executar uma workload com até 100 usuários sem um agrupador de conexões pode aumentar de maneira significativa o número de `total_auth_attempts` ao longo do processamento da workload. Observe que é melhor manter `total_auth_attempts` o mais próximo possível de zero.

Com o agrupamento de conexões do RDS Proxy, as tentativas de conexão aumentam no início da workload. Após a configuração do agrupamento de conexões, a média diminui. Os recursos usados pelas transações e pelo uso do back-end permanecem consistentes ao longo do processamento da workload.



Para obter mais informações sobre como usar o Performance Insights com o cluster de banco de dados do Aurora PostgreSQL, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora. Para analisar as métricas, consulte Análise de métricas usando o painel do Performance Insights.

Demonstrar os benefícios do agrupamento de conexões

Conforme mencionado anteriormente, se você determinar que o cluster de banco de dados do Aurora PostgreSQL tem um problema de rotatividade de conexão, é possível usar o RDS Proxy para melhorar a performance. A seguir, você encontrará um exemplo que mostra as diferenças no

processamento de uma workload quando as conexões estão agrupadas e quando não estão. O exemplo usa o pgbench para modelar uma workload de transação.

Assim como o psql, o pgbench é uma aplicação cliente do PostgreSQL que você pode instalar e executar em sua máquina cliente local. Também é possível instalá-la e executá-la por meio da instância do Amazon EC2 que você usa para gerenciar o cluster de banco de dados do Aurora PostgreSQL. Para obter mais informações, consulte pgbench na documentação do PostgreSQL.

Para analisar esse exemplo, crie primeiro o ambiente do pgbench em seu banco de dados. O comando a seguir é o modelo básico para inicializar as tabelas do pgbench no banco de dados especificado. Este exemplo usa a conta de usuário principal padrão, `postgres`, para o login. Se necessário, altere-a para seu cluster de banco de dados do Aurora PostgreSQL. Crie o ambiente do pgbench em um banco de dados na instância do gravador do cluster.

> **Note**
>
> O processo de inicialização do pgbench descarta e recria as tabelas chamadas `pgbench_accounts`, `pgbench_branches`, `pgbench_history` e `pgbench_tellers`. O banco de dados que você escolheu para *`dbname`* ao inicializar o pgbench não deve usar esses nomes.

```
pgbench -U postgres -h db-cluster-instance-1.111122223333.aws-region.rds.amazonaws.com
 -p 5432 -d -i -s 50 dbname
```

Para o pgbench, especifique os seguintes parâmetros:

-d

Gera um relatório de depuração à medida que o pgbench é executado.

-h

Especifica o endpoint da instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

-i

Inicializa o ambiente do pgbench no banco de dados para os testes de comparação.

-p

Identifica a porta usada para conexões de banco de dados. O padrão para o Aurora PostgreSQL geralmente é 5432 ou 5433.

-s

Especifica o fator de escalabilidade a ser usado para preencher as tabelas com linhas. O fator de escalabilidade padrão é 1, o que gera 1 linha na tabela `pgbench_branches`, 10 linhas na tabela `pgbench_tellers` e 100 mil linhas na tabela `pgbench_accounts`.

-U

Especifica a conta de usuário para a instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

Depois que o ambiente do pgbench estiver configurado, você poderá executar testes de comparação com e sem agrupamento de conexões. O teste padrão consiste em uma série de cinco comandos SELECT, UPDATE e INSERT por transação que são executados repetidamente pelo tempo especificado. Você pode especificar o fator de escalabilidade, o número de clientes e outros detalhes para modelar seus próprios casos de uso.

Como exemplo, o comando a seguir executa a comparação por 60 segundos (a opção -T, para tempo) com 20 conexões simultâneas (a opção -C). A opção -C faz com que o teste seja executado usando uma nova conexão a cada vez, e não uma vez por sessão do cliente. Essa configuração fornece uma indicação da sobrecarga da conexão.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 -C labdb
Password:**********
pgbench (14.3, server 13.3)
  starting vacuum...end.
  transaction type: <builtin: TPC-B (sort of)>
    scaling factor: 50
    query mode: simple
    number of clients: 20
    number of threads: 1
    duration: 60 s
    number of transactions actually processed: 495
    latency average = 2430.798 ms
    average connection time = 120.330 ms
    tps = 8.227750 (including reconnection times)
```

A execução do pgbench na instância do gravador de um cluster de banco de dados do Aurora PostgreSQL sem reutilizar conexões mostra que cerca de oito transações apenas são processadas a cada segundo. Isso totaliza 495 transações durante o teste de 1 minuto.

Se você reutilizar conexões, a resposta do cluster de banco de dados do Aurora PostgreSQL para o número de usuários será quase 20 vezes mais rápida. Com a reutilização, em vez de 495, são processadas ao todo 9.042 transações no mesmo período e para o mesmo número de conexões de usuários. A diferença é que, no exemplo a seguir, cada conexão está sendo reutilizada.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 labdb
Password:*********
pgbench (14.3, server 13.3)
      starting vacuum...end.
      transaction type: <builtin: TPC-B (sort of)>
      scaling factor: 50
      query mode: simple
      number of clients: 20
      number of threads: 1
      duration: 60 s
      number of transactions actually processed: 9042
      latency average = 127.880 ms
      initial connection time = 2311.188 ms
      tps = 156.396765 (without initial connection time)
```

Este exemplo mostra que o agrupamento de conexões pode melhorar significativamente os tempos de resposta. Para obter mais informações sobre como configurar o RDS Proxy para o cluster de banco de dados do Aurora PostgreSQL, consulte Amazon RDS Proxypara Aurora.

Tratamento de conexões inativas no PostgreSQL

Conexões inativas ocorrem quando uma sessão de banco de dados permanece ativa no servidor mesmo que a aplicação cliente tenha sido abandonada ou encerrada de forma anormal. Essa situação geralmente ocorre quando os processos do cliente falham ou são encerrados inesperadamente sem fechar adequadamente as respectivas conexões de banco de dados ou cancelar solicitações em andamento.

O PostgreSQL identifica e limpa com eficiência as conexões inativas quando os processos do servidor estão ociosos ou tentam enviar dados aos clientes. No entanto, a detecção é um desafio para sessões que estão ociosas, aguardando a entrada do cliente ou executando

consultas ativamente. Para lidar com esses cenários, o PostgreSQL oferece os parâmetros `tcp_keepalives_*`, `tcp_user_timeout` e `client_connection_check_interval`.

Tópicos

- Conceitos básicos sobre keepalive de TCP
- Principais parâmetros de keepalive de TCP no Aurora PostgreSQL
- Casos de uso para configurações de keepalive de TCP
- Práticas recomendadas

Conceitos básicos sobre keepalive de TCP

O keepalive de TCP é um mecanismo em nível de protocolo que ajuda a manter e verificar a integridade da conexão. Cada conexão TCP mantém configurações em nível de kernel que controlam o comportamento de keepalive. Quando o temporizador de keepalive expira, o sistema realiza o seguinte:

- Envia um pacote de sondagem sem dados e com o sinalizador ACK definido.
- Espera uma resposta do endpoint remoto de acordo com as especificações de TCP/IP.
- Gerencia o estado da conexão com base na resposta ou na falta dela.

Principais parâmetros de keepalive de TCP no Aurora PostgreSQL

Parameter	Descrição	Valores padrão
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300
`tcp_keepalives_interval`	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30
`tcp_keepalives_count`	Maximum lost keepalive messages before declaring connection dead	2

Parameter	Descrição	Valores padrão
`tcp_user_timeout`	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0
`client_connection_check_interval`	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

Casos de uso para configurações de keepalive de TCP

Manter as sessões ociosas ativas

Para evitar que conexões ociosas sejam encerradas por firewalls ou roteadores devido a inatividade:

- Configure `tcp_keepalives_idle` para enviar pacotes de keepalive em intervalos regulares.

Detectar conexões inativas

Para detectar conexões inativas imediatamente:

- Ajuste `tcp_keepalives_idle`, `tcp_keepalives_interval` e `tcp_keepalives_count`. Por exemplo, com os padrões do Aurora PostgreSQL, a detecção de uma conexão inativa leva cerca de 1 minuto (2 sondagens × 30 segundos). Reduzir esses valores pode acelerar a detecção.
- Use `tcp_user_timeout` para especificar o tempo máximo de espera para uma confirmação.

As configurações de keepalive de TCP ajudam o kernel a detectar conexões inativas, mas talvez o PostgreSQL não aja enquanto o soquete não for usado. Se uma sessão estiver executando uma consulta longa, as conexões inativas só poderão ser detectadas após a conclusão da consulta. No PostgreSQL 14 e versões posteriores, `client_connection_check_interval` pode acelerar a detecção de conexões inativas pesquisando periodicamente o soquete durante a execução da consulta.

Práticas recomendadas

- Defina intervalos razoáveis de keepalive: ajuste `tcp_user_timeout`, `tcp_keepalives_idle`, `tcp_keepalives_count` e `tcp_keepalives_interval` para contrabalançar a velocidade de detecção e o uso de recursos.
- Otimize para melhorar seu ambiente: alinhe as configurações ao comportamento da rede, às políticas de firewall e às necessidades da sessão.
- Aproveite os recursos do PostgreSQL: use `client_connection_check_interval` no PostgreSQL 14 e versões posteriores para realizar verificações de conexão eficientes.

Ajustar parâmetros de memória para o Aurora PostgreSQL

No Amazon Aurora PostgreSQL, você pode usar vários parâmetros que controlam a quantidade de memória usada para várias tarefas de processamento. Se uma tarefa consumir mais memória do que a quantidade definida para um determinado parâmetro, o Aurora PostgreSQL usa outros recursos para processamento, como gravação em disco. Isso pode fazer com que o cluster de banco de dados do Aurora PostgreSQL fique lento ou possivelmente seja interrompido, com um erro de falta de memória.

A configuração padrão para cada parâmetro de memória geralmente pode lidar com as tarefas de processamento pretendidas. No entanto, você também pode ajustar os parâmetros relacionados à memoria no cluster de banco de dados do Aurora PostgreSQL. Você pode fazer esse ajuste para garantir que seja alocada memória suficiente seja para processar sua workload específica.

A seguir, você encontrará informações sobre parâmetros que controlam o gerenciamento da memória. Você também pode aprender a avaliar a utilização da memória.

Verificar e definir valores de parâmetros

Os parâmetros que você pode definir para gerenciar a memória e avaliar o uso de memória do cluster de banco de dados do Aurora PostgreSQL incluem o seguinte:

- `work_mem`: especifica a quantidade de memória que o cluster de banco de dados do Aurora PostgreSQL usa para operações de classificação internas e tabelas de hash antes da gravação em arquivos temporários de disco.
- `log_temp_files`: registra em log a criação, os nomes e os tamanhos dos arquivos temporários. Quando esse parâmetro é ativado, uma entrada de registro em log é armazenada para cada arquivo temporário criado. Ative-o para ver com que frequência o cluster de banco de dados do

Aurora PostgreSQL precisa gravar em disco. Desative-o novamente depois de coletar informações sobre a geração de arquivos temporários do cluster de banco de dados do Aurora PostgreSQL, para evitar registros em log excessivos.

- `logical_decoding_work_mem`: especifica a quantidade de memória (em quilobytes) a ser usada por cada buffer de reordenamento interno antes da transferência para o disco. Essa memória é usada para decodificação lógica, que é o processo usado para criar uma réplica. Isso é feito por meio da conversão de dados do arquivo de log de gravação antecipada (WAL) na saída lógica do streaming necessária para o destino.

  O valor desse parâmetro cria um único buffer do tamanho especificado para cada conexão de replicação. Por padrão, é 65.536 KB. Depois que esse buffer é preenchido, o excesso é gravado em disco como arquivo. Para minimizar a atividade do disco, você pode ajustar o valor desse parâmetro como um valor bem maior que `work_mem`.

Como todos eles são parâmetros dinâmicos, você pode alterá-los para a sessão atual. Para fazer isso, conecte-se ao cluster de banco de dados do Aurora PostgreSQL com psql e usando a instrução `SET`, conforme mostrado a seguir.

```
SET parameter_name TO parameter_value;
```

As configurações da sessão duram apenas na sessão. Quando a sessão termina, o parâmetro é revertido para a respectiva configuração no grupo de parâmetros do cluster de banco de dados. Antes de alterar qualquer parâmetro, primeiro verifique os valores atuais consultando a tabela `pg_settings`, da maneira a seguir.

```
SELECT unit, setting, max_val
   FROM pg_settings WHERE name='parameter_name';
```

Por exemplo, para encontrar o valor do parâmetro `work_mem`, conecte-se à instância do gravador do cluster de banco de dados do Aurora PostgreSQL e execute a consulta a seguir.

```
SELECT unit, setting, max_val, pg_size_pretty(max_val::numeric)
  FROM pg_settings WHERE name='work_mem';
unit  | setting  | max_val   | pg_size_pretty
------+----------+-----------+----------------
 kB   | 1024     | 2147483647| 2048 MB
(1 row)
```

Alterar as configurações dos parâmetros para que persistam exige o uso de um grupo de parâmetros do cluster de banco de dados. Depois de executar seu cluster de banco de dados do Aurora PostgreSQL com valores diferentes para esses parâmetros usando a instrução `SET`, você pode criar um grupo personalizado de parâmetros e aplicar ao seu cluster de banco de dados do Aurora PostgreSQL. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.

Entender o parâmetro da memória de trabalho

O parâmetro da memória de trabalho (`work_mem`) especifica a quantidade máxima de memória que o Aurora PostgreSQL pode usar para processar consultas complexas. As consultas complexas incluem as que envolvem operações de classificação ou agrupamento, ou seja, consultas que usam as seguintes cláusulas:

- ORDER BY
- DISTINCT
- GROUP BY
- JOIN (MERGE e HASH)

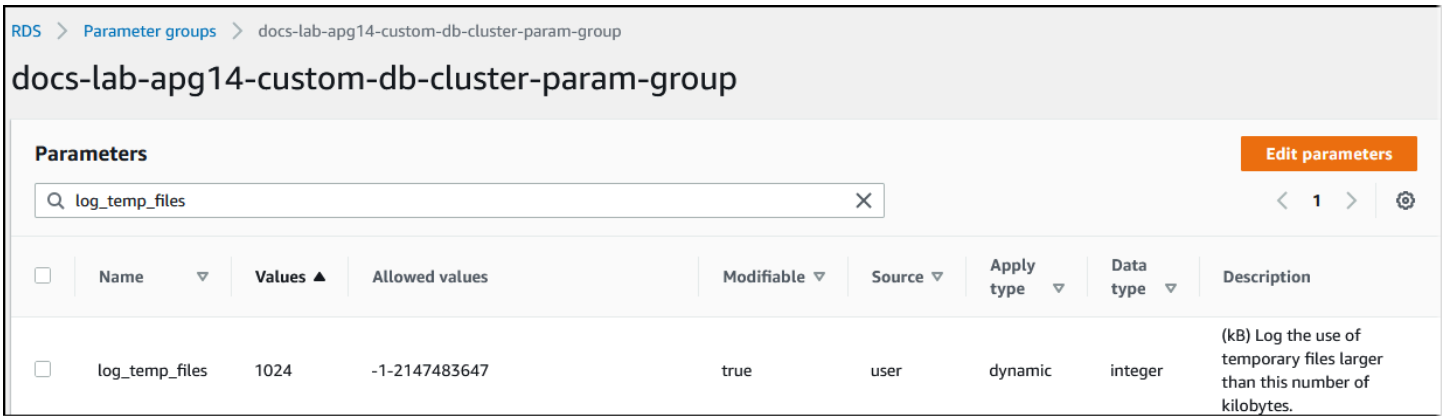
O planejador de consultas afeta indiretamente a forma como seu cluster de banco de dados do Aurora PostgreSQL usa a memória de trabalho. O planejador de consultas gera planos de execução para processar instruções SQL. Um plano específico pode dividir uma consulta complexa em várias unidades de trabalho que podem ser executadas em paralelo. Quando possível, o Aurora PostgreSQL usa a quantidade de memória especificada no parâmetro `work_mem` referente a cada sessão antes de gravar em disco para cada processo paralelo.

Vários usuários do banco de dados que executam várias operações simultaneamente e geram várias unidades de trabalho em paralelo podem esgotar a memória de trabalho alocada do seu cluster de banco de dados do Aurora PostgreSQL. Isso pode levar à criação excessiva de arquivos temporários e à E/S de disco ou, pior ainda, a um erro de falta de memória.

Identificar o uso temporário de arquivos

Sempre que a memória necessária para processar consultas exceder o valor especificado no parâmetro `work_mem`, os dados de trabalho serão transferidos para o disco em um arquivo temporário. Você pode ver com que frequência isso ocorre ativando o parâmetro `log_temp_files`. Por padrão, o parâmetro está desativado (definido como -1). Para capturar todas as informações do arquivo temporário, defina esse parâmetro como 0. Defina `log_temp_files` como qualquer outro número inteiro positivo a fim de capturar informações de arquivos temporários para arquivos iguais

ou maiores que essa quantidade de dados (em quilobytes). Na imagem a seguir, você pode ver um exemplo de Console de gerenciamento da AWS.



Depois de configurar o registro em log de arquivos temporários, você pode testar com sua própria workload para ver se a configuração da memória de trabalho é suficiente. Você também pode simular uma workload usando o pgbench, uma aplicação simples de comparação da comunidade do PostgreSQL.

O exemplo a seguir inicializa (`-i`) `pgbench` criando as tabelas e linhas necessárias para executar os testes. Neste exemplo, o fator de escalabilidade (`-s` 50) cria 50 linhas na tabela `pgbench_branches`, 500 linhas em `pgbench_tellers` e 5 milhões de linhas na tabela `pgbench_accounts` no banco de dados `labdb`.

```
pgbench -U postgres -h your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -i -s 50 labdb
Password:
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 15.46 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 61.13 s (drop tables 0.08 s, create tables 0.39 s, client-side generate 54.85
 s, vacuum 2.30 s, primary keys 3.51 s)
```

Depois de inicializar o ambiente, você pode executar a comparação para um período específico (`-T`) e o número de clientes (`-c`). Esse exemplo também usa a opção `-d` para gerar informações de

depuração à medida que as transações são processadas pelo cluster de banco de dados do Aurora PostgreSQL.

```
pgbench -h -U postgres your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -d -T 60 -c 10 labdb
Password:*******
pgbench (14.3)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 1408
latency average = 398.467 ms
initial connection time = 4280.846 ms
tps = 25.096201 (without initial connection time)
```

Para obter mais informações sobre o pg_upgrade, consulte pgbench na documentação do PostgreSQL.

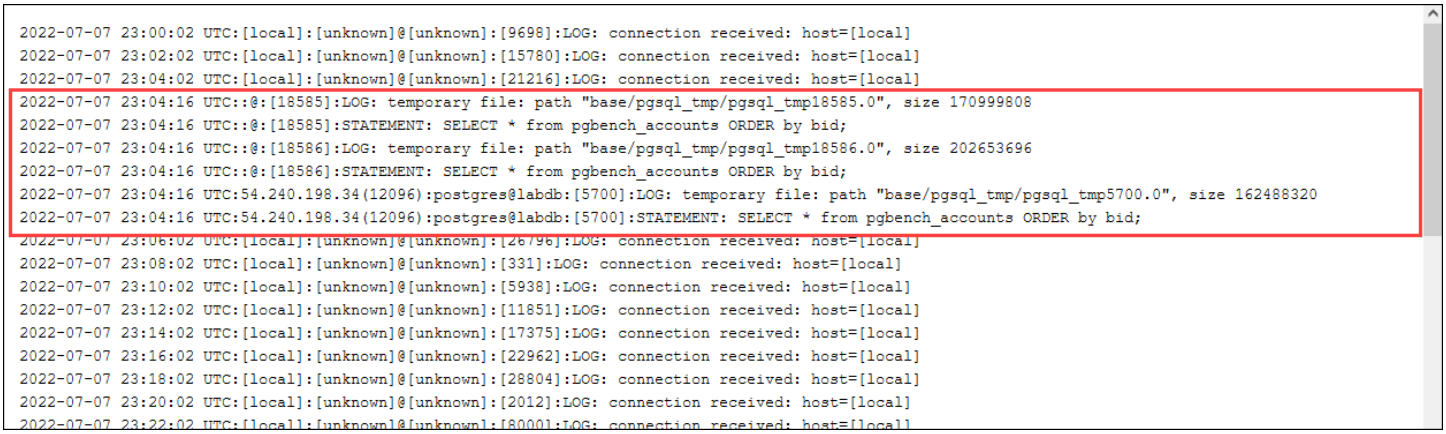
Você pode usar o comando de metacomando do psql (\d) para listar as relações, como tabelas, visualizações e índices criados pelo pgbench.

```
labdb=>  \d pgbench_accounts
 Table "public.pgbench_accounts"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 aid      | integer       |           | not null |
 bid      | integer       |           |          |
 abalance | integer       |           |          |
 filler   | character(84) |           |          |
Indexes:
    "pgbench_accounts_pkey" PRIMARY KEY, btree (aid)
```

Conforme mostrado na saída, a tabela `pgbench_accounts` está indexada na coluna `aid`. Para garantir que a próxima consulta use a memória de trabalho, consulte qualquer coluna não indexada, como a mostrada no exemplo a seguir.

```
postgres=> SELECT * FROM pgbench_accounts ORDER BY bid;
```

Verifique se há arquivos temporários no log. Para fazer isso, abra o Console de gerenciamento da AWS, escolha a instância do cluster de banco de dados do Aurora PostgreSQL e selecione a guia Logs & Events (Registros em log e eventos). Visualize os logs no console ou baixe-os para análise posterior. Conforme mostrado na imagem a seguir, o tamanho dos arquivos temporários necessários para processar a consulta indica que você deve considerar aumentar a quantidade especificada para o parâmetro `work_mem`.



Você pode configurar esse parâmetro de forma diferente para indivíduos e grupos, com base em suas necessidades operacionais. Por exemplo, você pode ajustar o parâmetro `work_mem` como 8 GB para a função chamada `dev_team`.

```
postgres=> ALTER ROLE dev_team SET work_mem='8GB';
```

Com essa definição para `work_mem`, são alocados até 8 GB de memória de trabalho a qualquer função membro da função `dev_team`.

Usar índices para um tempo de resposta mais rápido

Se suas consultas estiverem demorando muito para retornar resultados, você pode verificar se seus índices estão sendo usados conforme o esperado. Primeiro, ative `\timing`, o metacomando psql, da seguinte maneira.

```
postgres=>  \timing on
```

Depois de ativar o tempo, use uma instrução SELECT simples.

```
postgres=> SELECT COUNT(*) FROM
  (SELECT * FROM pgbench_accounts
  ORDER BY bid)
```

```
  AS accounts;
count
-------
5000000
(1 row)
Time: 3119.049 ms (00:03.119)
```

Conforme mostrado na saída, essa consulta levou pouco mais de 3 segundos para ser concluída. Para melhorar o tempo de resposta, crie um índice em `pgbench_accounts`, da maneira a seguir.

```
postgres=> CREATE INDEX ON pgbench_accounts(bid);
CREATE INDEX
```

Execute a consulta novamente e observe o tempo de resposta mais rápido. Neste exemplo, a consulta foi concluída cerca de cinco vezes mais rápido, em aproximadamente meio segundo.

```
postgres=>  SELECT COUNT(*) FROM (SELECT * FROM pgbench_accounts ORDER BY bid) AS
 accounts;
 count
-------
 5000000
(1 row)
Time: 567.095 ms
```

Ajustar a memória de trabalho para decodificação lógica

A replicação lógica está disponível em todas as versões do Aurora PostgreSQL desde sua introdução no PostgreSQL versão 10. Ao configurar a replicação lógica, você também pode definir o parâmetro `logical_decoding_work_mem` para especificar a quantidade de memória que o processo lógico de decodificação pode usar para o processo de decodificação e streaming.

Durante a decodificação lógica, os registros de log de gravação antecipada (WAL) são convertidos em instruções SQL que são então enviadas para outro destino para replicação lógica ou outra tarefa. Quando uma transação é gravada no WAL e depois convertida, toda a transação deve caber no valor especificado para `logical_decoding_work_mem`. Por padrão, esse parâmetro é definido como 65.536 KB. Qualquer excesso é gravado em disco. Isso significa que ele deve ser lido novamente do disco antes de ser enviado para seu destino, desacelerando assim o processo geral.

Você pode avaliar a quantidade de excesso de transações em sua workload atual em um momento específico usando a função `aurora_stat_file` conforme mostrado no exemplo a seguir.

```
SELECT split_part (filename, '/', 2)
   AS slot_name, count(1) AS num_spill_files,
   sum(used_bytes) AS slot_total_bytes,
   pg_size_pretty(sum(used_bytes)) AS slot_total_size
   FROM aurora_stat_file()
   WHERE filename like '%spill%'
   GROUP BY 1;
  slot_name | num_spill_files | slot_total_bytes | slot_total_size
------------+-----------------+------------------+-----------------
 slot_name  |       590       |      411600000   | 393 MB
(1 row)
```

Essa consulta retorna a contagem e o tamanho dos arquivos de despejo no seu cluster de banco de dados do Aurora PostgreSQL quando a consulta é invocada. Workloads de longa duração podem ainda não ter nenhum arquivo de despejo em disco. Para traçar o perfil de workloads de longa duração, recomendamos criar uma tabela para capturar as informações do arquivo de despejo à medida que a workload é executada. Você pode criar a tabela da seguinte maneira.

```
CREATE TABLE spill_file_tracking AS
    SELECT now() AS spill_time,*
    FROM aurora_stat_file()
    WHERE filename LIKE '%spill%';
```

Para ver como os arquivos de despejo são usados durante a replicação lógica, configure um editor e um assinante e inicie uma replicação simples. Para obter mais informações, consulte Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL. Com a replicação em andamento, você pode criar um trabalho que capture o conjunto de resultados da função de arquivo de despejo `aurora_stat_file()`, da maneira a seguir.

```
INSERT INTO spill_file_tracking
  SELECT now(),*
  FROM aurora_stat_file()
  WHERE filename LIKE '%spill%';
```

Use o comando psql a seguir para executar o trabalho uma vez por segundo.

```
\watch 0.5
```

Enquanto o trabalho estiver em execução, conecte-se à instância do gravador com base em outra sessão psql. Use a série de instruções a seguir para executar uma workload que exceda a configuração da memória e faça com que o Aurora PostgreSQL crie um arquivo de despejo.

```
labdb=> CREATE TABLE my_table (a int PRIMARY KEY, b int);
CREATE TABLE
labdb=> INSERT INTO my_table SELECT x,x FROM generate_series(0,10000000) x;
INSERT 0 10000001
labdb=> UPDATE my_table SET b=b+1;
UPDATE 10000001
```

O processo pode demorar vários minutos para ser concluído. Ao terminar, pressione a tecla Ctrl e a tecla C ao mesmo tempo para interromper a função de monitoramento. Depois, use o comando a seguir para criar uma tabela a fim de armazenar as informações sobre o uso do arquivo de despejo do cluster de banco de dados do Aurora PostgreSQL.

```
SELECT spill_time, split_part (filename, '/', 2)
    AS slot_name, count(1)
    AS spills, sum(used_bytes)
    AS slot_total_bytes, pg_size_pretty(sum(used_bytes))
    AS slot_total_size FROM spill_file_tracking
  GROUP BY 1,2 ORDER BY 1;
                  spill_time | slot_name             | spills | slot_total_bytes |
 slot_total_size
-------------------------------+-----------------------+--------+------------------
+-----------------
2022-04-15 13:42:52.528272+00 | replication_slot_name | 1      | 142352280        | 136
 MB
2022-04-15 14:11:33.962216+00 | replication_slot_name | 4      | 467637996        | 446
 MB
2022-04-15 14:12:00.997636+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:03.030245+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:05.059761+00 | replication_slot_name | 5      | 618410996        | 590
 MB
2022-04-15 14:12:07.22905+00  | replication_slot_name | 5      | 640585316        | 611
 MB
(6 rows)
```

O resultado mostra que a execução do exemplo criou cinco arquivos de despejo que usaram 611 MB de memória. Para evitar a gravação em disco, recomendamos definir o parâmetro `logical_decoding_work_mem` com o maior tamanho de memória subsequente, 1.024.

Usar métricas do Amazon CloudWatch para analisar o uso de recursos do Aurora PostgreSQL

O Aurora envia dados de métrica automaticamente para o CloudWatch em períodos de um minuto. Você pode analisar o uso de recursos do Aurora PostgreSQL usando métricas do CloudWatch. Você pode avaliar o throughput e o uso da rede com as métricas.

Avaliar o throughput da rede com o CloudWatch

Quando o uso do sistema se aproxima dos limites de recursos do seu tipo de instância, o processamento pode ficar lento. Você pode usar o CloudWatch Logs Insights para monitorar o uso dos recursos de armazenamento e garantir que recursos suficientes estejam disponíveis. Quando necessário, você pode modificar a instância de banco de dados para uma classe de instância maior.

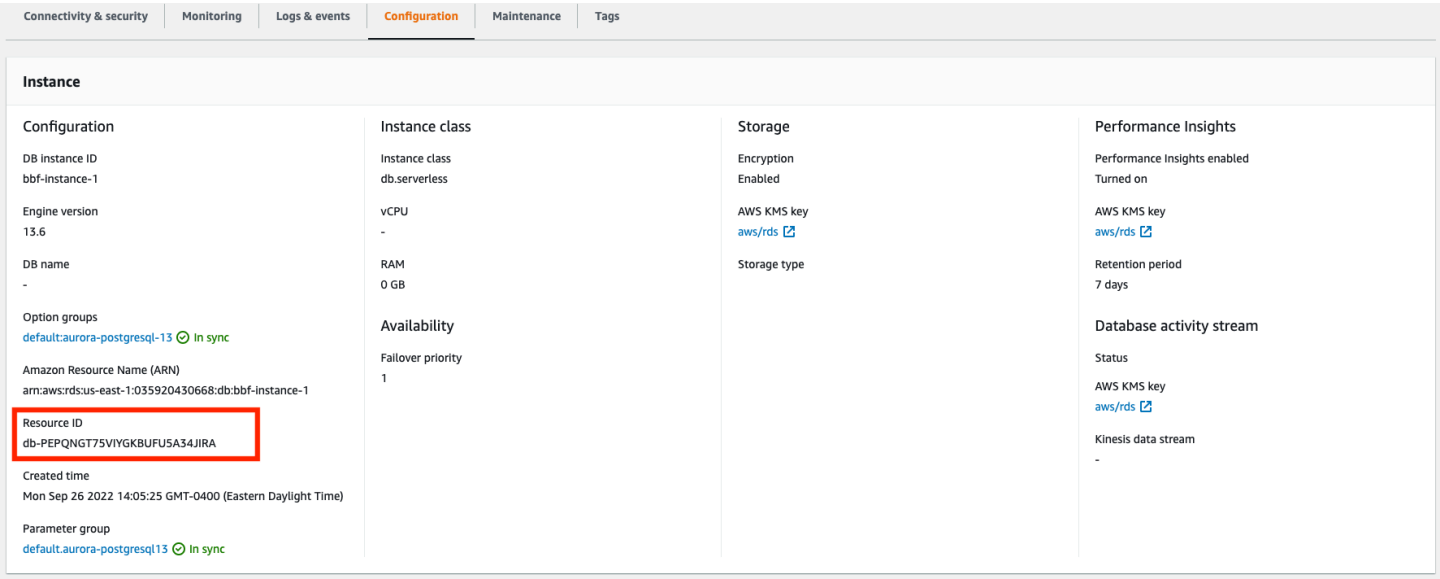
O processamento do armazenamento do Aurora pode ser lento devido a:

- Largura de banda da rede insuficiente entre o cliente e a instância de banco de dados.
- Largura de banda da rede insuficiente para o subsistema de armazenamento.
- Uma workload grande para seu tipo de instância.

Você pode consultar o CloudWatch Logs Insights para gerar um gráfico do uso dos recursos de armazenamento do Aurora para monitorar os recursos. O gráfico mostra a utilização e as métricas da CPU para ajudar você a decidir se deseja aumentar a escala verticalmente para um tamanho de instância maior. Para obter mais informações sobre a sintaxe de consulta do CloudWatch Logs Insights, consulte Sintaxe de consulta do CloudWatch Logs Insights.

Para usar o CloudWatch, você precisa exportar seus arquivos de log do Aurora PostgreSQL para o CloudWatch. Você também pode modificar o cluster existente para exportar logs para o CloudWatch. Para obter informações sobre como exportar logs para o CloudWatch, consulte Ativar a opção de publicação de logs no Amazon CloudWatch.

Você precisa do Resource ID (ID do recurso) da sua instância de banco de dados para consultar o CloudWatch Logs Insights. O Resource ID (ID do recurso) está disponível na guia Configuration (Configuração) em seu console:



Para consultar seus arquivos de log a fim de obter métricas de armazenamento de recursos:

1. Abra o console do CloudWatch, em https://console.aws.amazon.com/cloudwatch/.

   A página inicial de visão geral do CloudWatch é exibida.

2. Se necessário, altere a Região da AWS. Na barra de navegação, selecione a Região da AWS na qual seus recursos da AWS estão localizados. Para obter mais informações, consulte Regiões e endpoints.

3. No painel de navegação, escolha Logs e selecione Logs Insights.

   A página Logs Insights é exibida.

4. Selecione os arquivos de log na lista suspensa para analisar.

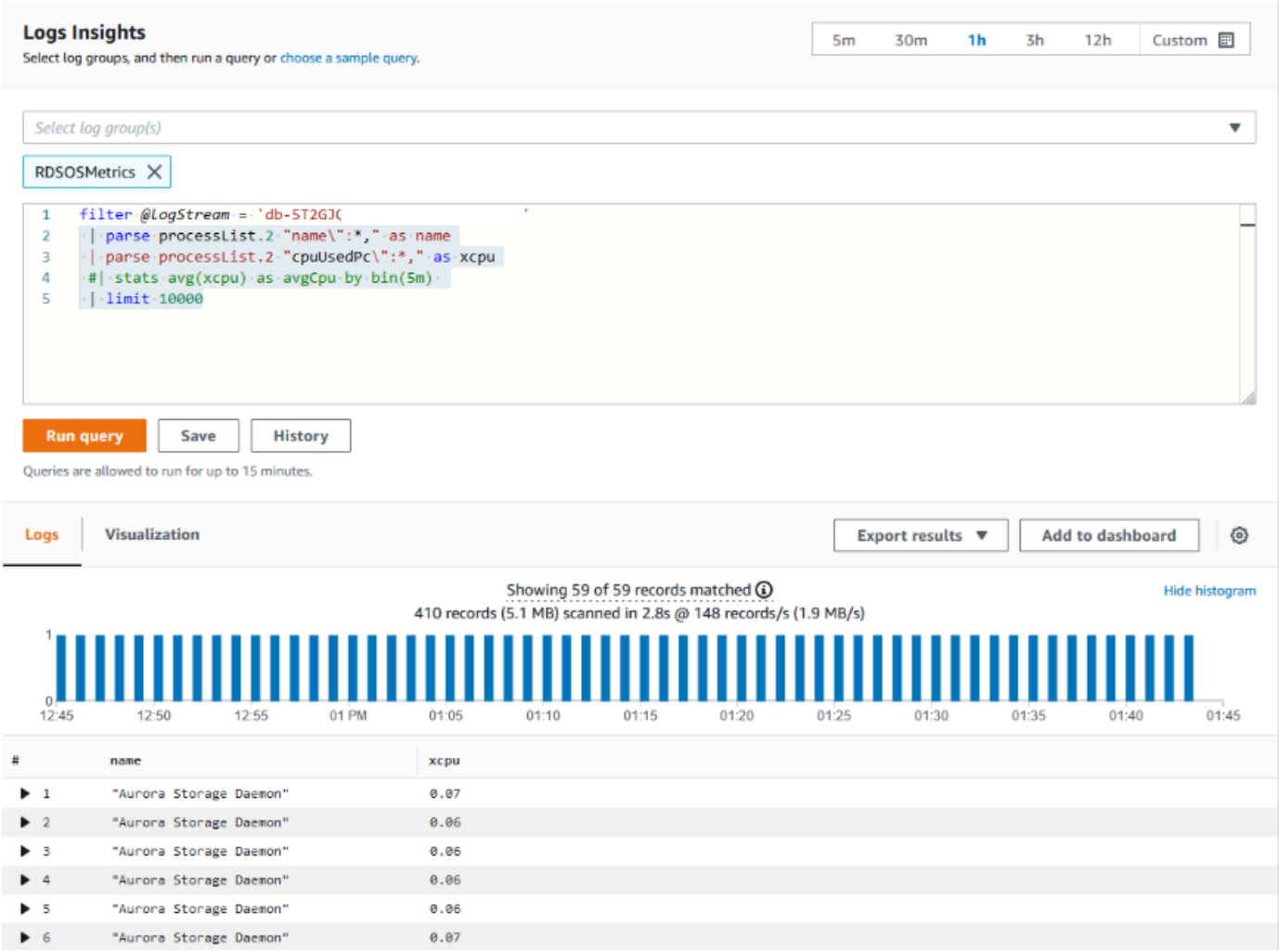
5. Insira a seguinte consulta no campo, substituindo `<resource ID>` pelo ID do recurso do seu cluster de banco de dados:

   ```
   filter @logStream = <resource ID> | parse @message "\"Aurora Storage
   Daemon\"*memoryUsedPc\":*,\"cpuUsedPc\":*," as a,memoryUsedPc,cpuUsedPc
   | display memoryUsedPc,cpuUsedPc #| stats avg(xcpu) as avgCpu by
   bin(5m) | limit 10000
   ```

6. Clique em Run query (Executar consulta).

   O gráfico de utilização do armazenamento é exibido.

   A imagem a seguir fornece a página Logs Insights e a exibição do gráfico.



Avaliação do uso de instâncias de banco de dados para Aurora PostgreSQL com métricas do CloudWatch

Você pode usar as métricas do CloudWatch para monitorar o throughput de sua instância e descobrir se sua classe de instância fornece recursos suficientes para suas aplicações. Para obter informações sobre os limites da classe da sua instância de banco de dados, acesse Especificações de hardware para classes de instância de banco de dadospara o Aurora e localize as especificações da sua classe de instância de banco de dados para descobrir a performance da sua rede.

Se o uso da sua instância de banco de dados estiver próximo do limite da classe da instância, a performance poderá começar a diminuir. As métricas do CloudWatch podem confirmar essa situação para que você possa planejar o aumento vertical manual da escala para uma classe de instância maior.

Combine os seguintes valores de métricas do CloudWatch para descobrir se você está se aproximando do limite da classe da instância:

- NetworkThroughput: o throughput de rede recebido e transmitido pelos clientes para cada instância no cluster de bancos de dados do Aurora. Esse valor de throughput não inclui o tráfego de rede entre instâncias no cluster de banco de dados e o volume do cluster.
- StorageNetworkThroughput: o throughput da rede recebido e enviado ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora.

Adicione NetworkThroughput a StorageNetworkThroughput para encontrar o throughput da rede recebido e enviado ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados do Aurora. O limite da classe de instância para sua instância deve ser maior do que a soma dessas duas métricas combinadas.

Você pode usar as seguintes métricas para analisar detalhes adicionais do tráfego de rede das aplicações clientes ao enviar e receber:

- NetworkReceiveThroughput: a quantidade de throughput de rede recebida dos clientes por cada instância no cluster de banco de dados do Aurora PostgreSQL. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de banco de dados do e o volume do cluster.
- NetworkTransmitThroughput: a quantidade de throughput de rede enviada aos clientes por cada instância no cluster de bancos de dados do Aurora. Essa taxa de transferência não inclui o tráfego de rede entre instâncias no cluster de banco de dados do e o volume do cluster.
- StorageNetworkReceiveThroughput: a quantidade de throughput de rede recebida do subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.
- StorageNetworkTransmitThroughput: a quantidade de throughput de rede enviada ao subsistema de armazenamento do Aurora por cada instância no cluster de banco de dados.

Adicione todas essas métricas para avaliar como o uso da rede se compara ao limite da classe da instância. O limite da classe de instância deve ser maior do que a soma dessas métricas combinadas.

Os limites de rede e a utilização da CPU para armazenamento são mútuos. Quando o throughput de rede aumenta, a utilização da CPU também aumenta. O monitoramento do uso da CPU e da rede fornece informações sobre como e por que os recursos estão sendo esgotados.

Para ajudar a minimizar o uso da rede, você pode considerar:

- Usar uma classe de instância maior.
- Usar estratégias de particionamento `pg_partman`.
- Dividir as solicitações de gravação em lotes para reduzir o total de transações.
- Direcionar a workload somente leitura para uma instância somente leitura.
- Excluir todos os índices não utilizados.
- Verificar se há objetos inchados e VACUUM. Em caso de inchaço severo, use a extensão `pg_repack` para PostgreSQL. Para obter mais informações sobre `pg_repack`, consulte Reorganize tables in PostgreSQL databases with minimal locks (Reorganizar tabelas em bancos de dados PostgreSQL com bloqueios mínimos).

Usar replicação lógica para realizar uma atualização de versão principal do Aurora PostgreSQL

Ao usar a replicação lógica e a clonagem rápida do Aurora, você pode realizar um upgrade de versão principal que usa a versão atual do banco de dados Aurora PostgreSQL enquanto migra gradualmente os dados alterados para o novo banco de dados da versão principal. Esse processo de upgrade com baixo tempo de inatividade é chamado de upgrade azul/verde. A versão atual do banco de dados é chamada de ambiente “azul” e a nova versão do banco de dados é chamada de ambiente “verde”.

A clonagem rápida do Aurora carrega totalmente os dados existentes ao tirar um snapshot do banco de dados de origem. A clonagem rápida usa um protocolo de cópia em gravação construído sobre a camada de armazenamento do Aurora, que permite criar um clone do banco de dados em pouco tempo. Esse método é muito eficaz ao realizar a atualização para um banco de dados grande.

A replicação lógica no PostgreSQL rastreia e transfere suas alterações de dados da instância inicial para uma nova instância executada em paralelo até que você mude para a versão mais recente do PostgreSQL. A replicação lógica usa um modelo de publicação e de assinatura. Para obter mais informações sobre a replicação lógica do Aurora PostgreSQL, consulte Replicação com Amazon Aurora PostgreSQL.

> **Tip**
>
> É possível minimizar o tempo de inatividade necessário para a atualização da versão principal utilizando o recurso de implantação azul/verde gerenciado do Amazon RDS. Para

obter mais informações, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Tópicos

- Requisitos
- Limitações
- Definir e conferir valores de parâmetros
- Atualizar o Aurora PostgreSQL para uma nova versão principal
- Executar tarefas de pós-atualização

Requisitos

Você deve cumprir os seguintes requisitos para realizar esse processo de upgrade com baixo tempo de inatividade:

- Você deve ter permissões rds_superuser.
- O cluster de banco de dados do Aurora PostgreSQL que você pretende fazer upgrade deve estar executando uma versão compatível que possa realizar upgrades de versão principal usando replicação lógica. Aplique patches e atualizações da versão secundária ao cluster de banco de dados. A função `aurora_volume_logical_start_lsn` usada nessa técnica é compatível com as seguintes versões do Aurora PostgreSQL:
  - Versão 15.2 e versões 15 posteriores
  - Versão 14.3 e versões 14 posteriores
  - Versão 13.6 e versões 13 posteriores
  - Versão 12.10 e versões 12 posteriores
  - Versão 11.15 e versões 11 posteriores
  - Versão 10.20 e versões 10 posteriores

  Para obter mais informações sobre a função `aurora_volume_logical_start_lsn`, consulte aurora_volume_logical_start_lsn.
- Todas as suas tabelas devem ter uma chave primária ou incluir uma coluna de identidade do PostgreSQL.

- Configure o grupo de segurança para sua VPC para permitir acesso de entrada e saída entre os dois clusters de banco de dados do Aurora PostgreSQL, antigo e novo. Você pode conceder acesso a um intervalo Encaminhamento Entre Domínios Sem Classificação (CIDR) específico ou a outro grupo de segurança em sua VPC ou em uma VPC par. (A VPC de par requer uma conexão de emparelhamento da VPC.)

> **Note**
>
> Para obter informações detalhadas sobre as permissões necessárias para configurar e gerenciar um cenário de replicação lógica em execução, consulte a documentação principal do PostgreSQL.

Limitações

Ao realizar um upgrade com baixo tempo de inatividade em seu cluster de banco de dados do Aurora PostgreSQL para uma nova versão principal, você usa o atributo nativo de replicação lógica do PostgreSQL. Ele tem os mesmos recursos e limitações da replicação lógica do PostgreSQL. Para obter mais informações, consulte Replicação lógica do PostgreSQL.

- Comandos DDL (Data Definition Language) não são replicados.
- A replicação não é compatível com alterações de esquema em um banco de dados ativo. O esquema é recriado em sua forma original durante o processo de clonagem. Se você alterar o esquema após a clonagem, mas antes de concluir a atualização, isso não será refletido na instância atualizada.
- Objetos grandes não são replicados, mas você pode armazenar dados em tabelas normais.
- A replicação só é compatível com tabelas, por exemplo, tabelas particionadas. A replicação para outros tipos de relações, como visualizações, visões materializadas ou tabelas externas, não é compatível.
- Os dados da sequência não são replicados e exigem uma atualização manual após o failover.

> **Note**
>
> Essa atualização não é compatível com a criação de scripts automáticos. Você deve executar todas as etapas manualmente.

Definir e conferir valores de parâmetros

Antes de atualizar, configure a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para atuar como um servidor de publicação. A instância deve usar um grupo de parâmetros do cluster de banco de dados personalizado com as seguintes configurações:

- `rds.logical_replication`: defina este parâmetro como 1. O parâmetro `rds.logical_replication` tem a mesma finalidade do parâmetro `wal_level` de um servidor PostgreSQL independente e de outros parâmetros que controlam o gerenciamento de arquivos de log de gravação antecipada.
- `max_replication_slots`: defina esse parâmetro como o número total de assinaturas que você pretende criar. Se você estiver usando o AWS DMS, defina esse parâmetro como o número de tarefas AWS DMS que você planeja usar para a captura de dados de alteração desse cluster de banco de dados.
- `max_wal_senders`: defina o número de conexões simultâneas, além de algumas extras, para disponibilizar para tarefas de gerenciamento e novas sessões. Se você estiver usando AWS DMS, o número de max_wal_senders deve ser igual ao número de sessões simultâneas mais o número de AWS DMS tarefas que podem estar funcionando a qualquer momento.
- `max_logical_replication_workers`: defina como o número de operadores de replicação lógica e operadores de sincronização de tabelas que você espera. Geralmente, é seguro definir o número de operadores de replicação como o mesmo valor usado para max_wal_senders. Os operadores são retirados do grupo de processos em segundo plano (max_worker_processes) alocados para o servidor.
- `max_worker_processes`: defina o número de processos em segundo plano para o servidor. Esse número deve ser grande o suficiente para alocar operadores para replicação, processos de autovacuum e outros processos de manutenção que possam ocorrer simultaneamente.

Ao atualizar para uma versão mais recente do Aurora PostgreSQL, você precisa duplicar todos os parâmetros que você modificou na versão anterior do grupo de parâmetros. Esses parâmetros são aplicados à versão atualizada. Você pode consultar a tabela `pg_settings` para obter uma lista de configurações de parâmetros para poder recriá-las em seu novo cluster de banco de dados do Aurora PostgreSQL.

Por exemplo, para obter as configurações dos parâmetros de replicação, execute a seguinte consulta:

```
SELECT name, setting FROM pg_settings WHERE name in
```

```
('rds.logical_replication', 'max_replication_slots',
'max_wal_senders', 'max_logical_replication_workers',
'max_worker_processes');
```

Atualizar o Aurora PostgreSQL para uma nova versão principal

Como preparar o editor (azul)

1. No exemplo a seguir, a instância do gravador de código-fonte (azul) é um cluster de banco de dados do Aurora PostgreSQL executando o PostgreSQL versão 11.15. Esse é o nó de publicação em nosso cenário de replicação. Para essa demonstração, nossa instância de gravador de origem hospeda uma tabela de amostra que contém uma série de valores:

   ```
   CREATE TABLE my_table (a int PRIMARY KEY);
   INSERT INTO my_table VALUES (generate_series(1,100));
   ```

2. Para criar uma publicação na instância de origem, conecte-se ao nó de gravação da instância com psql (a CLI do PostgreSQL) ou com o cliente de sua escolha). Insira o seguinte comando em cada banco de dados:

   ```
   CREATE PUBLICATION publication_name FOR ALL TABLES;
   ```

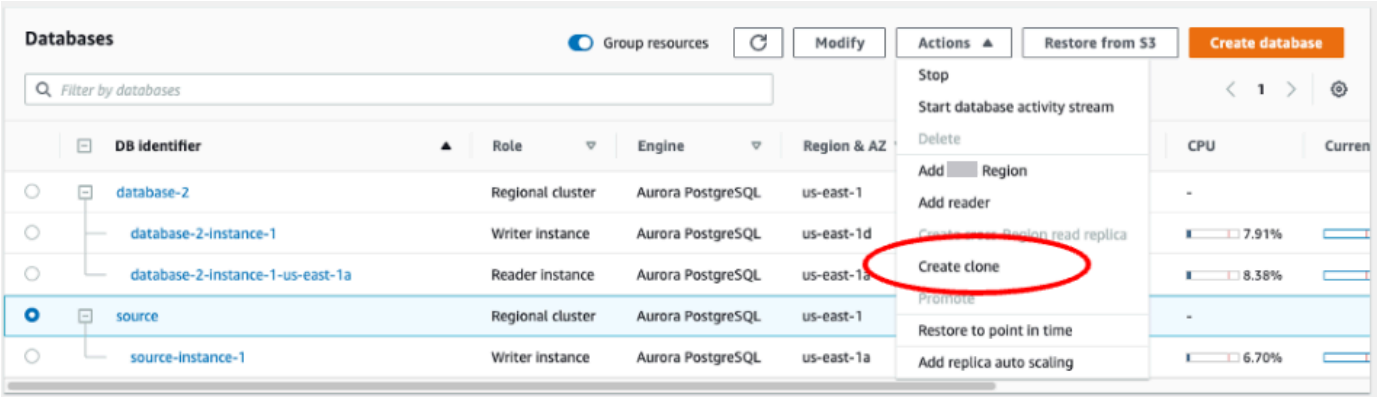
   O publication_name especifica o nome da publicação.

3. Também é necessário criar um slot de replicação na instância. O comando a seguir cria um slot de replicação e carrega o plug-in de decodificação lógica `pgoutput`. O plug-in altera o conteúdo lido do registro em log de gravação antecipada (WAL) para o protocolo de replicação lógica e filtra os dados de acordo com a especificação da publicação.

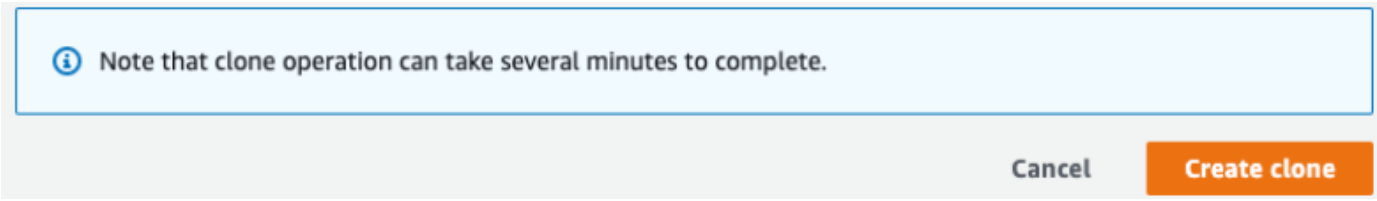
   ```
   SELECT pg_create_logical_replication_slot('replication_slot_name', 'pgoutput');
   ```

Como clonar o editor

1. Use o console do Amazon RDS para criar um clone da instância de origem. Destaque o nome da instância no console do Amazon RDS e selecione Create clone (Criar clone) no menu Actions (Ações).



2. Forneça um nome único para a instância. A maioria das configurações é padrões da instância de origem. Quando você tiver feito as alterações necessárias para a nova instância, selecione Create clone (Criar clone).



3. Enquanto a instância de destino está sendo iniciada, a coluna Status do nó do gravador exibe Creating (Criando) na coluna Status. Quando a instância estiver pronta, seu status mudará para Available (Disponível).

Como preparar o clone para uma atualização

1. O clone é a instância “verde” no modelo de implantação. É o host do nó de assinatura de replicação. Quando o nó estiver disponível, conecte-se ao psql e consulte o novo nó do gravador para obter o número de sequência de log (LSN). O LSN identifica o início de um registro no fluxo WAL.

```
SELECT aurora_volume_logical_start_lsn();
```

2. Na resposta da consulta, você encontra o número LSN. Você precisará desse número posteriormente no processo, então anote-o.

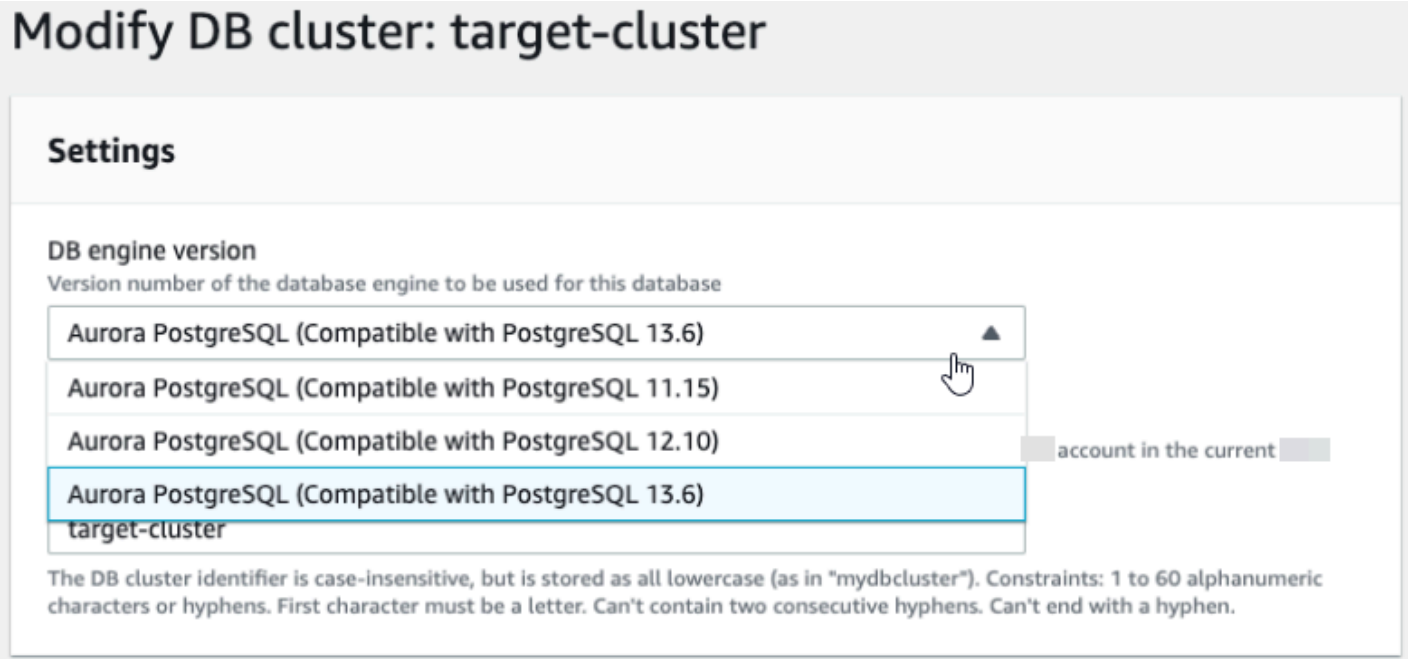
```
postgres=> SELECT aurora_volume_logical_start_lsn();
aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

3. Antes de atualizar o clone, descarte o slot de replicação do clone.

```
SELECT pg_drop_replication_slot('replication_slot_name');
```

Como atualizar um cluster para uma nova versão principal

- Depois de clonar o nó do provedor, use o console do Amazon RDS para iniciar uma atualização de versão principal no nó de assinatura. Destaque o nome da instância no console do RDS e selecione o botão Modify (Modificar). Selecione a versão atualizada e seus grupos de parâmetros atualizados e aplique as configurações imediatamente para atualizar a instância de destino.



- Você também pode usar a CLI para realizar uma atualização:

```
aws rds modify-db-cluster –db-cluster-identifier $TARGET_Aurora_ID –engine-version
 13.6 –allow-major-version-upgrade –apply-immediately
```

Como preparar o assinante (verde)

1. Quando o clone ficar disponível depois do upgrade, conecte-se com o psql e defina a assinatura. Para fazer isso, é necessário especificar as seguintes opções no comando `CREATE SUBSCRIPTION`:

   - `subscription_name`: o nome da inscrição.

- `admin_user_name`: o nome de um usuário administrativo com permissões rds_superuser.
- `admin_user_password`: a senha associada ao usuário administrativo.
- `source_instance_URL`: o URL da instância do servidor de publicação.
- `database`: o banco de dados ao qual o servidor de assinatura se conectará.
- `publication_name`: o nome do servidor de publicação.
- `replication_slot_name`: o nome do slot de replicação.

```
CREATE SUBSCRIPTION subscription_name
CONNECTION 'postgres://admin_user_name:admin_user_password@source_instance_URL/
database' PUBLICATION publication_name
WITH (copy_data = false, create_slot = false, enabled = false, connect = true,
 slot_name = 'replication_slot_name');
```

2. Depois de criar a assinatura, consulte a visualização pg_replication_origin para recuperar o valor do roname, que é o identificador da origem da replicação. Cada instância tem um `roname`:

```
SELECT * FROM pg_replication_origin;
```

Por exemplo:

```
postgres=>
SELECT * FROM pg_replication_origin;

roident | roname
---------+----------
1 | pg_24586
```

3. Forneça a LSN que você salvou da consulta anterior do nó de publicação e o `roname` retornado do nó de assinatura [INSTANCE] no comando. Esse comando usa a função pg_replication_origin_advance para especificar o ponto de partida na sequência de log para replicação.

```
SELECT pg_replication_origin_advance('roname', 'log_sequence_number');
```

`roname` é o identificador retornado pela visualização pg_replication_origin.

`log_sequence_number` é o valor retornado pela consulta anterior da função `aurora_volume_logical_start_lsn`.

4. Depois, use a cláusula `ALTER SUBSCRIPTION... ENABLE` para ativar a replicação lógica.

```
ALTER SUBSCRIPTION subscription_name ENABLE;
```

5. Nesse momento, você pode confirmar que a replicação está funcionando. Adicione um valor à instância de publicação e, depois, confirme se o valor foi replicado no nó de assinatura.

   Depois, use o seguinte comando para monitorar o atraso de replicação no nó de publicação:

```
SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';
```

   Por exemplo:

```
postgres=> SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
 confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';

current_time                  | slot_name             | active | active_pid |
 diff_size | diff_bytes
-------------------------------+-----------------------+--------+------------
+-----------+------------
2022-04-13 15:11:00.243401+00  | replication_slot_name | t      | 21854      | 136
 bytes | 136
(1 row)
```

   Você pode monitorar o atraso na replicação usando os valores `diff_size` e `diff_bytes`. Quando esses valores atingem 0, isso mostra que a réplica alcançou a instância do banco de dados de origem.

Executar tarefas de pós-atualização

Quando a atualização for concluída, o status da instância será exibido como Available (Disponível) na coluna Status do painel do console. Na nova instância, recomendamos fazer o seguinte:

- Redirecione suas aplicações para apontar para o nó do gravador.
- Adicione nós de leitura para gerenciar o número de casos e fornecer alta disponibilidade no caso de um problema com o nó do gravador.
- Os clusters de banco de dados do Aurora PostgreSQL ocasionalmente exigem atualizações do sistema operacional. Essas atualizações às vezes incluem uma versão mais recente da biblioteca glibc. Durante essas atualizações, recomendamos que você siga as diretrizes descritas em Agrupamentos permitidos no Aurora PostgreSQL..
- Atualize as permissões do usuário na nova instância para garantir o acesso.

Depois de testar sua aplicação e seus dados na nova instância, recomendamos que você faça um backup final da instância inicial antes de removê-la. Para ter mais informações sobre o uso de replicação lógica em um host do Aurora, consulte Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL.

Gerenciar conversões personalizadas no Aurora PostgreSQL

No PostgreSQL, a conversão de tipo refere-se ao processo de conversão de um valor de um tipo de dados em outro. O PostgreSQL oferece conversões integradas para várias conversões comuns, mas também é possível criar conversões personalizadas para definir como as conversões de tipos específicos devem se comportar.

Um conversão especifica como realizar uma conversão de um tipo de dados em outro. Por exemplo, converter texto `'123'` em número inteiro `123` ou numérico `45.67` em texto `'45.67'`.

Para ter informações abrangentes sobre os conceitos e a sintaxe de conversão do PostgreSQL, consulte CREATE CAST na documentação do PostgreSQL.

A partir das versões do Aurora PostgreSQL 13.23, 14.20, 15.15, 16.11, 17.7 e 18.1, é possível usar a extensão rds_casts para instalar outras conversões para tipos integrados e, ao mesmo tempo, criar conversões próprias para tipos personalizados

Tópicos

- Instalar e usar a extensão rds_casts

- Conversões compatíveis
- Criar ou eliminar conversões
- Criar conversões personalizadas com a estratégia de contexto adequada

Instalar e usar a extensão rds_casts

Para criar a extensão `rds_casts`, conecte-se à instância do gravador do cluster de banco de dados do Aurora PostgreSQL como um `rds_superuser` e execute o seguinte comando:

```
CREATE EXTENSION IF NOT EXISTS rds_casts;
```

Conversões compatíveis

Crie a extensão em cada banco de dados em que você deseja usar conversões personalizadas. Depois de criar a extensão, use o seguinte comando para visualizar todas as conversões disponíveis:

```
SELECT * FROM rds_casts.list_supported_casts();
```

Essa função lista as combinações de conversão disponíveis (tipo de origem, tipo de destino, contexto de coerção e função de conversão). Por exemplo, se você quiser criar uma conversão de `text` em `numeric` como uma conversão `implicit`. É possível usar a seguinte consulta para descobrir se a conversão está disponível para criação:

```
SELECT * FROM rds_casts.list_supported_casts()
WHERE source_type = 'text' AND target_type = 'numeric';
 id | source_type | target_type |          qualified_function          |
 coercion_context
----+-------------+-------------+--------------------------------------
+------------------
 10 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | implicit
 11 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | assignment
 13 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | explicit
 20 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | implicit
 21 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | assignment
 23 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | explicit
```

A extensão rds_casts oferece dois tipos de função de conversão para cada conversão:

- _inout functions: usam o mecanismo de conversão de E/S padrão do PostgreSQL, comportando-se de forma idêntica às conversões criadas com o método INOUT.
- _custom functions: oferecem lógica de conversão aprimorada que lida com casos extremos, como converter strings vazias em valores NULL para evitar erros de conversão

As funções `inout` replicam o comportamento de conversão nativo do PostgreSQL, enquanto as funções `custom` estendem essa funcionalidade, lidando com cenários que as conversões INOUT padrão não conseguem atender, como converter strings vazias em números inteiros.

Criar ou eliminar conversões

É possível criar e eliminar conversões compatíveis usando dois métodos:

Criação de conversão

Método 1: usar o comando nativo CREATE CAST

```
CREATE CAST (text AS numeric)
WITH FUNCTION rds_casts.rds_text_to_numeric_custom
AS IMPLICIT;
```

Método 2: usar a função rds_casts.create_cast

```
SELECT rds_casts.create_cast(10);
```

A função `create_cast` obtém o ID da saída `list_supported_casts()`. Esse método é mais simples e garante o uso da combinação correta de função e contexto. Esse ID com certeza permanecerá o mesmo em diferentes versões do Postgres.

Para verificar se a conversão foi criada com êxito, consulte o catálogo de sistema pg_cast:

```
SELECT oid, castsource::regtype, casttarget::regtype, castfunc::regproc, castcontext,
 castmethod
FROM pg_cast
WHERE castsource = 'text'::regtype AND casttarget = 'numeric'::regtype;
  oid   | castsource | casttarget |              castfunc              | castcontext
 | castmethod
--------+------------+------------+------------------------------------+-------------
+------------
```

```
 356372 | text       | numeric    | rds_casts.rds_text_to_numeric_custom | i
 | f
```

A coluna `castcontext` mostra: e para EXPLICIT, a para ASSIGNMENT ou `i` para IMPLICIT.

Eliminar conversões

Método 1: usar o comando DROP CAST

```
DROP CAST IF EXISTS (text AS numeric);
```

Método 2: usar a função rds_casts.drop_cast

```
SELECT rds_casts.drop_cast(10);
```

A função `drop_cast` usa o mesmo ID usado ao criar a conversão. Esse método garante a eliminação da conversão exata que foi criada com o ID correspondente.

Criar conversões personalizadas com a estratégia de contexto adequada

Ao criar várias conversões para tipos inteiros, erros de ambiguidade de operador poderão ocorrer se todas as conversões forem criadas como IMPLICIT. O seguinte exemplo demonstra esse problema criando duas conversões implícitas de texto em números inteiros com largura diferente:

```
-- Creating multiple IMPLICIT casts causes ambiguity
postgres=> CREATE CAST (text AS int4) WITH FUNCTION
 rds_casts.rds_text_to_int4_custom(text) AS IMPLICIT;
CREATE CAST
postgres=> CREATE CAST (text AS int8) WITH FUNCTION
 rds_casts.rds_text_to_int8_custom(text) AS IMPLICIT;
CREATE CAST

postgres=> CREATE TABLE test_cast(col int);
CREATE TABLE
postgres=> INSERT INTO test_cast VALUES ('123'::text);
INSERT 0 1
postgres=> SELECT * FROM test_cast WHERE col='123'::text;
ERROR:  operator is not unique: integer = text
LINE 1: SELECT * FROM test_cast WHERE col='123'::text;
                                        ^
```

```
HINT:  Could not choose a best candidate operator. You might need to add explicit type
 casts.
```

O erro ocorre porque o PostgreSQL não consegue determinar qual conversão implícita usar ao comparar uma coluna de inteiros com um valor de texto. As conversões implícitas int4 e int8 são ambas candidatas válidas, o que cria ambiguidade.

Para evitar essa ambiguidade de operador, use o contexto ASSIGNMENT para inteiros com largura menor e o contexto IMPLICIT para inteiros com largura maior:

```
-- Use ASSIGNMENT for smaller integer widths
CREATE CAST (text AS int2)
WITH FUNCTION rds_casts.rds_text_to_int2_custom(text)
AS ASSIGNMENT;

CREATE CAST (text AS int4)
WITH FUNCTION rds_casts.rds_text_to_int4_custom(text)
AS ASSIGNMENT;

-- Use IMPLICIT for larger integer widths
CREATE CAST (text AS int8)
WITH FUNCTION rds_casts.rds_text_to_int8_custom(text)
AS IMPLICIT;

postgres=> INSERT INTO test_cast VALUES ('123'::text);
INSERT 0 1
postgres=> SELECT * FROM test_cast WHERE col='123'::text;
 col
-----
 123
(1 row)
```

Com essa estratégia, somente a conversão int8 é implícita, possibilitando que o PostgreSQL determine de forma inequívoca qual conversão usar.

Práticas recomendadas para consultas paralelas no Aurora PostgreSQL

A execução de consultas paralelas é um recurso do PostgreSQL que permite que uma única consulta SQL seja dividida em tarefas menores que são realizadas simultaneamente por vários processos de operador em segundo plano. Em vez de executar uma consulta inteiramente em um único processo de backend, o PostgreSQL pode distribuir partes da consulta, como verificações,

junções, agregações ou classificações, em vários núcleos de CPU. O processo líder coordena essa execução e reúne os resultados dos operadores paralelos.

No entanto, para a maioria das workloads de produção, especialmente sistemas de processamento de transações on-line (OLTP) com alta simultaneidade, recomendamos desabilitar a execução automática de consultas paralelas. Embora o paralelismo possa acelerar as consultas em grandes conjuntos de dados em workloads de analytics ou geração de relatórios, ele apresenta riscos significativos que geralmente superam os benefícios em ambientes de produção movimentados.

A execução paralela também gera custos indiretos significativos. Cada operador paralelo é um processo completo de backend no PostgreSQL, que requer bifurcação (cópia de estruturas de memória e inicialização do estado do processo) e autenticação de processos (consumo de slots de conexão do limite de `max_connections`). Cada operador também consome sua própria memória, inclusive `work_mem` para operações de classificação e hash. Quando há vários operadores por consulta, o uso da memória se multiplica rapidamente (p. ex., 4 operadores × 64 MB de `work_mem` = 256 MB por consulta). Por isso, as consultas paralelas podem consumir consideravelmente mais recursos do sistema do que as consultas de processo único. Se elas não forem ajustadas adequadamente, poderá haver saturação da CPU (vários operadores consomem a capacidade de processamento disponível), aumento da troca de contexto (o sistema operacional alterna frequentemente entre vários processos de operador, aumentando os custos indiretos e reduzindo o throughput) ou exaustão da conexão (visto que cada operador paralelo consome um slot de conexão, uma única consulta com quatro operadores usa cinco conexões no total, uma líder + quatro operadores, o que pode esgotar rapidamente o grupo de conexões quando há alta simultaneidade, impedindo novas conexões de cliente e causando falhas na aplicação). Esses problemas são particularmente graves em workloads com alta simultaneidade, nas quais várias consultas podem tentar a execução paralela simultaneamente.

O PostgreSQL decide se deve usar o paralelismo com base em estimativas de custo. Em alguns casos, o planejador pode mudar automaticamente para um plano paralelo se isso parecer mais barato, mesmo quando não for o ideal na prática. Isso pode ocorrer se as estatísticas do índice estiverem desatualizadas ou se o inchaço fizer com que as verificações sequenciais pareçam mais atrativas do que as pesquisas de índice. Devido a esse comportamento, os planos paralelos automáticos às vezes podem provocar regressões no desempenho da consulta ou na estabilidade do sistema.

Para extrair o máximo benefício das consultas paralelas no Aurora PostgreSQL, é importante testá-las e ajustá-las com base na workload, monitorar o impacto sobre o sistema e desabilitar a seleção automática de planos paralelos em favor do controle em nível de consulta.

Parâmetros de configuração

O PostgreSQL usa vários parâmetros para controlar o comportamento e a disponibilidade de consultas paralelas. Compreendê-los e ajustá-los é fundamental para alcançar um desempenho previsível:

Parâmetro	Descrição	Padrão
`max_parallel_workers`	Número máximo de processos de operador em segundo plano que podem ser executados no total.	GREATEST($DBInstanceVCPU/2,8)
`max_parallel_workers_per_gather`	Número máximo de operadores por nó do plano de consulta (p. ex., por `Gather`).	2
`parallel_setup_cost`	Custo adicional do planejador para iniciar a infraestrutura de consulta paralela.	1.000
`parallel_tuple_cost`	Custo por tupla processada em modo paralelo (afeta a decisão do planejador).	0,1
`force_parallel_mode`	Força o planejador a testar planos paralelos (`off`, on, `regress`).	`off`

Considerações importantes

- `max_parallel_workers` controla o conjunto total de operadores paralelos. Se definido com um valor muito baixo, algumas consultas podem voltar à execução em série.
- `max_parallel_workers_per_gather` determina quantos operadores uma única consulta pode usar. Um valor maior aumenta a simultaneidade, mas também o uso de recursos.
- `parallel_setup_cost` e `parallel_tuple_cost` afetam o modelo de custo do planejador. Reduzi-los pode aumentar a probabilidade de os planos paralelos serem escolhidos.

- `force_parallel_mode` é útil para testes, mas não deve ser usado na produção, a menos que necessário.

> **Note**
>
> O valor padrão do `max_parallel_workers` parâmetro é calculado dinamicamente com base no tamanho da instância usando a fórmula`GREATEST($DBInstanceVCPU/2, 8)`. Isso significa que quando você escalar a computação de uma instância do Aurora para um tamanho maior com mais vCPUs, o número máximo de operadores paralelos disponíveis aumentará automaticamente. Por isso, consultas que antes eram executadas em série ou com paralelismo limitado podem utilizar mais operadores paralelos de maneira repentina após uma operação de aumento da escala vertical e possivelmente provocar aumentos inesperados no uso da conexão, na utilização da CPU e no consumo de memória. É importante monitorar o comportamento da consulta paralela após qualquer evento de ajuste de escala da computação e, se necessário, ajustar `max_parallel_workers_per_gather` para manter o uso previsível dos recursos.

Identificar o uso de consultas paralelas

As consultas podem mudar para planos paralelos com base na distribuição de dados ou nas estatísticas. Por exemplo:

```
SELECT count(*) FROM customers WHERE last_login < now() - interval '6 months';
```

Essa consulta pode usar um índice para dados recentes, mas mudar para uma verificação sequencial paralela de dados históricos.

É possível registrar em log os planos de execução de consultas carregando o módulo `auto_explain`. Para saber mais, consulte Como registrar em log planos de execução de consultas no Centro de Conhecimento da AWS.

É possível monitorar os planos de execução de consultas na instância de banco de dados do Aurora PostgreSQL para detectar os planos de execução que contribuem para a carga atual do banco de dados e monitorar as estatísticas de desempenho dos planos de execução ao longo do tempo usando o parâmetro `aurora_compute_plan_id`. Para saber mais, consulte Monitorar planos de execução de consultas e pico de memória do Aurora PostgreSQL

É possível monitorar eventos de espera no CloudWatch Database Insights relacionados à consulta paralela. Para saber mais sobre eventos de espera relacionados à consulta paralela, consulte IPC: eventos de espera paralelos.

A partir da versão 18 do PostgreSQL, é possível monitorar a atividade de operadores paralelos usando novas colunas em `pg_stat_database` e `pg_stat_statements`:

- `parallel_workers_to_launch`: número de operadores paralelos que se pretende iniciar.
- `parallel_workers_launched`: número de operadores paralelos realmente iniciados.

Essas métricas ajudam a identificar discrepâncias entre o paralelismo planejado e real, o que pode indicar restrições de recursos ou problemas de configuração. Use as seguintes consultas para monitorar a execução paralela:

Para métricas de operadores paralelos em nível de banco de dados:

```
SELECT datname, parallel_workers_to_launch, parallel_workers_launched
FROM pg_stat_database
WHERE datname = current_database();
```

Para métricas de operadores paralelos em nível de consulta:

```
SELECT query, parallel_workers_to_launch, parallel_workers_launched
FROM pg_stat_statements
ORDER BY parallel_workers_launched;
```

Como controlar o paralelismo

Há várias maneiras de controlar o paralelismo de consultas, cada uma projetada para diferentes cenários e requisitos.

Para desabilitar o paralelismo automático globalmente, modifique seu grupo de parâmetros para definir:

```
max_parallel_workers_per_gather = 0;
```

Para configurações persistentes e específicas do usuário, o comando ALTER ROLE oferece uma maneira para definir parâmetros que se aplicarão a todas as sessões futuras de determinado usuário.

Por exemplo:

`ALTER ROLE username SET max_parallel_workers_per_gather = 4;` garante que, toda vez que esse usuário se conectar ao banco de dados, as respectivas sessões usarão essa configuração de operador paralelo quando necessário.

O controle em nível de sessão pode ser obtido usando o comando SET, que modifica os parâmetros durante a sessão atual do banco de dados. Isso é particularmente útil quando é necessário ajustar temporariamente as configurações sem afetar outros usuários ou sessões futuras. Depois de definidos, esses parâmetros permanecem em vigor até serem redefinidos explicitamente ou até que a sessão termine. Os comandos são simples:

```
SET max_parallel_workers_per_gather = 4;
-- Run your queries
RESET max_parallel_workers_per_gather;
```

Para ter um controle ainda mais granular, o SET LOCAL permite modificar os parâmetros de uma única transação. Isso é ideal quando é necessário ajustar as configurações de um conjunto específico de consultas em uma transação e, após isso, reverter automaticamente as configurações para os valores anteriores. Essa abordagem ajuda a evitar efeitos indesejados em outras operações na mesma sessão.

Utilize o gerenciamento de planos de consultas (QPM)

No Aurora PostgreSQL, o atributo de gerenciamento de planos de consulta (QPM) foi projetado para garantir a adaptabilidade e a estabilidade do plano, independentemente das alterações no ambiente de banco de dados que poderiam causar a regressão do plano de consulta. Para ter mais informações, consulte Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL. O QPM fornece algum controle sobre o otimizador. Revise os planos aprovados no QPM para garantir que estejam alinhados com as configurações atuais de paralelismo. Atualize ou remova planos desatualizados que possam estar forçando uma execução paralela subótima.

Você também pode corrigir os planos usando pg_hint_plan. Para obter mais informações, consulte Corrigir planos usando pg_hint_plan. Você pode usar a dica chamada `Parallel` para forçar a execução paralela. Para obter mais informações, consulte Dicas para planos paralelos.

Diagnosticar o comportamento das consultas paralelas

Use `EXPLAIN (ANALYZE, VERBOSE)` para confirmar se uma consulta usou execução paralela:

- Procure nós como `Gather`, `Gather Merge` ou `Parallel Seq Scan`.
- Compare os planos com e sem paralelismo.

Para desabilitar temporariamente o paralelismo para comparação:

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
RESET max_parallel_workers_per_gather;
```

Solução de problemas de armazenamento no Aurora PostgreSQL

Se a quantidade de memória de trabalho necessária para operações de classificação ou criação de índice exceder a quantidade alocada pelo parâmetro `work_mem`, o Aurora PostgreSQL gravará o excesso de dados em arquivos de disco temporários. Quando ele grava os dados, o Aurora PostgreSQL usa o mesmo espaço de armazenamento que é utilizado para logs de erro e mensagem, ou seja, armazenamento local. Cada instância em seu cluster de banco de dados do Aurora PostgreSQL tem uma quantidade de armazenamento local disponível. A quantidade de armazenamento é baseada em sua classe de instância de banco de dados. Para aumentar a quantidade de armazenamento local, você precisa modificar a instância para usar uma classe de instância de banco de dados maior. Para conhecer as especificações de classes de instância de banco de dados, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

Você pode monitorar o espaço de armazenamento local de seu cluster de banco de dados do Aurora PostgreSQL observando a métrica do Amazon CloudWatch para `FreeLocalStorage`. Essa métrica informa a quantidade de armazenamento disponível para cada instância no cluster de banco de dados do Aurora para tabelas temporárias e logs. Para obter mais informações, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.

As operações de classificação, indexação e agrupamento começam na memória de trabalho, mas geralmente precisam ser transferidas para o armazenamento local. Se seu cluster de banco de dados do Aurora PostgreSQL ficar sem armazenamento local devido a esses tipos de operação, você poderá resolver o problema realizando uma das ações a seguir.

- Aumente a quantidade de memória de trabalho. Isso reduz a necessidade de usar o armazenamento local. Por padrão, o PostgreSQL aloca 4 MB para cada operação de classificação, grupo e índice. Para conferir o valor da memória de trabalho atual para a instância do gravador do

cluster de banco de dados do Aurora PostgreSQL, conecte-se à instância usando `psql` e execute o comando a seguir.

```
postgres=> SHOW work_mem;
work_mem
----------
 4MB
(1 row)
```

Você pode aumentar a memória de trabalho no nível da sessão antes de classificar, agrupar e outras operações da forma a seguir.

```
SET work_mem TO '1 GB';
```

Para obter mais informações sobre a memória de trabalho, consulte Resource Consumption (Consumo de recursos) na documentação do PostgreSQL.

- Altere o período de retenção de logs para que os logs sejam armazenados por períodos mais curtos. Para saber como, consulte Arquivos de log do banco de dados Aurora PostgreSQL.

Se o cluster de banco de dados do Aurora PostgreSQL for maior do que 40 TB, não use classes de instância db.t2, db.t3 ou db.t4g. Recomendamos usar as classes de instância de banco de dados T somente para servidores de desenvolvimento e teste, ou outros servidores que não sejam de produção. Para obter mais informações, consulte Tipos de classe de instância de banco de dados.

Replicação com Amazon Aurora PostgreSQL

A seguir, você recebe informações sobre replicação com o Amazon Aurora PostgreSQL, que incluem como monitorar e usar a replicação lógica.

Tópicos

- Usar réplicas do Aurora
- Melhorar a disponibilidade de leitura das réplicas do Aurora
- Monitorar a replicação do Aurora PostgreSQL
- Visão geral da replicação lógica do PostgreSQL com o Aurora
- Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL
- Desativar a replicação lógica

- Monitorar o cache de gravação simultânea e os slots lógicos para a replicação lógica do Aurora PostgreSQL
- Exemplo: usar a replicação lógica com clusters de banco de dados do Aurora PostgreSQL
- Exemplo: replicação lógica usando o Aurora PostgreSQL e o AWS Database Migration Service
- Configurar a autenticação do IAM para conexões de replicação lógica

Usar réplicas do Aurora

Uma Aurora réplica é um endpoint independente em um cluster de banco de dados do Aurora, cuja melhor utilidade é dimensionar operações de leitura e aumentar a disponibilidade. Um cluster de banco de dados do Aurora pode incluir até réplicas de 15 Aurora localizadas nas zonas de disponibilidade da região da AWS do cluster de banco de dados do Aurora.

O volume do cluster de banco de dados é composto por várias cópias dos dados do cluster de banco de dados. Contudo, os dados no volume do cluster são representados como um volume lógico, único, para a instância de banco de dados do gravador principal e para réplicas do Aurora no cluster de banco de dados. Para mais informações sobre réplicas do Aurora, consulte Réplicas do Aurora.

As réplicas do Aurora funcionam bem para a escalabilidade de leitura porque são totalmente dedicadas a operações de leitura em seu volume de cluster. A instância de banco de dados do gravador grava operações. O volume do cluster é compartilhado com todas as instâncias em seu cluster de banco de dados Aurora PostgreSQL. Assim, nenhum trabalho extra é necessário para replicar uma cópia dos dados para cada réplica do Aurora.

Com o Aurora PostgreSQL, quando a réplica do Aurora é excluída, o endpoint da instância é removido imediatamente, e a réplica do Aurora é removida do endpoint leitor. Se houver instruções em execução na réplica do Aurora que está sendo excluída, haverá um período de carência de três minutos. As instruções existentes podem ser concluídas durante o período de carência. Quando o período de carência termina, a réplica do Aurora é fechada e excluída.

Os clusters de banco de dados do Aurora PostgreSQL são compatíveis com réplicas do Aurora em diferentes regiões da AWS, usando o banco de dados global do Aurora. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.

> **Note**
>
> Com o atributo de disponibilidade de leitura, se você quiser reinicializar as réplicas do Aurora no cluster de banco de dados, terá que fazê-lo manualmente. Para os clusters de banco

de dados criados antes desse recurso, a reinicialização da instância de banco de dados do gravador reinicializa automaticamente as réplicas do Aurora. A reinicialização automática restabelece um ponto de entrada que garante a consistência de leitura/gravação no cluster de banco de dados.

Melhorar a disponibilidade de leitura das réplicas do Aurora

O Aurora PostgreSQL melhora a disponibilidade de leitura no cluster de banco de dados atendendo continuamente às solicitações de leitura quando a instância de banco de dados do gravador é reiniciada ou quando a réplica do Aurora não consegue acompanhar o tráfego de gravação.

O recurso de disponibilidade de leitura está disponível por padrão nas seguintes versões do Aurora PostgreSQL:

- 16.1 e todas as versões posteriores
- Versão 15.2 e versões 15 posteriores
- Versão 14.7 e versões 14 posteriores
- Versão 13.10 e versões 13 posteriores
- Versão 12.14 e versões 12 posteriores

O recurso de disponibilidade de leitura é compatível com as seguintes versões do banco de dados do Aurora:

- 16.1 e todas as versões posteriores
- 15.4 e versões 15 posteriores
- 14.9 e versões 14 posteriores
- 13.12 e versões 13 posteriores
- 12.16 e versões 12 posteriores

Para usar o recurso de disponibilidade de leitura para um cluster de banco de dados criado em uma dessas versões antes desse lançamento, reinicie a instância do gravador do cluster de banco de dados.

Ao modificar parâmetros estáticos do cluster de banco de dados do Aurora PostgreSQL, é necessário reiniciar a instância do gravador para que as alterações entrem em vigor. Por exemplo,

é necessário reiniciar a instância do gravador ao definir o valor de `shared_buffers`. Com o recurso de disponibilidade de leitura das réplicas do Aurora, o cluster de banco de dados mantém a disponibilidade aprimorada, o que reduz o impacto nelas quando a instância do gravador é reiniciada. As instâncias do leitor não são reiniciadas e continuam respondendo às solicitações de leitura. Para aplicar alterações de parâmetros estáticos, reinicialize cada instância individual do leitor.

A réplica do Aurora de um cluster de banco de dados do Aurora PostgreSQL pode se recuperar de erros de replicação, como reinicializações do gravador, failover, replicação lenta e problemas de rede, recuperando-se rapidamente para o estado do banco de dados na memória após a reconexão com o gravador. Essa abordagem permite que as instâncias de réplica do Aurora alcancem a consistência com as atualizações de armazenamento mais recentes enquanto o banco de dados do cliente ainda está disponível.

As transações em andamento que entram em conflito com a recuperação da replicação podem receber um erro, mas o cliente pode repetir essas transações depois que os leitores acompanharem o gravador.

Monitorar réplicas do Aurora

É possível monitorar as réplicas do Aurora ao se recuperar de uma desconexão do gravador. Use as métricas abaixo para verificar as informações mais recentes sobre a instância do leitor e rastrear transações em andamento somente leitura.

- A função `aurora_replica_status` é atualizada para retornar as informações mais atualizadas para a instância do leitor quando ela ainda está conectada. O carimbo de data/hora da última atualização no `aurora_replica_status` está sempre vazio para a linha correspondente à instância de banco de dados na qual a consulta é executada. Isso indica que a instância do leitor tem os dados mais recentes.
- Quando a réplica do Aurora se desconecta da instância do gravador e se reconecta, o seguinte evento do banco de dados é emitido:

  ```
  Read replica has been disconnected from the writer instance and reconnected.
  ```
- Quando uma consulta somente leitura é cancelada devido a um conflito de recuperação, você pode ver uma ou mais das seguintes mensagens de erro no log de erros do banco de dados:

  ```
  Canceling statement due to conflict with recovery.
  ```

  ```
  User query may not have access to page data to replica disconnect.
  ```

```
User query might have tried to access a file that no longer exists.

When the replica reconnects, you will be able to repeat your command.
```

Limitações

As seguintes limitações aplicam-se às réplicas do Aurora com o recurso de disponibilidade de leitura:

- As réplicas do Aurora do cluster de banco de dados secundário poderão ser reiniciadas se os dados não puderem ser transmitidos da instância de gravador durante a recuperação da replicação.
- As réplicas do Aurora não serão compatíveis com a recuperação de replicação on-line se uma já estiver em andamento e for reiniciada.
- As réplicas do Aurora serão reiniciadas quando sua instância de banco de dados estiver próxima da conclusão do ID da transação. Para obter mais informações sobre a conclusão de ID de transação, consulte Evitar falhas de conclusão de ID de transação.
- As réplicas do Aurora podem ser reiniciadas quando o processo de replicação é bloqueado em determinadas circunstâncias.

Monitorar a replicação do Aurora PostgreSQL

A escalabilidade de leitura e a alta disponibilidade dependem de um tempo de atraso mínimo. Você pode monitorar até que ponto uma réplica do Aurora está atrasada em relação à instância de banco de dados do gravador do cluster de banco de dados do Aurora PostgreSQL monitorando a métrica `ReplicaLag` do Amazon CloudWatch. Como as réplicas do Aurora leem no mesmo volume do cluster que a instância de banco de dados do gravador, a métrica `ReplicaLag` tem um significado diferente para um cluster de banco de dados do Aurora PostgreSQL. A métrica `ReplicaLag` de uma réplica do Aurora indica o atraso do cache da página da réplica do Aurora em comparação com o da instância de banco de dados do gravador.

Para obter mais informações sobre como monitorar as instâncias do RDS e as métricas do CloudWatch, consulte Monitorar métricas em um cluster do Amazon Aurora.

Visão geral da replicação lógica do PostgreSQL com o Aurora

Ao usar o recurso de replicação lógica do PostgreSQL com seu cluster de banco de dados do Aurora PostgreSQL, você pode replicar e sincronizar tabelas individuais em vez de toda a instância do

banco de dados. A replicação lógica usa um modelo de publicação e de assinatura para replicar as alterações de uma fonte para um ou mais destinatários. Ela funciona usando registros de alterações do log de gravação antecipada (WAL) do PostgreSQL. A fonte, ou o editor, envia os dados WAL das tabelas especificadas a um ou mais destinatários (assinante), replicando, dessa forma, as alterações e mantendo a tabela do assinante sincronizada com a tabela do editor. O conjunto de alterações do editor é identificado por meio de uma publicação. Os assinantes recebem as alterações criando uma assinatura que define a conexão com o banco de dados do editor e suas publicações. Um slot de replicação é o mecanismo utilizado nesse esquema para monitorar o andamento de uma assinatura.

Para clusters de banco de dados do Aurora PostgreSQL, os registros WAL são salvos no armazenamento do Aurora. O cluster de banco de dados do Aurora PostgreSQL que atua como editor em um cenário de replicação lógica lê os dados WAL do armazenamento do Aurora, os decodifica e os envia ao assinante para que as alterações possam ser aplicadas à tabela nessa instância. O editor usa um decodificador lógico para decodificar os dados a serem utilizados pelos assinantes. Por padrão, os clusters de banco de dados do Aurora PostgreSQL usam o plug-in `pgoutput` nativo do PostgreSQL ao enviar dados. Outros decodificadores lógicos estão disponíveis. Por exemplo, o Aurora PostgreSQL também é compatível com o plug-in `wal2json` que converte dados WAL em JSON.

A partir das versões 14.5, 13.8, 12.12 e 11.17 do Aurora PostgreSQL, o Aurora PostgreSQL incrementa o processo de replicação lógica do PostgreSQL com um cache de gravação para melhorar a performance. Os logs de transações do WAL são armazenados em cache localmente, em um buffer, para reduzir a quantidade de E/S do disco, ou seja, a leitura do armazenamento do Aurora durante a decodificação lógica. O cache de gravação é usado por padrão sempre que você usa a replicação lógica para o cluster de banco de dados do Aurora PostgreSQL. O Aurora fornece várias funções que você pode usar para gerenciar o cache. Para obter mais informações, consulte Monitorar o cache de gravação simultânea de replicação lógica do Aurora PostgreSQL.

A replicação lógica é compatível com todas as versões do Aurora PostgreSQL atualmente disponíveis. Para obter informações, Amazon Aurora PostgreSQL updates (Atualizações do Amazon Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

A replicação lógica é compatível com o Babelfish para Aurora PostgreSQL a partir das seguintes versões:

- 15.7 e versões posteriores
- 16.3 e versões posteriores

> **Note**
>
> Além do recurso nativo de replicação lógica do PostgreSQL introduzido no PostgreSQL 10, o Aurora PostgreSQL também é compatível com a extensão `pglogical`. Para obter mais informações, consulte Usar pglogical para sincronizar dados entre instâncias.

Para obter informações detalhadas sobre a replicação lógica do PostgreSQL, consulte Logical replication (Replicação lógica) e Logical decoding concepts (Conceitos da decodificação lógica) na documentação do PostgreSQL.

> **Note**
>
> O PostgreSQL 16 adicionou suporte para decodificação lógica por meio de réplicas de leitura. Esse recurso não é aceito no Aurora PostgreSQL.

Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL

A configuração da replicação lógica exige privilégios `rds_superuser`. Seu cluster de banco de dados do Aurora PostgreSQL deve estar configurado para usar um grupo de parâmetros de cluster de banco de dados personalizado para que você possa definir os parâmetros necessários conforme detalhado no procedimento a seguir. Para obter mais informações, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

Como configurar a replicação lógica PostgreSQL para seu cluster de banco de dados do Aurora PostgreSQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione seu cluster de banco de dados do Aurora PostgreSQL.
3. Abra a guia Configuration (Configuração). Entre os detalhes da instância, encontre o link do Grupo de parâmetros com Grupo de parâmetros do cluster de banco de dados para Tipo.
4. Clique no link para abrir os parâmetros personalizados associados ao seu cluster de banco de dados do Aurora PostgreSQL.

5. No campo Parameters (Parâmetros), digite `rds` para encontrar o parâmetro `rds.logical_replication`. O valor padrão para esse parâmetro é `0`, o que significa que ele está desativado por padrão.
6. Selecione Edit parameters (Editar parâmetros) para acessar os valores das propriedades e depois selecione 1 no seletor para ativar o recurso. Dependendo do uso esperado, talvez você também precise alterar as configurações dos parâmetros a seguir. No entanto, em muitos casos, os valores padrão são suficientes.

   - `max_replication_slots`: defina esse parâmetro com um valor que seja pelo menos igual ao número total planejado de publicações e assinaturas de replicação lógica. Se você estiver usando AWS DMS, esse parâmetro deverá se igualar pelo menos às suas tarefas planejadas de captura de dados de alteração do cluster, além de publicações e assinaturas de replicação lógica.
   - `max_wal_senders` e `max_logical_replication_workers`: defina esses parâmetros como um valor pelo menos equivalente ao número de slots de replicação lógica que você planeja manter ativos ou ao número de tarefas AWS DMS ativas para a captura de dados de alterações. Deixar um slot de replicação lógica inativo evita que o vácuo remova tuplas obsoletas das tabelas. Por isso, recomendamos monitorar os slots de replicação e remover os inativos conforme necessário.
   - `max_worker_processes`: defina esse parâmetro com um valor que seja pelo menos igual ao total dos valores `max_logical_replication_workers`, `autovacuum_max_workers` e `max_parallel_workers`. Em classes de instância de banco de dados pequenas, os processos de operador em segundo plano podem afetar as workloads de aplicações. Por isso, monitore a performance do banco de dados se você definir `max_worker_processes` como maior do que o valor padrão. (O valor padrão é o resultado de `GREATEST(${DBInstanceVCPU*2},8}`, o que significa que, por padrão, isso é oito ou duas vezes o equivalente de CPU da classe da instância de banco de dados, o que for maior).

   > **Note**
   >
   > Você pode modificar valores de parâmetros em um grupo de parâmetros de banco de dados criado pelo cliente, mas não pode alterar os valores dos parâmetros em um grupo de parâmetros de banco de dados padrão.

7. Escolha Salvar alterações.

8. Reinicialize a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para que as alterações tenham efeito. No console do Amazon RDS, selecione a instância de banco de dados primária do cluster e selecione Reboot (Reinicializar) no menu Actions (Ações).
9. Quando a instância estiver disponível, você poderá verificar se a replicação lógica está ativada da forma a seguir.

    a. Use o `psql` para se conectar à instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

    ```
    psql --host=your-db-cluster-instance-1.aws-region.rds.amazonaws.com --port=5432
     --username=postgres --password --dbname=labdb
    ```

    b. Verifique se a replicação lógica foi habilitada usando o comando a seguir.

    ```
    labdb=> SHOW rds.logical_replication;
     rds.logical_replication
    -------------------------
     on
    (1 row)
    ```

    c. Verifique se o `wal_level` está definido como `logical`.

    ```
    labdb=> SHOW wal_level;
      wal_level
    -----------
     logical
    (1 row)
    ```

Para ver um exemplo de como usar a replicação lógica para manter uma tabela de banco de dados sincronizada com as alterações de um cluster de banco de dados do Aurora PostgreSQL de origem, consulte Exemplo: usar a replicação lógica com clusters de banco de dados do Aurora PostgreSQL.

Desativar a replicação lógica

Depois de concluir suas tarefas de replicação, você deve interromper o processo de replicação, eliminar os slots de replicação e desativar a replicação lógica. Antes de descartar os slots, garanta que eles não sejam mais necessários. Os slots de replicação ativos não podem ser descartados.

Como desativar a replicação lógica

1. Descarte todos os slots de replicação.

   Para descartar todos os slots de replicação, conecte-se ao editor e execute o comando SQL a seguir

   ```
   SELECT pg_drop_replication_slot(slot_name)
     FROM pg_replication_slots
    WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
   ```

   Os slots de replicação não poderão estar ativos quando você executar esse comando.
2. Modifique o grupo de parâmetros de cluster de banco de dados personalizado que está associado ao editor, conforme detalhado em Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL, mas defina o parâmetro `rds.logical_replication` como 0.

   Para obter mais informações sobre grupos de parâmetros personalizados, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.
3. Reinicie o cluster de banco de dados do Aurora PostgreSQL do editor para que a alteração no `rds.logical_replication` tenha efeito.

Monitorar o cache de gravação simultânea e os slots lógicos para a replicação lógica do Aurora PostgreSQL

Monitore o cache de gravação simultânea de replicação lógica e gerencie slots lógicos para melhorar a performance do cluster de banco de dados do Aurora PostgreSQL. A seguir, encontre mais informações sobre o cache de gravação simultânea e os slots lógicos.

Tópicos

- Monitorar o cache de gravação simultânea de replicação lógica do Aurora PostgreSQL
- Gerenciar slots lógicos para o Aurora PostgreSQL

Monitorar o cache de gravação simultânea de replicação lógica do Aurora PostgreSQL

Por padrão, as versões 14.5, 13.8, 12.12 e 11.17 e posteriores do Aurora PostgreSQL usam um cache de gravação para melhorar a performance da replicação lógica. Sem o cache de gravação,

o Aurora PostgreSQL usa a camada de armazenamento do Aurora em sua implementação do processo nativo de replicação lógica do PostgreSQL. Ele faz isso gravando dados WAL no armazenamento e, depois, lendo os dados do armazenamento para decodificá-los e enviá-los (replicar) para seus destinos (assinantes). Isso pode causar gargalos durante a replicação lógica para clusters de banco de dados do Aurora PostgreSQL.

O cache de gravação simultânea minimiza a dependência da camada de armazenamento do Aurora. Em vez de gravar e ler dessa camada de maneira consistente, o Aurora PostgreSQL usa um buffer para armazenar em cache o fluxo lógico do WAL para uso durante o processo de replicação, reduzindo a necessidade de acessar o disco. Esse buffer é o cache nativo do PostgreSQL usado na replicação lógica e é identificado nos parâmetros do cluster de banco de dados do Aurora PostgreSQL como `rds.logical_wal_cache`.

Ao usar a replicação lógica com seu cluster de banco de dados do Aurora PostgreSQL (para as versões que são compatíveis com o cache de gravação), você pode monitorar a taxa de acerto do cache para ver se ela está funcionando bem em seu caso de uso. Para fazer isso, conecte-se à instância de gravação do cluster de banco de dados do Aurora PostgreSQL usando `psql` e, depois, use a função do Aurora, `aurora_stat_logical_wal_cache`, conforme mostrado no exemplo a seguir.

```
SELECT * FROM aurora_stat_logical_wal_cache();
```

A função retorna o resultado a seguir.

```
name       | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
-----------+------------+-----------+------------+-----------+----------+--------------
test_slot1 | 79183      | 24        | 0          | 24        | 100.00%  | 2022-08-05
 17:39...
test_slot2 |            | 1         | 0          |  1        | 100.00%  | 2022-08-05
 17:34...
(2 rows)
```

Os valores `last_reset_timestamp` foram reduzidos para facilitar a leitura. Para ter mais informações sobre essa função, consulte aurora_stat_logical_wal_cache.

O Aurora PostgreSQL fornece as duas funções a seguir para monitorar o cache de gravação.

- A função `aurora_stat_logical_wal_cache`: para obter a documentação de referência, consulte aurora_stat_logical_wal_cache.

- A função `aurora_stat_reset_wal_cache`: para obter a documentação de referência, consulte aurora_stat_reset_wal_cache.

Se você achar que o tamanho do cache do WAL ajustado automaticamente não é suficiente para as workloads, altere o valor de `rds.logical_wal_cache` manualmente. Considere o seguinte:

- Quando o parâmetro `rds.logical_replication` está desabilitado, `rds.logical_wal_cache` é definido como zero (0).
- Quando o parâmetro `rds.logical_replication` está habilitado, `rds.logical_wal_cache` tem um valor padrão de 16 MB.
- O parâmetro `rds.logical_wal_cache` é estático e requer a reinicialização da instância de banco de dados para que as alterações sejam aplicadas. Esse parâmetro é definido em termos de blocos de 8 Kb. Observe que qualquer valor positivo menor que 32 Kb é tratado como 32 Kb. Para ter mais informações sobre `wal_buffers`, consulte Write Ahead Log na documentação do PostgreSQL.

Gerenciar slots lógicos para o Aurora PostgreSQL

A atividade de streaming é capturada na visualização `pg_replication_origin_status`. Para ver o conteúdo dessa visualização, use a função `pg_show_replication_origin_status()`, conforme mostrado a seguir:

```
SELECT * FROM pg_show_replication_origin_status();
```

Você pode obter uma lista dos seus slots lógicos usando a consulta SQL a seguir.

```
SELECT * FROM pg_replication_slots;
```

Para descartar um slot lógico, use o `pg_drop_replication_slot` com o nome do slot, conforme mostrado no comando a seguir.

```
SELECT pg_drop_replication_slot('test_slot');
```

Exemplo: usar a replicação lógica com clusters de banco de dados do Aurora PostgreSQL

O procedimento a seguir mostra como iniciar a replicação lógica entre dois clusters de banco de dados do Aurora PostgreSQL. Tanto o editor quanto o assinante devem ser configurados para replicação lógica, conforme detalhado em Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL.

O cluster de banco de dados do Aurora PostgreSQL que é o editor designado também deve permitir o acesso ao slot de replicação. Para fazer isso, modifique o grupo de segurança associado à nuvem pública virtual (VPC) do cluster de banco de dados do Aurora PostgreSQL com base no serviço da Amazon VPC. Permita o acesso de entrada adicionando o grupo de segurança associado à VPC do assinante ao grupo de segurança do editor. Para obter mais informações, consulte Controle o tráfego para recursos usando grupos de segurança no Guia do usuário da Amazon VPC.

Com essas etapas preliminares concluídas, você pode usar os comandos do PostgreSQL `CREATE PUBLICATION` no editor e `CREATE SUBSCRIPTION` no assinante, conforme detalhado no procedimento a seguir.

Como iniciar o processo de replicação lógica entre dois clusters de banco de dados do Aurora PostgreSQL.

Essas etapas pressupõem que seus clusters de banco de dados do Aurora PostgreSQL tenham uma instância do gravador com um banco de dados para criar as tabelas de exemplo.

1. No cluster de banco de dados do Aurora PostgreSQL do editor

   a. Crie uma tabela usando a declaração SQL a seguir.

   ```
   CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
   ```

   b. Insira dados no banco de dados publisher usando a seguinte instrução SQL.

   ```
   INSERT INTO LogicalReplicationTest VALUES (generate_series(1,10000));
   ```

   c. Verifique se os dados existem na tabela usando a declaração SQL a seguir.

   ```
   SELECT count(*) FROM LogicalReplicationTest;
   ```

d. Crie uma publicação para essa tabela usando a declaração `CREATE PUBLICATION`, conforme explicado a seguir.

```
CREATE PUBLICATION testpub FOR TABLE LogicalReplicationTest;
```

2. No cluster de banco de dados do Aurora PostgreSQL do assinante

a. Crie a mesma tabela `LogicalReplicationTest` no assinante que você criou no editor, da forma a seguir.

```
CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
```

b. Verifique se essa tabela está vazia.

```
SELECT count(*) FROM LogicalReplicationTest;
```

c. Crie uma assinatura para receber as alterações do editor. Você precisa usar os detalhes a seguir sobre o cluster de banco de dados do Aurora PostgreSQL do editor.

- host: a instância de banco de dados do gravador do cluster de banco de dados do Aurora PostgreSQL do editor.
- port: a porta na qual a instância de banco de dados do gravador está escutando. O padrão do PostgreSQL é 5432.
- dbname: o nome do banco de dados.

```
CREATE SUBSCRIPTION testsub CONNECTION
   'host=publisher-cluster-writer-endpoint port=5432 dbname=db-name user=user
 password=password'
   PUBLICATION testpub;
```

> **Note**
>
> Especifique uma senha diferente do prompt mostrado aqui como prática recomendada de segurança.

Depois da criação da assinatura, um slot da replicação lógica é criado no publisher.

d. Para verificar neste exemplo se os dados iniciais são replicados no assinante, use a seguinte instrução SQL no banco de dados assinante.

```
SELECT count(*) FROM LogicalReplicationTest;
```

Todas as alterações adicionais no publisher são replicadas para o assinante.

A replicação lógica afeta a performance. Recomendamos que você desative a replicação lógica após a conclusão das tarefas de replicação.

Exemplo: replicação lógica usando o Aurora PostgreSQL e o AWS Database Migration Service

É possível usar o AWS Database Migration Service (AWS DMS) para replicar um banco de dados como uma parte de um banco de dados. Use o AWS DMS para migrar os dados de um banco de dados do Aurora PostgreSQL para outro banco de dados de código aberto ou comercial. Para obter mais informações sobre o AWS DMS, consulte o Manual do usuário do AWS Database Migration Service.

O exemplo a seguir mostra como configurar a replicação lógica de um banco de dados do Aurora PostgreSQL como publisher e usar o AWS DMS para migração. Este exemplo usa os mesmos publisher e assinante criados em Exemplo: usar a replicação lógica com clusters de banco de dados do Aurora PostgreSQL.

Para configurar a replicação lógica com o AWS DMS, são necessários os detalhes do publisher e do assinante no Amazon RDS. Especificamente, você precisa dos detalhes da instância de banco de dados do gravador do publisher e da instância de banco de dados do assinante.

Obtenha as seguintes informações da instância de banco de dados do gravador do publisher:

- O identificador da nuvem privada virtual (VPC)
- O grupo de sub-redes
- A zona de disponibilidade (AZ)
- O grupo de segurança da VPC
- O ID da instância de banco de dados

Obtenha as seguintes informações da instância de banco de dados do assinante:

- O ID da instância de banco de dados
- O mecanismo de origem

Para usar o AWS DMS para a replicação lógica com o Aurora PostgreSQL

1. Prepare o banco de dados publisher para trabalhar com o AWS DMS.

   Para isso, os bancos de dados PostgreSQL 10.x e posteriores exigem a aplicação de funções de wrapper do AWS DMS no banco de dados publisher. Para obter detalhes sobre isso e as etapas posteriores, consulte as instruções em Using PostgreSQL version 10.x and later as a source for AWS DMS (Usar PostgreSQL versão 10.x e posterior como origem para o DMS) no Manual do usuário do AWS Database Migration Service.
2. Faça login no Console de gerenciamento da AWS e abra o console do AWS DMS em https://console.aws.amazon.com/dms/v2. Na parte superior direita, escolha a mesma região da AWS em que o publisher e o assinante estão localizados.
3. Crie uma instância de replicação do AWS DMS.

   Escolha valores que sejam iguais aos da instância de banco de dados do gravador do publisher. Esses valores incluem as seguintes configurações:

   - Para VPC, escolha a mesma VPC da instância de banco de dados do gravador.
   - Para Replication Subnet Group (Grupo de sub-redes de replicação), escolha o mesmo grupo de sub-redes da instância de banco de dados de gravador. Crie um novo, se necessário.
   - Para Availability zone (Zona de disponibilidade), escolha a mesma zona da instância de banco de dados do gravador.
   - Para VPC Security Group (Grupo de segurança da VPC), escolha o mesmo grupo da instância de banco de dados do gravador.
4. Crie um endpoint do AWS DMS para a origem.

   Especifique o publisher como o endpoint de origem usando as seguintes configurações:

   - Em Endpoint Type (Tipo de endpoint), escolha Source endpoint (Endpoint de origem).
   - Escolha Select RDS DB Instance (Selecionar instância de banco de dados RDS).
   - Em RDS Instance (Instância do RDS), escolha o identificador da instância de banco de dados do gravador do publisher.
   - Em Source engine (Mecanismo de origem), escolha postgres.

5. Crie um endpoint do AWS DMS para o destino.

   Especifique o assinante como o endpoint de destino usando as seguintes configurações:

   - Em Endpoint Type (Tipo de endpoint), selecione Target endpoint (Endpoint de destino).
   - Escolha Select RDS DB Instance (Selecionar instância de banco de dados RDS).
   - Em RDS Instance (Instância do RDS), escolha o identificador da instância de banco de dados do gravador do assinante.
   - Escolha um valor para Source engine (Mecanismo de origem). Por exemplo, se o assinante for um banco de dados RDS para PostgreSQL, escolha postgres. Se o assinante for um banco de dados Aurora PostgreSQL, escolha aurora-postgresql.

6. Crie uma tarefa de migração de banco de dados do AWS DMS.

   Você usa uma tarefa de migração de banco de dados para especificar as tabelas a serem migradas, mapear dados usando um esquema de destino e criar tabelas novas no banco de dados de destino. No mínimo, use as seguintes configurações para Task configuration (Configuração da tarefa):

   - Em Replication instance (Instância de replicação), escolha a instância de replicação criada em uma etapa anterior.
   - Em Source database endpoint (Endpoint do banco de dados de origem), escolha a origem do publisher criada em uma etapa anterior.
   - Em Target database endpoint (Endpoint do banco de dados de destino), escolha o destino do assinante criado em uma etapa anterior.

   O restante dos detalhes da tarefa dependem de seu projeto de migração. Para obter mais informações sobre como especificar todos os detalhes para tarefas do DMS, consulte o tópico sobre como Trabalhar com tarefas do AWS DMS, no Manual do usuário do AWS Database Migration Service.

Depois que o AWS DMS cria a tarefa, ele começa a migrar os dados do publisher para o assinante.

Configurar a autenticação do IAM para conexões de replicação lógica

A partir das versões 11 e posterior do Aurora PostgreSQL, você pode usar a autenticação do AWS Identity and Access Management (IAM) para conexões de replicação. Esse recurso aprimora a segurança ao permitir que você gerencie o acesso ao banco de dados usando perfis do IAM em vez

de senhas. Ele funciona em nível de cluster e segue o mesmo modelo de segurança da autenticação padrão do IAM.

A autenticação do IAM para conexões de replicação é um atributo opcional. Para habilitá-lo, defina o parâmetro `rds.iam_auth_for_replication` como `1` no seu grupo de parâmetros do cluster de banco de dados. Como esse é um parâmetro dinâmico, seu cluster de banco de dados não precisa ser reiniciado, permitindo que você aproveite a autenticação do IAM com workloads existentes sem tempo de inatividade. Antes de habilitar esse atributo, você deve atender aos Pré-requisitos listados abaixo.

Pré-requisitos

Para usar a autenticação do IAM para conexões de replicação, você precisa cumprir todos os requisitos abaixo:

- Seu cluster de banco de dados do Aurora PostgreSQL deve ser da versão 11 ou posterior.
- No cluster de banco de dados do Aurora PostgreSQL do publicador:
  - Habilite a autenticação do banco de dados do IAM.

    Para obter mais informações, consulte Habilitar e desabilitar a autenticação de banco de dados do IAM.
  - Habilite a replicação lógica definindo o parâmetro `rds.logical_replication` como `1`.

    Para obter mais informações, consulte Configurar a replicação lógica para seu cluster de banco de dados do Aurora PostgreSQL.

  Na replicação lógica, o publicador é o cluster de banco de dados do Aurora PostgreSQL de origem que envia dados aos clusters de assinantes. Para acessar mais informações, consulte Visão geral da replicação lógica do PostgreSQL com o Aurora.

> **Note**
>
> Tanto a autenticação do IAM quanto a replicação lógica devem estar habilitadas no cluster de banco de dados do Aurora PostgreSQL do publicador. Se nenhuma delas estiver habilitada, você não poderá usar a autenticação do IAM para conexões de replicação.

Habilitar a autenticação do IAM para conexões de replicação

Conclua as etapas a seguir para habilitar a autenticação do IAM para a conexão de replicação.

1. Verifique se seu cluster de banco de dados do Aurora PostgreSQL atende a todos os pré-requisitos para autenticação do IAM com conexões de replicação. Para obter detalhes, consulte Pré-requisitos.
2. Configure o parâmetro `rds.iam_auth_for_replication` modificando o grupo de parâmetros do cluster de banco de dados:
   - Defina o parâmetro `rds.iam_auth_for_replication` como `1`. Esse parâmetro dinâmico não exige reinicialização.
3. Conecte-se ao seu banco de dados e conceda os perfis necessários ao seu usuário de replicação:

   Os comandos SQL a seguir concedem os perfis necessários para habilitar a autenticação do IAM para conexões de replicação:

```
-- Grant IAM authentication role
GRANT rds_iam TO replication_user_name;
-- Grant replication privileges
ALTER USER replication_user_name WITH REPLICATION;
```

Depois de concluir essas etapas, o usuário especificado deve usar a autenticação do IAM para conexões de replicação.

> **Important**
>
> Quando você habilita o recurso, os usuários com os perfis `rds_iam` e `rds_replication` devem usar a autenticação do IAM para conexões de replicação. Isso se aplicará se os perfis forem atribuídos diretamente ao usuário ou herdados por meio de outros perfis.

Desabilitar a autenticação do IAM para conexões de replicação

Você pode desabilitar a autenticação do IAM para conexões de replicação usando qualquer um dos seguintes métodos:

- Defina o parâmetro `rds.iam_auth_for_replication` para `0` no grupo de parâmetros do cluster do banco de dados.
- Você também pode desabilitar qualquer um desses recursos em seu cluster de banco de dados do Aurora PostgreSQL:
  - Desabilitar a replicação lógica definindo o parâmetro `rds.logical_replication` como `0`
  - Desabilitar a autenticação do IAM

Quando você desabilita o recurso, as conexões de replicação podem usar senhas de banco de dados para autenticação, se configuradas.

> **Note**
>
> As conexões de replicação para usuários sem o perfil `rds_iam` podem usar a autenticação por senha mesmo quando o recurso está habilitado.

Limitações e considerações

As limitações e as considerações se aplicam quando você usa a autenticação do IAM para conexões de replicação.

- A autenticação do IAM para conexões de replicação está disponível somente para as versões 11 e posterior do Aurora PostgreSQL.
- O publicador deve oferecer suporte à autenticação do IAM para conexões de replicação.
- Por padrão, o token de autenticação do IAM expira após 15 minutos. Talvez seja necessário atualizar as conexões de replicação de longa duração antes que o token expire.

Encaminhamento de gravação local no Aurora PostgreSQL

O encaminhamento de gravação local (no cluster) permite que as aplicações emitam transações de leitura/gravação diretamente em uma réplica do Aurora. Os comandos de gravação são então encaminhadas à instância de banco de dados do gravador para serem confirmados. É possível usar o encaminhamento de gravação local para as aplicações que têm gravações ocasionais e exigem consistência de leitura após gravação, que é a capacidade de ler a gravação mais recente em uma transação.

Sem o encaminhamento de gravação, as aplicações devem dividir todo o tráfego de leitura e de gravação, mantendo dois conjuntos de conexões a bancos de dados para enviar o tráfego ao endpoint apropriado. As réplicas de leitura recebem atualizações de maneira assíncrona da instância do gravador. Além disso, como o atraso na replicação pode diferir entre as réplicas de leitura, é difícil ter consistência de leitura global em todas as réplicas. É necessário realizar todas as leituras que exigem consistência de leitura após gravação na instância de banco de dados do gravador. Também é necessário desenvolver uma lógica de aplicação personalizada complexa para aproveitar as vantagens de várias réplicas de leitura para escalabilidade e, ao mesmo tempo, garantir a consistência.

Com o encaminhamento de gravação, é possível evitar a necessidade de dividir essas transações ou de enviá-las exclusivamente à instância do gravador. Além disso, não é necessário desenvolver uma lógica complexa de aplicações para ter consistência de leitura após gravação.

O encaminhamento de gravação local está disponível em todas as regiões que oferecem o Aurora PostgreSQL. O recurso é compatível com as seguintes versões do Aurora PostgreSQL:

- 16.4 e versões 16 posteriores
- 15.8 e versões 15 posteriores
- 14.13 e versões 14 posteriores

O encaminhamento de gravação local é usado para encaminhar gravações de réplicas na região. Para encaminhar gravações de uma réplica global, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Tópicos

- Limitações e considerações referentes ao encaminhamento de gravação local no Aurora PostgreSQL
- Configurar o Aurora PostgreSQL para encaminhamento de gravação local
- Trabalhar com encaminhamento de gravação local para o Aurora PostgreSQL
- Monitorar o encaminhamento de gravação local no Aurora PostgreSQL

Limitações e considerações referentes ao encaminhamento de gravação local no Aurora PostgreSQL

No momento, as seguintes limitações se aplicam ao encaminhamento de gravação local no Aurora PostgreSQL:

- O encaminhamento de gravação local não é compatível com o RDS Proxy.
- Determinadas instruções não são permitidas ou podem gerar resultados obsoletos ao serem usadas no Aurora PostgreSQL com encaminhamento de gravação. Além disso, funções definidas pelo usuário e procedimentos definidos pelo usuário não são compatíveis. Assim, a configuração `EnableLocalWriteForwarding` é desativada por padrão para clusters de banco de dados. Antes de ativá-la, verifique se o código do aplicativo não é afetado por nenhuma dessas restrições.
- Os seguintes tipos de instruções SQL não são compatíveis com o encaminhamento de gravação:

  > **Note**
  >
  > Essas instruções podem ser usadas implicitamente por você na aplicação ou inferidas pelo protocolo do PostgreSQL. Por exemplo, o tratamento de exceções PL/SQL pode ocasionar o uso de SAVEPOINT, que não é uma instrução aceita.

  - `ANALYZE`
  - `CLUSTER`
  - `COPY`
  - Cursores: cursores não são compatíveis, então feche-os antes de usar o encaminhamento de gravação local.
  - Instruções Data Definition Language (DDL)
  - `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
  - `LISTEN / NOTIFY`
  - `LOCK`
  - `SAVEPOINT`
  - `SELECT INTO`
  - `SET CONSTRAINTS`
  - Atualizações de sequência: `nextval()`, `setval()`

- TRUNCATE
- Comandos de confirmação em duas fases: PREPARE TRANSACTION, COMMIT PREPARED, ROLLBACK PREPARED
- Funções definidas pelo usuário e procedimentos definidos pelo usuário.
- VACUUM

É possível considerar o uso das seguintes instruções SQL com o encaminhamento de gravação:

- Uma declaração DML pode consistir em várias partes, como uma declaração INSERT ... SELECT ou DELETE ... WHERE. Nesse caso, a declaração inteira é encaminhada para a instância de banco de dados do gravador e é executada nela.
- Instruções de linguagem de manipulação de dados (DML), como INSERT, DELETE e UPDATE.
- Instruções EXPLAIN com as instruções desta lista
- Instruções PREPARE e EXECUTE.
- Instruções SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }.

Configurar o Aurora PostgreSQL para encaminhamento de gravação local

Com as seções a seguir, é possível habilitar o encaminhamento de gravação local para o cluster de banco de dados do Amazon Aurora PostgreSQL, configurando níveis de consistência e gerenciando transações com encaminhamento de gravação.

Habilitar o encaminhamento de gravação local

Por padrão, o encaminhamento de gravação local não está habilitado para clusters de banco de dados do Aurora PostgreSQL. Você habilita o encaminhamento de gravação local no cluster, não na instância.

Console

Usando o Console de gerenciamento da AWS, marque a caixa de seleção Ativar o encaminhamento de gravação local abaixo de Encaminhamento de gravação de réplica de leitura quando você criar ou modificar um cluster de banco de dados.

AWS CLI

Para habilitar o encaminhamento de gravação local com a AWS CLI, use a opção `--enable-local-write-forwarding`. Essa opção funciona quando você cria um cluster de banco de dados usando o comando `create-db-cluster`. Ela também funciona quando você modifica um cluster de banco de dados usando o comando `modify-db-cluster`. É possível desabilitar o encaminhamento de gravação local usando a opção `--no-enable-local-write-forwarding` com esses mesmos comandos da CLI.

O exemplo a seguir cria um cluster de banco de dados do Aurora PostgreSQL com encaminhamento de gravação local habilitado.

```
aws rds create-db-cluster \
--db-cluster-identifier write-forwarding-test-cluster \
--enable-local-write-forwarding \
--engine aurora-postgresql \
--engine-version 16.4 \
--master-username myuser \
--master-user-password mypassword \
--backup-retention 1
```

Depois, você cria instâncias de banco de dados do gravador e do leitor para poder usar o encaminhamento de gravação. Para ter mais informações, consulte Criar um cluster de bancos de dados do Amazon Aurora.

API do RDS

Para habilitar o encaminhamento de gravação local usando a API do Amazon RDS, defina o parâmetro `EnableLocalWriteForwarding` como `true`. Esse parâmetro funciona quando você cria um cluster de banco de dados usando a operação `CreateDBCluster`. Ele também funciona quando você modifica um cluster de banco de dados usando a operação `ModifyDBCluster`. É possível desabilitar o encaminhamento de gravação local definindo o parâmetro `EnableLocalWriteForwarding` como `false`.

Habilitar o encaminhamento de gravação local para sessões de banco de dados

`apg_write_forward.consistency_mode` é um parâmetro de banco de dados e de cluster de banco de dados que permite o encaminhamento de gravação. É possível especificar `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF` para o nível de consistência de leitura. Para saber mais sobre os níveis

de consistência, consulte Isolamento e consistência para o encaminhamento de gravação local no Aurora PostgreSQL.

As seguintes regras se aplicam a esse parâmetro:

- O valor padrão é `SESSION`.
- O encaminhamento de gravação local estará disponível apenas se você definir `apg_write_forward.consistency_mode` como `EVENTUAL`, `SESSION` ou `GLOBAL`. Esse parâmetro é relevante somente em instâncias do leitor de clusters de banco de dados que têm o encaminhamento de gravação local habilitado.
- Definir o valor como `OFF` desabilita o encaminhamento de gravação local na sessão.

Isolamento e consistência para o encaminhamento de gravação local no Aurora PostgreSQL

É possível controlar o grau de consistência de leitura em uma réplica de leitura. Você pode ajustar o nível de consistência de leitura para garantir que todas as operações de gravação encaminhadas da sessão estejam visíveis na réplica de leitura antes de qualquer consulta subsequente. Também é possível usar essa configuração para garantir que as consultas na réplica de leitura sempre vejam as atualizações mais recentes da instância de banco de dados do gravador. Isso acontece mesmo para aquelas submetidas por outras sessões ou outros clusters. Para especificar esse tipo de comportamento para a aplicação, escolha o valor apropriado para o parâmetro `apg_write_forward.consistency_mode` de nível de sessão. O parâmetro `apg_write_forward.consistency_mode` tem efeito somente em réplicas de leitura que tenham o encaminhamento de gravação local habilitado.

> **Note**
>
> Para o parâmetro `apg_write_forward.consistency_mode`, é possível especificar o valor `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Por padrão, o valor é definido como `SESSION`. Definir o valor como `OFF` desabilita o encaminhamento de gravação.

À medida que você aumenta o nível de consistência, a aplicação passa mais tempo aguardando as alterações serem propagadas para as réplicas de leitura. É possível escolher o equilíbrio entre latência mais baixa e garantia de que as alterações feitas em outros locais estejam totalmente disponíveis antes da execução das consultas.

Com cada configuração de modo de consistência disponível, o efeito é o seguinte:

- `SESSION`: uma sessão em uma réplica de leitura que usa o encaminhamento de gravação local vê os resultados de todas as alterações feitas nessa sessão. As alterações são visíveis independentemente de a transação ser confirmada. Se necessário, a consulta aguardará que os resultados das operações de gravação encaminhadas sejam replicados na instância de banco de dados do leitor atual. Ela não aguarda resultados atualizados de operações de gravação realizadas em outras sessões no cluster de banco de dados atual.
- `EVENTUAL`: uma sessão em uma réplica de leitura que usa o encaminhamento de gravação local pode ver dados ligeiramente obsoletos devido ao atraso de replicação. Os resultados das operações de gravação na mesma sessão ficam visíveis apenas quando a operação de gravação é realizada na instância de banco de dados do gravador e replicada na réplica de leitura. A consulta não espera que os resultados atualizados estejam disponíveis. Assim, ela pode recuperar os dados mais antigos ou os dados atualizados, dependendo do tempo das declarações e da quantidade de atraso da replicação.
- `GLOBAL`: uma sessão em uma réplica de leitura vê as alterações feitas por essa sessão. Ela também vê todas as alterações confirmadas da instância de banco de dados do gravador e das outras réplicas de leitura. Cada consulta pode aguardar por um período que varia de acordo com a quantidade de atraso da sessão. A consulta prosseguirá quando a réplica de leitura estiver atualizada com todos os dados confirmados da instância de banco de dados do gravador, a partir do momento em que a consulta foi iniciada.

  > **Note**
  >
  > O modo de consistência global prejudica a latência das consultas realizadas em uma sessão. Ele realizará uma espera mesmo quando a sessão não tiver enviado nenhuma consulta de gravação.

- `OFF`: o encaminhamento de gravação local está desabilitado.

Em sessões que usam o encaminhamento de gravação, é possível usar os níveis de isolamento `REPEATABLE READ` e `READ COMMITTED`. No entanto, o nível de isolamento `SERIALIZABLE` não é compatível.

Para obter mais informações sobre todos os parâmetros envolvidos no encaminhamento de gravação, consulte Configurações de parâmetros padrão para o encaminhamento de gravação.

Modos de acesso a transações com encaminhamento de gravação

Se o modo de acesso a transações estiver definido como somente leitura, o encaminhamento de gravação local não será usado. Você só poderá definir o modo de acesso como leitura e gravação enquanto estiver conectado a um cluster de banco de dados e a uma sessão que tenha o encaminhamento de gravação local habilitado.

Para ter mais informações sobre os modos de acesso a transações, consulte SET TRANSACTION.

Trabalhar com encaminhamento de gravação local para o Aurora PostgreSQL

Com as seções a seguir, é possível conferir se um cluster de banco de dados tem o encaminhamento de gravação local habilitado, ver as considerações de compatibilidade e consultar os parâmetros configuráveis e a configuração de autenticação. Essas informações fornecem os detalhes para utilizar o recurso de encaminhamento de gravação local no Aurora PostgreSQL de maneira eficaz.

> **Note**
>
> Quando uma instância do gravador em um cluster que use o encaminhamento de gravação local é reiniciada, todas as transações e consultas ativas e encaminhadas nas instâncias do leitor que usam o encaminhamento de gravação local são fechadas automaticamente. Quando a instância do gravador estiver novamente disponível, você poderá repetir essas transações.

Conferir se um cluster de banco de dados tem o encaminhamento de gravação local habilitado

Para determinar se você pode usar o encaminhamento de gravação local em um cluster de banco de dados, confirme se o cluster tem o atributo `LocalWriteForwardingStatus` definido como `enabled`.

No Console de gerenciamento da AWS, na guia Configuração da página de detalhes do cluster, você vê o status Habilitado para Encaminhamento de gravação de réplica de leitura local.

Para ver o status da configuração do encaminhamento de gravação local para todos os clusters, execute o comando da AWS CLI a seguir.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
{
"LocalWriteForwardingStatus": "enabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-1"
},
{
"LocalWriteForwardingStatus": "disabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-2"
},
{
"LocalWriteForwardingStatus": "requested",
"DBClusterIdentifier": "test-global-cluster-2"
},
{
"LocalWriteForwardingStatus": "null",
"DBClusterIdentifier": "aurora-postgresql-v2-cluster"
}
]
```

Um cluster de banco de dados pode ter os seguintes valores para `LocalWriteForwardingStatus`:

- `disabled`: o encaminhamento de gravação local está desabilitado.
- `disabling`: o encaminhamento de gravação local está sendo desabilitado.
- `enabled`: o encaminhamento de gravação local está habilitado.
- `enabling`: o encaminhamento de gravação local está sendo habilitado.
- `null`: o encaminhamento de gravação local não está disponível para esse cluster de banco de dados.
- `requested`: o encaminhamento de gravação local foi solicitado, mas ainda não está ativo.

Configurações de parâmetros padrão para o encaminhamento de gravação

Os grupos de parâmetros do cluster do Aurora incluem configurações para o recurso de encaminhamento de gravação local. Como são parâmetros de cluster, todas as instâncias de banco de dados em cada cluster têm os mesmos valores para essas variáveis. Detalhes sobre esses parâmetros são resumidos na tabela a seguir, com notas de uso após a tabela.

Parameter	Escopo	Type	Valor padrão	Valores válidos
`apg_write_forward.connect_timeout`	Sessão	segundos	30	0–2147483647
`apg_write_forward.consistency_mode`	Sessão	enum	Sessão	`SESSION`, `EVENTUAL`, `GLOBAL`, e `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Sessão	milissegundos	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sessão	milissegundos	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Global	int	25	1–100

O parâmetro `apg_write_forward.max_forwarding_connections_percent` é o limite máximo em conexões de banco de dados que pode ser usado em consultas encaminhadas de leitores. Ele é expresso como uma porcentagem da configuração `max_connections` para a instância de banco de dados do gravador. Por exemplo, se `max_connections` for `800` e `apg_write_forward.max_forwarding_connections_percent` for `10`, o gravador permitirá um máximo de 80 sessões encaminhadas simultaneamente. Essas conexões vêm do mesmo grupo de conexões gerenciado pela configuração `max_connections`. Essa configuração se aplica somente à instância de banco de dados do gravador quando o encaminhamento de gravação local está habilitado no cluster.

Use as seguintes configurações para controlar as solicitações de encaminhamento de gravação local:

- `apg_write_forward.consistency_mode`: um parâmetro ao nível da sessão que controla o grau de consistência de leitura em uma réplica de leitura. Os valores válidos são `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Por padrão, o valor é definido como `SESSION`. Definir o valor como `OFF` desabilita o encaminhamento de gravação local na sessão. Para saber mais sobre os níveis de consistência, consulte Isolamento e consistência para o encaminhamento de gravação local no Aurora PostgreSQL. Esse parâmetro é relevante somente em instâncias do leitor que tenham o encaminhamento de gravação local habilitado.
- `apg_write_forward.connect_timeout`: o número máximo de segundos que a réplica de leitura espera ao estabelecer uma conexão com a instância de banco de dados do gravador antes de desistir. Um valor de `0` significa esperar indefinidamente.
- `apg_write_forward.idle_in_transaction_session_timeout`: o número de milissegundos que a instância de banco de dados do gravador espera pela atividade em uma conexão encaminhada de uma réplica de leitura que tenha uma transação aberta antes de fechá-la. Se a sessão permanecer ociosa além desse período na transação, o Aurora encerrará a sessão. Um valor de `0` desabilita o tempo limite.
- `apg_write_forward.idle_session_timeout`: o número de milissegundos que a instância de banco de dados do gravador espera pela atividade em uma conexão encaminhada de uma réplica de leitura antes de fechá-la. Se a sessão permanecer ociosa além desse período, o Aurora encerrará a sessão. Um valor de `0` desativa o tempo limite.

rdswriteforwarduser

O `rdswriteforwarduser` é um usuário que vamos utilizar para estabelecer conexão entre a réplica de leitura e a instância de banco de dados do gravador.

> **Note**
>
> `rdswriteforwarduser` herda os privilégios CONNECT para bancos de dados de clientes por meio do perfil PUBLIC. Se os privilégios do perfil PUBLIC forem revogados, você precisará conceder privilégios CONNECT para os bancos de dados aos quais precisa encaminhar gravações.

Monitorar o encaminhamento de gravação local no Aurora PostgreSQL

Usando as seções a seguir, é possível monitorar o encaminhamento de gravação local nos clusters do Aurora PostgreSQL, incluindo métricas pertinentes do CloudWatch e eventos de espera com o objetivo de monitorar o desempenho e identificar possíveis problemas.

Métricas do Amazon CloudWatch e variáveis de status do Aurora PostgreSQL para encaminhamento de gravação

As métricas do Amazon CloudWatch a seguir se aplicam às instâncias de banco de dados do gravador quando o encaminhamento de gravação é utilizado em uma ou mais réplicas de leitura.

Métrica do CloudWatch	Unidades e descrição
`AuroraLocal ForwardingWriterDML Throughput`	Contagem (por segundo). Número de instruções DML encaminhadas processadas a cada segundo por essa instância de banco de dados de gravador.
`AuroraLocalForwardingWriter OpenSessions`	Contagem. Número de sessões abertas nessa instância de banco de dados de gravação que processa consultas encaminhadas.
`AuroraLocal ForwardingWriterTotalSessions`	Contagem. Número de sessões encaminhadas na instância de banco de dados de gravação.

As métricas do CloudWatch a seguir se aplicam a cada réplica de leitura. Essas métricas são medidas em cada instância de banco de dados do leitor em um cluster secundário com o encaminhamento de gravação local habilitado.

Métrica do CloudWatch	Unidade e descrição
`AuroraForwardingReplicaCommitThroughput`	Contagem (por segundo). Número de confirmações em sessões encaminhadas por essa réplica a cada segundo.

Métrica do CloudWatch	Unidade e descrição
`AuroraForwardingReplicaDMLLatency`	Milissegundos. Tempo médio de resposta, em milissegundos, de DMLs encaminhadas na réplica.
`AuroraForwardingReplicaDMLThroughput`	Contagem (por segundo). Número de instruções DML encaminhadas processadas por segundo nessa réplica.
`AuroraForwardingReplicaErrorSessionsLimit`	Contagem. Número de sessões rejeitadas pela instância de banco de dados do gravador por ter atingido o limite máximo de conexões ou o máximo de conexões de encaminhamento de gravação.
`AuroraForwardingReplicaOpenSessions`	Contagem. O número de sessões que estão usando o encaminhamento de gravação local em uma instância de réplica.
`AuroraForwardingReplicaReadWaitLatency`	Milissegundos. Tempo médio de espera, em milissegundos, que a réplica aguarda para ser consistente com o LSN da instância de banco de dados do gravador. O grau em que a instância de banco de dados de leitura aguarda depende da configuração `apg_write_forward.consistency_mode`. Para obter mais informações sobre essa configuração, consulte the section called “Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL”.

Eventos de espera para encaminhamento de gravação local no Aurora PostgreSQL

O Amazon Aurora gera os seguintes eventos de espera quando você usa o encaminhamento de gravação com o Aurora PostgreSQL.

Tópicos

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

O evento `IPC:AuroraWriteForwardConnect` ocorre quando um processo de backend na réplica de leitura está aguardando a abertura de uma conexão com a instância de banco de dados do gravador.

Possíveis causas do maior número de esperas

Esse evento aumenta à medida que aumenta o número de tentativas de conexão de uma réplica de leitura no nó do gravador.

Ações do

Reduza o número de conexões simultâneas de uma réplica de leitura no nó do gravador.

IPC:AuroraWriteForwardConsistencyPoint

O evento `IPC:AuroraWriteForwardConsistencyPoint` descreve por quanto tempo uma consulta de um nó na réplica de leitura aguardará até que os resultados das operações de gravação encaminhadas sejam replicados na região atual. Esse evento só será gerado se o parâmetro do nível da sessão `apg_write_forward.consistency_mode` estiver definido como um dos seguintes casos:

- `SESSION`: as consultas em uma réplica de leitura aguardam os resultados de todas as alterações feitas nessa sessão.
- `GLOBAL`: as consultas em uma réplica de leitura aguardam os resultados das alterações feitas por essa sessão, além de todas as alterações confirmadas da instância de banco de dados do gravador e da réplica de leitura.

Para obter informações sobre as configurações do parâmetro `apg_write_forward.consistency_mode`, consulte the section called “Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL”.

Possíveis causas do maior número de esperas

As causas comuns de tempos de espera mais longos incluem o seguinte:

- Aumento do atraso na réplica, conforme medido pela métrica do Amazon CloudWatch `ReplicaLag`. Para obter mais informações sobre essa métrica, consulte Monitorar a replicação do Aurora PostgreSQL.
- Maior carga na instância de banco de dados do gravador ou na réplica de leitura.

Ações do

Altere seu modo de consistência, dependendo dos requisitos da aplicação.

IPC:AuroraWriteForwardExecute

O evento `IPC:AuroraWriteForwardExecute` ocorre quando um processo de backend na réplica de leitura está aguardando a conclusão de uma consulta encaminhada e o recebimento dos resultados no nó do gravador do cluster de banco de dados.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- Buscar um grande número de linhas no nó do gravador.
- O aumento da latência da rede entre o nó do gravador e a réplica de leitura aumenta o tempo necessário para a réplica de leitura receber dados do nó do gravador.
- O aumento da carga na réplica de leitura pode atrasar a transmissão da solicitação de consulta da réplica de leitura para o nó do gravador.
- O aumento da carga no nó do gravador pode atrasar a transmissão dos dados do nó do gravador para a réplica de leitura.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Otimize as consultas para recuperar somente os dados necessários.

- Otimize as operações de linguagem de manipulação de dados (DML) para modificar somente os dados necessários.
- Se a réplica de leitura ou o nó do gravador estiver limitado pela CPU ou pela largura de banda da rede, pense em alterá-lo para um tipo de instância com maior capacidade de CPU ou maior largura de banda da rede.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

O evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` ocorre quando um processo de backend na réplica de leitura que está usando o modo de consistência GLOBAL está esperando para receber o ponto de consistência global do nó do gravador antes de realizar uma consulta.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre a réplica de leitura e o nó do gravador aumenta o tempo necessário para a réplica de leitura receber dados do nó do gravador.
- O aumento da carga na réplica de leitura pode atrasar a transmissão da solicitação de consulta da réplica de leitura para o nó do gravador.
- O aumento da carga no nó do gravador pode atrasar a transmissão dos dados do nó do gravador para a réplica de leitura.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Altere seu modo de consistência, dependendo dos requisitos da aplicação.
- Se a réplica de leitura ou o nó do gravador estiver limitado pela CPU ou pela largura de banda da rede, pense em alterá-lo para um tipo de instância com maior capacidade de CPU ou maior largura de banda da rede.

IPC:AuroraWriteForwardXactAbort

O evento `IPC:AuroraWriteForwardXactAbort` ocorre quando um processo de backend na réplica de leitura está aguardando o resultado de uma consulta de limpeza remota. As consultas

de limpeza são emitidas para retornar o processo ao estado apropriado após a interrupção de uma transação por gravação. O Amazon Aurora as executa porque um erro foi encontrado ou porque um usuário emitiu um comando `ABORT` explícito ou cancelou uma consulta em execução.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre a réplica de leitura e o nó do gravador aumenta o tempo necessário para a réplica de leitura receber dados do nó do gravador.
- O aumento da carga na réplica de leitura pode atrasar a transmissão da solicitação de consulta de limpeza da réplica de leitura para o nó do gravador.
- O aumento da carga no nó do gravador pode atrasar a transmissão dos dados do nó do gravador para a réplica de leitura.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Investigue a causa da transação abortada.
- Se a réplica de leitura ou a instância de banco de dados do gravador estiver limitada pela CPU ou pela largura de banda da rede, pense em alterá-la para um tipo de instância com maior capacidade de CPU ou maior largura de banda da rede.

IPC:AuroraWriteForwardXactCommit

O evento `IPC:AuroraWriteForwardXactCommit` ocorre quando um processo de backend na réplica de leitura está aguardando o resultado de um comando de confirmação de transação encaminhado.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre a réplica de leitura e o nó do gravador aumenta o tempo necessário para a réplica de leitura receber dados do nó do gravador.
- O aumento da carga na réplica de leitura pode atrasar a transmissão da solicitação de consulta da réplica de leitura para o nó do gravador.

- O aumento da carga no nó do gravador pode atrasar a transmissão dos dados do nó do gravador para a réplica de leitura.

Ações do

Se a réplica de leitura ou o nó do gravador estiver limitado pela CPU ou pela largura de banda da rede, pense em alterá-lo para um tipo de instância com maior capacidade de CPU ou maior largura de banda da rede.

IPC:AuroraWriteForwardXactStart

O evento `IPC:AuroraWriteForwardXactStart` ocorre quando um processo de backend na réplica de leitura está aguardando o resultado de um comando de início de transação encaminhado.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre a réplica de leitura e o nó do gravador aumenta o tempo necessário para a réplica de leitura receber dados do nó do gravador.
- O aumento da carga na réplica de leitura pode atrasar a transmissão da solicitação de consulta da réplica de leitura para o nó do gravador.
- O aumento da carga no nó do gravador pode atrasar a transmissão dos dados do nó do gravador para a réplica de leitura.

Ações do

Se a réplica de leitura ou o nó do gravador estiver limitado pela CPU ou pela largura de banda da rede, pense em alterá-lo para um tipo de instância com maior capacidade de CPU ou maior largura de banda da rede.

Usar o Aurora PostgreSQL como base de conhecimento para o Amazon Bedrock

É possível usar um cluster de banco de dados do Aurora PostgreSQL como Base de Conhecimento para Amazon Bedrock. Para obter mais informações, consulte Create a vector store in Amazon Aurora. Uma Base de Conhecimento extrai automaticamente os dados de texto não estruturados armazenados em um bucket do Amazon S3, converte-os em blocos e vetores de texto e armazena-

os em um banco de dados do PostgreSQL. Com as aplicações de IA generativa, é possível usar agentes do Amazon Bedrock para consultar os dados armazenados na base de conhecimento e usar os resultados dessas consultas para otimizar as respostas fornecidas pelos modelos de base. Esse fluxo de trabalho é chamado de Geração Aumentada de Recuperação (RAG). Para ter mais informações sobre a RAG, consulte Geração Aumentada de Recuperação (RAG).

Para ter informações detalhadas sobre como usar o Aurora PostgreSQL para criar aplicações de IA generativa usando RAG, consulte esta publicação de blog.

Tópicos

- Pré-requisitos
- Preparar o Aurora PostgreSQL para ser usado como base de conhecimento para o Amazon Bedrock
- Criação de uma Base de Conhecimento no console do Bedrock
- Criar rapidamente uma Base de Conhecimento para Amazon Bedrock do Aurora PostgreSQL

Pré-requisitos

Familiarize-se com os pré-requisitos a seguir para usar o cluster do Aurora PostgreSQL como base de conhecimento para o Amazon Bedrock. Em geral, é necessário configurar os seguintes serviços para uso com o Bedrock:

- Cluster de banco de dados do Amazon Aurora PostgreSQL criado em qualquer uma das seguintes versões:
  - 16.1 e todas as versões posteriores
  - 15.4 e versões posteriores
  - 14.9 e versões posteriores
  - 13.12 e versões posteriores
  - 12.16 e versões posteriores

> **Note**
>
> É necessário habilitar a extensão `pgvector` no banco de dados de destino e usar a versão 0.5.0 ou posterior. Para ter mais informações, consulte pgvector v0.5.0 com indexação HNSW.

- API de dados do RDS
- Um usuário gerenciado no AWS Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.

Preparar o Aurora PostgreSQL para ser usado como base de conhecimento para o Amazon Bedrock

Siga as etapas explicadas nas seções abaixo para preparar o Aurora PostgreSQL para ser usado como Base de Conhecimento para Amazon Bedrock.

Criar e configurar o Aurora PostgreSQL

Para configurar o Amazon Bedrock com um cluster de banco de dados do Aurora PostgreSQL, é necessário primeiro criar um cluster de banco de dados do Aurora PostgreSQL e anotar os campos importantes para configurá-lo com o Amazon Bedrock. Para ter mais informações sobre como criar um cluster de banco de dados do Aurora PostgreSQL, consulte Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele.

- Habilite a API de dados ao criar o cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações sobre as versões compatíveis, consulte Usar a API de dados do Amazon RDS.
- Anote os nomes dos recursos da Amazon (ARNs) do cluster de banco de dados do Aurora PostgreSQL. Você precisará deles para configurar o cluster de banco de dados para uso com o Amazon Bedrock. Para ter mais informações, consulte Nomes de recurso da Amazon (ARNs).

Conectar-se a um banco de dados e instalar o pgvector

É possível se conectar ao Aurora PostgreSQL usando qualquer um dos utilitários de conexão. Para ter informações mais detalhadas sobre esses utilitários, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora PostgreSQL. Como alternativa, é possível usar o editor de consultas do console do RDS para executar as consultas. Você precisa de um cluster de banco de dados do Aurora com a API de dados do RDS habilitada para usar o editor de consultas.

1. Faça login no banco de dados com o usuário principal e configure o pgvector. Use o comando a seguir se a extensão não estiver instalada:

```
CREATE EXTENSION IF NOT EXISTS vector;
```

Use a versão `pgvector` 0.5.0 e posterior que aceita a indexação HNSW. Para ter mais informações, consulte pgvector v0.5.0 com indexação HNSW.

2. Use o seguinte comando para conferir a versão do `pg_vector` instalado:

```
SELECT extversion FROM pg_extension WHERE extname='vector';
```

Configurar objetos e privilégios de banco de dados

1. Crie um esquema específico que o Bedrock possa usar para consultar os dados. Use o seguinte comando para criar um esquema:

```
CREATE SCHEMA bedrock_integration;
```

2. Crie um perfil que o Bedrock possa usar para consultar o banco de dados. Use o seguinte comando para criar um perfil:

```
CREATE ROLE bedrock_user WITH PASSWORD 'password' LOGIN;
```

> **Note**
>
> Anote essa senha, pois você a usará posteriormente para criar uma senha do Secrets Manager.

Se você estiver usando o cliente `psql`, utilize os seguintes comandos para criar um perfil:

```
CREATE ROLE bedrock_user LOGIN;
\PASSWORD password;
```

3. Conceda ao `bedrock_user` permissões para gerenciar o esquema `bedrock_integration`. Isso oferecerá a capacidade de criar tabelas ou índices no esquema.

```
GRANT ALL ON SCHEMA bedrock_integration to bedrock_user;
```

4. Faça login como `bedrock_user` e crie uma tabela no `bedrock_integration schema`.

```
CREATE TABLE bedrock_integration.bedrock_kb (id uuid PRIMARY KEY, embedding
 vector(n), chunks text, metadata json, custom_metadata jsonb);
```

Esse comando criará a tabela `bedrock_kb` no esquema `bedrock_integration` com incorporações do Titan.

Substitua n no tipo de dados `vector(n)` pela dimensão apropriada para o modelo de incorporação que você está usando. Use as recomendações abaixo para ajudar a selecionar suas dimensões:

- Para o modelo Titan v2, use `vector(1024)`, `vector(512)` ou `vector (256)`. Para saber mais, consulte Texto de incorporações do Amazon Titan.
- Para o modelo Titan v1.2, use `vector(1536)`. Para saber mais, consulte Amazon Titan Multimodal Embeddings G1.
- Para o modelo Cohere Embed, use `vector(1024)`. Para saber mais, consulte Modelos Cohere Embed.
- Para o Cohere Embed Multilingual v3, use `vector(1024)`.

As quatro primeiras colunas são obrigatórias. Para tratamento de metadados, o Bedrock grava dados de seus arquivos de metadados na coluna `custom_metadata`. Recomendamos criar essa coluna se você planeja usar metadados e filtragem. Se você não criar uma coluna `custom_metadata`, adicione colunas individuais para cada atributo de metadados em sua tabela antes de iniciar a ingestão. Para ter mais informações, consulte Configure and customize queries and response generation.

5. Siga estas etapas para criar os índices necessários que o Bedrock usa para consultar seus dados:

   - Crie um índice com um operador cosseno que o Bedrock possa usar para consultar os dados.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops);
     ```

   - Recomendamos definir o valor de `ef_construction` como 256 para a versão `pgvector` 0.6.0 e posterior que usam a criação paralela de índices.

```
CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
 vector_cosine_ops) WITH (ef_construction=256);
```

- Crie um índice que o Bedrock possa usar para consultar os dados de texto.

```
CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (to_tsvector('simple',
 chunks));
```

- Se você criou uma coluna para metadados personalizados, crie um índice que o Bedrock possa usar para consultar os metadados.

```
CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (custom_metadata);
```

Criar um segredo no Secrets Manager

O Secrets Manager permite que você armazene suas credenciais do Aurora para que elas possam ser transmitidas com segurança às aplicações. Se você não escolheu a opção do AWS Secrets Manager ao criar o cluster de banco de dados do Aurora PostgreSQL, poderá criar um segredo agora. Para ter mais informações sobre como criar um segredo de banco de dados AWS Secrets Manager, consulte AWS Secrets Manager database secret.

Criação de uma Base de Conhecimento no console do Bedrock

Ao preparar o Aurora PostgreSQL para ser usado como armazenamento de vetores para uma Base de Conhecimento, reúna os detalhes a seguir que devem ser fornecidos ao console do Amazon Bedrock.

- ARN do cluster de banco de dados do Amazon Aurora: o ARN do cluster de banco de dados.
- ARN secreto: o ARN da chave do AWS Secrets Manager para o cluster de banco de dados.
- Nome do banco de dados: o nome do banco de dados. Por exemplo, é possível usar o banco de dados padrão *`postgres`*.
- Nome da tabela: recomendamos que você forneça um nome qualificado para o esquema ao criar a tabela usando o comando semelhante ao seguinte:

```
CREATE TABLE bedrock_integration.bedrock_kb;
```

  Esse comando criará a tabela `bedrock_kb` no esquema `bedrock_integration`.

- Ao criar a tabela, configure-a com as colunas e os tipos de dados especificados. É possível usar os nomes de coluna de sua preferência em vez dos listados na tabela. Lembre-se de anotar os nomes que você escolheu para consulta durante a configuração da Base de Conhecimento.

Nome da coluna	Tipo de dados	Descrição
id	Chave primária de UUID	Contém identificadores exclusivos para cada registro.
blocos	Texto	Contém os fragmentos de texto bruto das fontes de dados.
incorporação	Vector	Contém as incorporações de vetores das fontes de dados.
metadata	JSON	Contém os metadados necessários para realizar a atribuição da fonte e permitir a ingestão e a consulta de dados.
custom_metadata	JSONB	(Opcional) Define a coluna de destino em que o Amazon Bedrock grava detalhes de metadados de suas fontes de dados.

Com esses detalhes, agora é possível criar uma Base de Conhecimento no console do Bedrock. Para ter informações mais detalhadas sobre a configuração de um índice vetorial e a criação de informações da Base de Conhecimento, consulte Create a vector store in Amazon Aurora e Create a vector store in Amazon Aurora.

Agora que você já adicionou o Aurora como Base de Conhecimento, é possível ingerir suas fontes de dados para pesquisar e consultar. Para ter mais informações, consulte Ingest your data sources into the Knowledge Base.

Criar rapidamente uma Base de Conhecimento para Amazon Bedrock do Aurora PostgreSQL

O fluxo de trabalho de geração aumentada via recuperação (RAG) do Amazon Bedrock depende de dados vetoriais armazenados em um banco de dados do Aurora PostgreSQL para potencializar a recuperação de conteúdo. Anteriormente, configurar o Aurora PostgreSQL como armazenamento de dados vetoriais para as bases de conhecimento do Bedrock era um processo de várias etapas, exigindo numerosas ações manuais em diferentes interfaces de usuário. Isso tornava difícil para cientistas de dados e desenvolvedores aproveitarem o Aurora em seus projetos do Bedrock.

Para melhorar a experiência do usuário, a AWS criou uma nova opção de criação rápida baseada no CloudFormation que simplifica o processo de configuração. Com a criação rápida do Aurora, agora é possível provisionar um cluster de banco de dados do Aurora PostgreSQL pré-configurado como armazenamento de vetores para suas bases de conhecimento do Amazon Bedrock com um único clique.

Tópicos

- Regiões e versões compatíveis do Aurora PostgreSQL
- Noções básicas sobre o processo de criação rápida
- Benefícios do uso da criação rápida do Aurora
- Limitações do processo de criação rápida do Aurora

Regiões e versões compatíveis do Aurora PostgreSQL

A opção de criação rápida do Aurora está disponível em todas as regiões da AWS que oferecem suporte às bases de conhecimento do Amazon Bedrock. Por padrão, ela cria um cluster de banco de dados do Aurora PostgreSQL com a versão 15.7. Para obter mais informações sobre as regiões compatíveis, consulte Supported models and regions for Amazon Bedrock Knowledge Bases.

Noções básicas sobre o processo de criação rápida

O processo de criação rápida provisiona automaticamente os seguintes recursos para configurar um banco de dados do Amazon Aurora PostgreSQL como armazenamento de dados vetoriais para sua base de conhecimento do Bedrock:

Um cluster de banco de dados do Aurora PostgreSQL em sua conta, configurado com configurações padrão.

- As unidades de capacidade do Aurora (ACUs) são definidas de 0 a 16. Isso permite que o armazenamento de vetores seja reduzido verticalmente a zero quando não estiver em uso, reduzindo os custos computacionais. As ACUs podem ser ajustadas posteriormente no console do Amazon RDS.
- Índice Hierarchical Navigable Small World (HNSW) usando a distância euclidiana como medida de similaridade para as incorporações de vetores do Bedrock armazenadas no Aurora.
- A instância de banco de dados é uma instância com tecnologia sem servidor v2.
- O cluster está associado à VPC e às sub-redes padrão e tem a API de dados do RDS habilitada.
- As credenciais de administrador do cluster são gerenciadas pelo AWS Secrets Manager.

Além das configurações padrão, as configurações a seguir são definidas para você. Ao passar pelo processo, você verá telas que explicam o fluxo de trabalho.

- Propagando o cluster do Aurora com os objetos de banco de dados necessários:
  - Crie a extensão pgvector, o esquema, a função e as tabelas necessárias para a base de conhecimento do Bedrock.
  - Registre um usuário de banco de dados com privilégios limitados para que o Bedrock interaja com o cluster.
- Um banner de progresso será exibido durante todo o processo de provisionamento de recursos, permitindo que você acompanhe o status dos seguintes subeventos:
  - Criação do cluster do Aurora
  - Propagação do cluster do Aurora
  - Criação da base de conhecimento

  O banner permanece visível até que a base de conhecimento seja totalmente criada, mesmo que você navegue para fora da página e retorne.
- Você pode clicar em `View details` no banner de progresso para ver o status de cada etapa. Para obter mais informações sobre eventos durante a criação da base de conhecimento, escolha o link do CloudFormation na tela de exibição de detalhes. Quando o processo estiver concluído, sua nova base de conhecimento do Bedrock estará pronta para uso.
- Os IDs de pilha de todos os recursos de criação rápida podem ser encontrados nas tags da base de conhecimento do Bedrock, caso você precise referenciá-los.

Uma base de conhecimento do Bedrock, com a configuração para o cluster do Aurora recém-provisionado como armazenamento de vetores, é criada.

Benefícios do uso da criação rápida do Aurora

- O processo de criação rápida baseado no CloudFormation reduz significativamente o tempo e a complexidade necessários para usar o Aurora como armazenamento de vetores.
- O Aurora oferece excelente desempenho, escalabilidade de vetores e benefícios de custo com a capacidade de escalar para zero custos computacionais quando não está em uso.
- O processo de criação rápida simplifica a experiência de ponta a ponta, permitindo que você crie e configure facilmente suas bases de conhecimento do Bedrock usando o Aurora.
- Os clientes podem criar com base no modelo do CloudFormation para personalizar o provisionamento com suas próprias configurações.

Limitações do processo de criação rápida do Aurora

- Com a opção de criação rápida do Aurora, o cluster de banco de dados é provisionado com configurações padrão. No entanto, essas configurações padrão podem não atender aos seus requisitos específicos ou ao caso de uso pretendido. A criação rápida não oferece opções para modificar as configurações durante o processo de provisionamento. As configurações são definidas automaticamente para agilizar a experiência de implantação. Se você precisar personalizar a configuração do cluster de banco de dados do Aurora, poderá fazer isso após a implantação inicial por criação rápida no console do Amazon RDS.
- Embora o fluxo de criação rápida simplifique o processo de configuração, o tempo para criar o cluster de banco de dados do Aurora ainda é de aproximadamente 10 minutos, o mesmo de uma implantação manual. Isso se deve ao tempo necessário para provisionar a infraestrutura do Aurora.
- A opção de criação rápida foi projetada para experimentação e configuração rápida. Os recursos criados por meio da criação rápida podem não ser adequados para uso em produção, e você não poderá migrá-los diretamente para um ambiente de produção na VPC.

Integração do Amazon Aurora PostgreSQL com outros produtos da AWS

O Amazon Aurora integra-se a outros produtos da AWS para que você possa estender seu cluster de banco de dados do Aurora PostgreSQL de forma a usar recursos adicionais na Nuvem AWS. Seu cluster de banco de dados do Aurora PostgreSQL pode usar produtos da AWS para fazer o seguinte:

- Coletar, visualizar e avaliar rapidamente a performance das instâncias de banco de dados do Aurora PostgreSQL com o Amazon RDS Performance Insights. O Performance Insights expande os recursos de monitoramento do Amazon RDS existentes para ilustrar a performance do banco de dados e ajudar a analisar todos os problemas que o afetam. Com o painel do Performance Insights, você pode visualizar a carga do banco de dados e filtrá-la por esperas, instruções SQL, hosts ou usuários. Para obter mais informações sobre o Performance Insights, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.
- É possível configurar seu cluster de banco de dados do Aurora PostgreSQL para publicar dados de log em um grupo no Amazon CloudWatch Logs. O CloudWatch Logs fornece armazenamento resiliente para seus registros de log. Com o CloudWatch Logs, você pode executar análise em tempo real de dados de log e usar o CloudWatch para criar alarmes e visualizar métricas. Para obter mais informações, consulte Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs.
- Importe dados de um bucket do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL ou exporte dados de um cluster de banco de dados do Aurora PostgreSQL para um bucket do Amazon S3. Para obter mais informações, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL e Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3.
- Adicione previsões baseadas em machine learning a aplicações de banco de dados utilizando a linguagem SQL. O machine learning do Aurora usa uma integração altamente otimizada entre o banco de dados Aurora e os produtos de machine learning (ML) da AWS SageMaker AI e Amazon Comprehend. Para obter mais informações, consulte Usar machine learning do Amazon Aurora com o Aurora PostgreSQL.
- Invoque funções do AWS Lambda em um cluster de banco de dados do Aurora PostgreSQL. Para isso, use a extensão `aws_lambda` PostgreSQL fornecida com Aurora PostgreSQL. Para obter mais informações, consulte Invocar uma função do AWS Lambda de um cluster de banco de dados do Aurora PostgreSQL.

- Integre consultas do Amazon Redshift e do Aurora PostgreSQL. Para obter mais informações, consulte Conceitos básicos do uso de consultas federadas no PostgreSQL no Guia do desenvolvedor de banco de dados do Amazon Redshift.

Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL

Você pode importar dados que foram armazenados usando o Amazon Simple Storage Service em uma tabela em uma instância de cluster de banco de dados do Aurora PostgreSQL. Para fazer isso, primeiro instale a extensão `aws_s3` do Aurora PostgreSQL . Essa extensão fornece as funções usadas para importar dados de um bucket do Amazon S3. Um bucket é um contêiner do Amazon S3 para objetos e arquivos. Os dados podem estar em um arquivo de valores separados por vírgula (CSV), em um arquivo de texto ou em um arquivo compactado (gzip). Veja a seguir como instalar a extensão e importar dados do Amazon S3 para uma tabela.

Seu banco de dados deve estar executando o PostgreSQL versão 10.7 ou superior para importar do Amazon S3 para o . Aurora PostgreSQL.

Se você não tiver dados armazenados no Amazon S3, crie um bucket e armazene os dados. Para ter mais informações, consulte os tópicos a seguir no Guia do usuário do Amazon Simple Storage Service.

- Criar um bucket
- Adicionar um objeto a um bucket

A importação entre contas do Amazon S3 é compatível. Para ter mais informações, consulte Conceder permissões entre contas no Guia do usuário do Amazon Simple Storage Service.

Você pode usar a chave gerenciada pelo cliente para criptografia ao importar dados do S3. Para ter mais informações, consulte Chaves do KMS armazenadas no AWS KMS no Guia do usuário do Amazon Simple Storage Service.

> **Note**
>
> A importação de dados do Amazon S3 não é compatível com o Aurora Serverless v1. É compatível com o Aurora Serverless v2.

Tópicos

- Instalar a extensão aws_s3
- Visão geral da importação de dados do Amazon S3
- Configurar o acesso a um bucket do Amazon S3
- Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL
- Referência de funções

Instalar a extensão aws_s3

Antes de usar o Amazon S3 com o seu cluster de banco de dados do Aurora PostgreSQL, você precisa instalar a extensão `aws_s3`. Essa extensão fornece as funções para importar dados do Amazon S3. Ela também fornece funções para exportar dados de uma instância de um cluster de banco de dados do Aurora PostgreSQL para um bucket do Amazon S3. Para ter mais informações, consulte Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3. A extensão `aws_s3` depende de algumas das funções auxiliares da extensão `aws_commons`, que é instalada automaticamente quando necessária.

Como instalar a extensão **`aws_s3`**

1. Use psql (ou pgAdmin) para se conectar à instância gravadora de seu cluster de banco de dados do Aurora PostgreSQL como um usuário que tem privilégios `rds_superuser`. Se você manteve o nome padrão durante o processo de configuração, se conectará como `postgres`.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Para instalar a extensão, execute o comando a seguir.

   ```
   postgres=> CREATE EXTENSION aws_s3 CASCADE;
   NOTICE: installing required extension "aws_commons"
   CREATE EXTENSION
   ```

3. Para verificar se a extensão está instalada, você pode usar o metacomando psql `\dx`.

   ```
   postgres=> \dx
          List of installed extensions
       Name     | Version |   Schema   |                 Description
   -------------+---------+------------+---------------------------------------------
   ```

```
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

As funções para importar dados do Amazon S3 e exportar dados para o Amazon S3 agora estão disponíveis para uso.

Visão geral da importação de dados do Amazon S3

Para importar dados do S3 para o do Aurora PostgreSQL

Primeiro, reúna os detalhes que você precisa fornecer à função. Isso inclui o nome da tabela na sua instância do cluster de banco de dados do Aurora PostgreSQL, e o nome do bucket, o caminho do arquivo, o tipo de arquivo e a Região da AWS em que os dados do Amazon S3 estão armazenados. Para ter mais informações, consulte Visualizar um objeto no Guia do usuário do Amazon Simple Storage Service.

> **Note**
>
> A importação de dados multipart do Amazon S3 não é compatível atualmente.

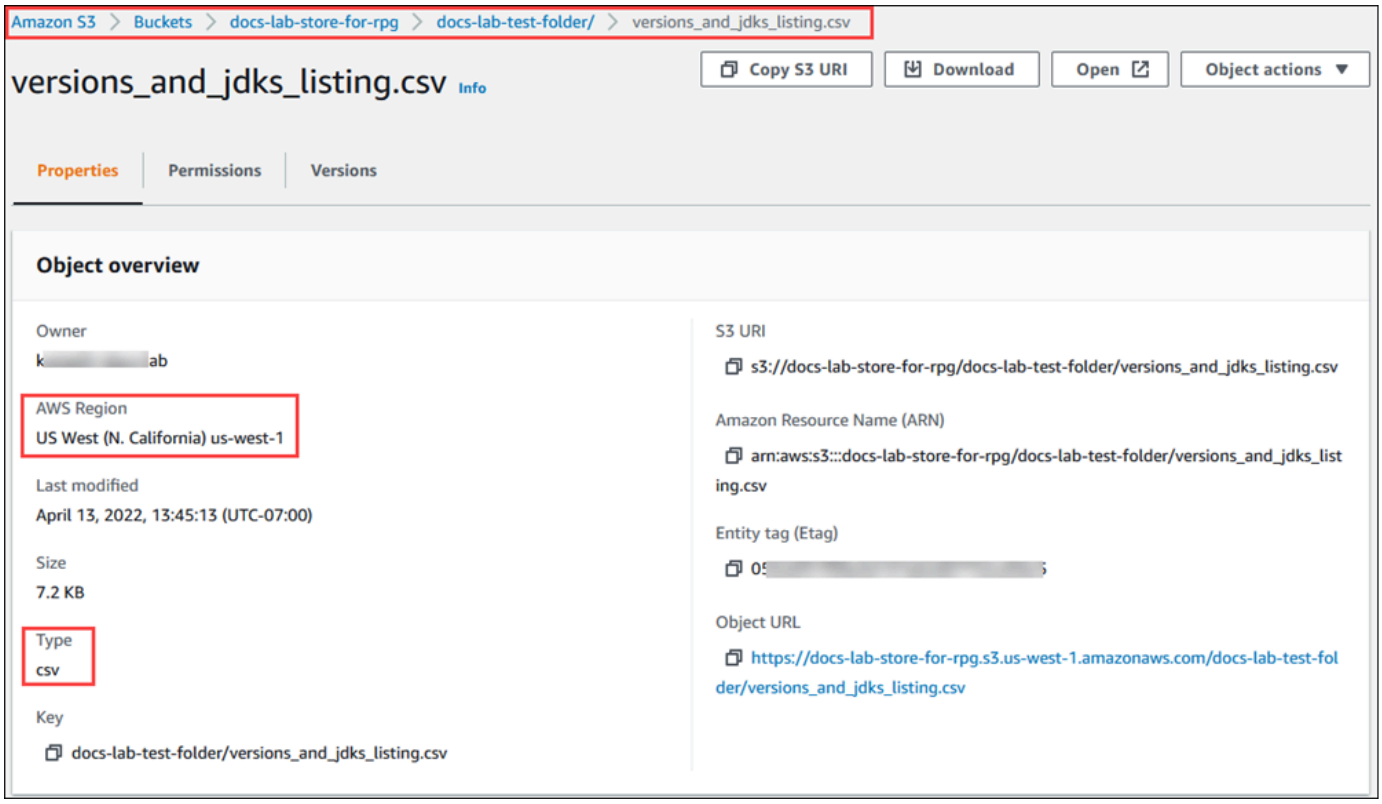
1. Obtenha o nome da tabela para a qual a função `aws_s3.table_import_from_s3` deverá importar os dados. Por exemplo, o comando a seguir cria uma tabela `t1` que pode ser utilizada em etapas posteriores.

   ```
   postgres=> CREATE TABLE t1
       (col1 varchar(80),
       col2 varchar(80),
       col3 varchar(80));
   ```

2. Veja os detalhes sobre o bucket do Amazon S3 e os dados a importar. Para fazer isso, abra o console do Amazon S3 em https://console.aws.amazon.com/s3/ e escolha Buckets. Encontre o bucket que contém seus dados na lista. Escolha o bucket, abra a página Object overview (Visão geral do objeto) e escolha Properties (Propriedades).

   Anote o nome do bucket, o caminho, a Região da AWS e o tipo de arquivo. Posteriormente, você precisará do nome do recurso da Amazon (ARN) para configurar o acesso ao Amazon S3 por

meio de um perfil do IAM. Para ter mais informações, consulte Configurar o acesso a um bucket do Amazon S3. A imagem a seguir mostra um exemplo.



3. Você pode verificar o caminho dos dados no bucket do Amazon S3 usando o comando `aws s3 cp` da AWS CLI. Se as informações estiverem corretas, esse comando baixará uma cópia do arquivo do Amazon S3.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Configure permissões em seu cluster de banco de dados do Aurora PostgreSQLpara permitir acesso ao arquivo no bucket do Amazon S3. Para fazer isso, use um perfil AWS Identity and Access Management (do IAM) ou credenciais de segurança. Para ter mais informações, consulte Configurar o acesso a um bucket do Amazon S3.

5. Forneça o caminho e outros detalhes do objeto do Amazon S3 coletados (consulte a etapa 2) para a função `create_s3_uri` a fim de construir um objeto URI do Amazon S3. Para saber mais sobre essa função, consulte aws_commons.create_s3_uri. Veja a seguir um exemplo de como construir esse objeto durante uma sessão psql.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
```

```
   'us-west-1'
) AS s3_uri \gset
```

Na próxima etapa, passe esse objeto (`aws_commons._s3_uri_1`) para a função `aws_s3.table_import_from_s3` a fim de importar os dados para a tabela.

6. Invoque a função `aws_s3.table_import_from_s3` para importar os dados do Amazon S3 para a sua tabela. Para obter informações de referência, consulte aws_s3.table_import_from_s3. Para obter exemplos, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL.

Configurar o acesso a um bucket do Amazon S3

Para importar dados de um arquivo do Amazon S3, conceda ao cluster de bancos de dados Aurora PostgreSQL permissão para acessar o bucket do Amazon S3 no qual o arquivo se encontra. Conceda acesso a um bucket do Amazon S3 de uma das duas maneiras, conforme descrito nos tópicos a seguir.

Tópicos

- Usar uma função do IAM para acessar um bucket do Amazon S3
- Usar credenciais de segurança para acessar um bucket do Amazon S3
- Solução de problemas de acesso ao Amazon S3

Usar uma função do IAM para acessar um bucket do Amazon S3

Antes de carregar dados de um arquivo do Amazon S3, conceda ao cluster de bancos de dados Aurora PostgreSQL permissão para acessar o bucket do Amazon S3 no qual o arquivo está. Dessa forma, não é necessário gerenciar informações adicionais de credenciais ou fornecê-las na chamada da função aws_s3.table_import_from_s3.

Para fazer isso, crie uma política do IAM que conceda acesso ao bucket do Amazon S3. Crie uma função do IAM e associe a política a ela. Depois, atribua uma função do IAM ao cluster de banco de dados.

> **Note**
>
> Não é possível associar uma função do IAM a um cluster de banco de dados Aurora Serverless v1, portanto, as etapas a seguir não se aplicam.

Para oferecer ao Amazon S3 acesso a uma do cluster de banco de dados do Aurora PostgreSQL por meio de uma função do IAM

1. Crie uma política do IAM.

   Essa política concede ao bucket e ao objeto as permissões para que o cluster de bancos de dados Aurora PostgreSQL acesse o Amazon S3.

   Inclua na política as seguintes ações necessárias para permitir a transferência de arquivos de um bucket do Amazon S3 para o do Aurora PostgreSQL:

   - `s3:GetObject`
   - `s3:ListBucket`

   Inclua na política os recursos a seguir para identificar o bucket e os objetos do Amazon S3 no bucket. Isso mostra o formato do nome de recurso da Amazon (ARN) para acessar o Amazon S3.

   - arn:aws:s3:::*amzn-s3-demo-bucket*
   - arn:aws:s3:::*amzn-s3-demo-bucket*/*

   Para ter mais informações sobre como criar uma política do IAM para o Aurora PostgreSQL, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM. Consulte também Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

   O comando da AWS CLI a seguir cria uma política do IAM denominada `rds-s3-import-policy` com essas opções. Ele concede acesso a um bucket denominado *amzn-s3-demo-bucket*.

   > **Note**
   >
   > Anote o nome do recurso da Amazon (ARN) da política que é retornado por esse comando. O ARN será necessário para uma etapa posterior, quando você anexar a política a um perfil do IAM.

Example

Para Linux, macOS ou Unix:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Para Windows:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
        ]
      }
    ]
  }'
```

2. Criar um perfil do IAM.

   Faça isso para que o Aurora PostgreSQL possa assumir essa função do IAM para acessar os buckets do Amazon S3. Para ter mais informações, consulte Criar um perfil para delegar permissões a um usuário do IAM no Guia do usuário do IAM.

   Convém usar as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas baseadas em recursos para limitar as permissões do serviço a um recurso específico. Essa é a maneira mais eficiente de se proteger contra o problema "confused deputy".

   Se você utilizar ambas as chaves de contexto de condição global, e o valor `aws:SourceArn` contiver o ID da conta, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão utilizar o mesmo ID de conta quando utilizados na mesma instrução de política.

   - Use `aws:SourceArn` se quiser acesso entre serviços para um único recurso.
   - Use `aws:SourceAccount` se você quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

   Na política, certifique-se de usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. O exemplo a seguir mostra como fazer isso utilizando o comando da AWS CLI para criar uma função chamada `rds-s3-import-role`.

   Example

   Para Linux, macOS ou Unix:

```
aws iam create-role \
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
```

```
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
              "StringEquals": {
                 "aws:SourceAccount": "111122223333",
                 "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                 }
              }
       }
     ]
   }'
```

Para Windows:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
   }'
```

3. Anexe a política do IAM que você criou ao perfil do IAM que você criou.

   O comando da AWS CLI a seguir vincula a política criada na etapa anterior à função `rds-s3-import-role`. Substitua *`your-policy-arn`* pelo ARN da política que você anotou naquela ocasião.

Example

Para Linux, macOS ou Unix:

```
aws iam attach-role-policy \
   --policy-arn your-policy-arn \
   --role-name rds-s3-import-role
```

Para Windows:

```
aws iam attach-role-policy ^
   --policy-arn your-policy-arn ^
   --role-name rds-s3-import-role
```

4. Adicione o perfil do IAM ao cluster de banco de dados.

   Faça isso usando o Console de gerenciamento da AWS ou a AWS CLI, conforme descrito a seguir.

Console

Para adicionar um perfil do IAM a um cluster de banco de dados do PostgreSQL usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha o nome da de cluster de banco de dados do PostgreSQL para exibir os detalhes.
3. Na guia Connectivity & security (Conectividade e segurança), na seção Manage IAM roles (Gerenciar funções do IAM), escolha a função a ser adicionada sob Add IAM roles to this cluster (Adicionar funções do IAM a este cluster/esta instância).
4. Em Feature (Recurso), escolha s3Import.
5. Escolha Add role (adicionar função).

AWS CLI

Para adicionar um perfil do IAM a um cluster de banco de dados do PostgreSQL usando a CLI

- Use o comando a seguir para adicionar a função ao cluster de banco de dados do PostgreSQL chamado `my-db-cluster`. Substitua *`your-role-arn`* pelo ARN da função que você anotou em uma etapa anterior. Use `s3Import` para o valor da opção `--feature-name`.

  Example

  Para Linux, macOS ou Unix:

  ```
  aws rds add-role-to-db-cluster \
     --db-cluster-identifier my-db-cluster \
     --feature-name s3Import \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Para Windows:

  ```
  aws rds add-role-to-db-cluster ^
     --db-cluster-identifier my-db-cluster ^
     --feature-name s3Import ^
     --role-arn your-role-arn ^
     --region your-region
  ```

API do RDS

Para adicionar um perfil do IAM a um cluster de banco de dados do PostgreSQL utilizando a API do Amazon RDS, chame a operação AddRoleToDBCluster.

Usar credenciais de segurança para acessar um bucket do Amazon S3

Se preferir, você poderá usar credenciais de segurança para conceder acesso a um bucket do Amazon S3 em vez de conceder acesso com uma função do IAM. Faça isso especificando o parâmetro `credentials` na chamada da função aws_s3.table_import_from_s3.

O parâmetro `credentials` é uma estrutura do tipo `aws_commons._aws_credentials_1`, que contém credenciais da AWS. Use a função aws_commons.create_aws_credentials para definir a

chave de acesso e a chave secreta em uma estrutura `aws_commons._aws_credentials_1`, conforme mostrado a seguir.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

Depois de criar a estrutura `aws_commons._aws_credentials_1` , use a função aws_s3.table_import_from_s3 com o parâmetro `credentials` para importar os dados, conforme mostrado a seguir.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

Outra opção é incluir a chamada de função aws_commons.create_aws_credentials em linha dentro da chamada de função `aws_s3.table_import_from_s3`.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Solução de problemas de acesso ao Amazon S3

Se você encontrar problemas de conexão ao tentar importar dados do Amazon S3, consulte o seguinte para obter recomendações:

- Solução de problemas de identidade e acesso do Amazon Aurora
- Solução de problemas do Amazon S3 no Guia do usuário do Amazon Simple Storage Service
- Troubleshooting Amazon S3 and IAM (Solucionar problemas no Amazon S3 e no IAM) no Manual do usuário do IAM

Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL

Importe dados do bucket do Amazon S3 usando a função `table_import_from_s3` da extensão aws_s3. Para obter informações de referência, consulte aws_s3.table_import_from_s3.

> **Note**
>
> Os exemplos a seguir usam o método de perfil do IAM para conceder acesso ao bucket do Amazon S3. Assim, as chamadas de função do `aws_s3.table_import_from_s3` não incluem parâmetros de credenciais.

Veja a seguir um exemplo típico.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

Os parâmetros são os seguintes:

- `t1` – o nome da tabela no cluster de banco de dados do PostgreSQL na qual os dados devem ser copiados.
- `''` – uma lista opcional de colunas na tabela de banco de dados. É possível usar esse parâmetro para indicar quais colunas dos dados do S3 devem ser inseridos em quais colunas da tabela. Se nenhuma coluna for especificada, todas as colunas serão copiadas para a tabela. Para obter um exemplo de uso de uma lista de colunas, consulte Importar um arquivo do Amazon S3 que usa um delimitador personalizado.
- `(format csv)` – argumentos COPY do PostgreSQL. O processo de cópia usa os argumentos e o formato do comando COPY PostgreSQL para importar os dados. As opções de formato incluem valores separados por vírgula (CSV), conforme mostrado neste exemplo, texto e binário. O padrão é texto.
- `s3_uri` – uma estrutura que contém as informações que identificam o arquivo do Amazon S3. Para obter um exemplo de como usar a função aws_commons.create_s3_uri para criar uma estrutura `s3_uri`, consulte Visão geral da importação de dados do Amazon S3.

Para ter mais informações sobre essa função, consulte aws_s3.table_import_from_s3.

A função `aws_s3.table_import_from_s3` retorna texto. Para especificar outros tipos de arquivos para importação de um bucket do Amazon S3, veja um dos exemplos a seguir.

> **Note**
>
> Importar 0 byte causará um erro.

Tópicos

- Importar um arquivo do Amazon S3 que usa um delimitador personalizado
- Importar um arquivo do Amazon S3 compactado (gzip)
- Importar um arquivo do Amazon S3 codificado

Importar um arquivo do Amazon S3 que usa um delimitador personalizado

O exemplo a seguir mostra como importar um arquivo que usa um delimitador personalizado. Ele também mostra como controlar onde colocar os dados na tabela de banco de dados usando o parâmetro `column_list` da função aws_s3.table_import_from_s3.

Para este exemplo, suponha que as informações a seguir estejam organizadas em colunas delimitadas por barras verticais no arquivo do Amazon S3.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

Como importar um arquivo que usa um delimitador personalizado

1. Crie uma tabela no banco de dados para os dados importados.

   ```
   postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
   ```

2. Use o seguinte formato da função aws_s3.table_import_from_s3 para importar dados do arquivo do Amazon S3.

Você pode incluir a chamada de função aws_commons.create_s3_uri em linha dentro da chamada de função `aws_s3.table_import_from_s3` para especificar o arquivo.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   'test',
   'a,b,d,e',
   'DELIMITER ''|''',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
east-2')
);
```

Os dados estão agora na tabela nas colunas a seguir.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Importar um arquivo do Amazon S3 compactado (gzip)

O exemplo a seguir mostra como importar um arquivo do Amazon S3 compactado com gzip. O arquivo que você importa precisa ter os seguintes metadados do Simple Storage Service (Amazon S3):

- Chave: `Content-Encoding`
- Valor:: `gzip`

Se você carregar o arquivo usando o Console de gerenciamento da AWS, os metadados geralmente são aplicados pelo sistema. Para obter informações sobre o carregamento de arquivos para o Simple Storage Service (Amazon S3) usando o Console de gerenciamento da AWS, a AWS CLI ou a API, consulte Carregar objetos no Guia do usuário do Amazon Simple Storage Service.

Para ter mais informações sobre metadados do Simple Storage Service (Amazon S3) e detalhes sobre metadados fornecidos pelo sistema, consulte Editar metadados de objeto no console do Simple Storage Service (Amazon S3) no Guia do usuário do Amazon Simple Storage Service.

Importe o arquivo gzip para a sua do cluster de bancos de dados Aurora PostgreSQL, conforme mostrado a seguir.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

Importar um arquivo do Amazon S3 codificado

O exemplo a seguir mostra como importar um arquivo do Amazon S3 que tenha codificação Windows-1252.

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

Referência de funções

Funções

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Importa dados do Amazon S3 para uma tabela do do Aurora PostgreSQL. A extensão `aws_s3` fornece a função `aws_s3.table_import_from_s3`. O valor de retorno é texto.

Sintaxe

Os parâmetros necessários são `table_name`, `column_list` e `options`. Eles identificam a tabela do banco de dados e especificam como os dados são copiados nessa tabela.

Você também pode usar os seguintes parâmetros:

- O parâmetro `s3_info` especifica o arquivo do Amazon S3 a ser importado. Quando você usa esse parâmetro, o acesso ao Amazon S3 é fornecido por uma função do IAM para a do cluster de banco de dados do PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- O parâmetro `credentials` especifica as credenciais para acessar o Amazon S3. Ao usar esse parâmetro, você não usa uma função do IAM.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

Parâmetros

table_name

Uma string de texto necessária contendo o nome da tabela de banco de dados PostgreSQL para a qual importar os dados.

column_list

Uma string de texto necessária contendo uma lista opcional das colunas da tabela de banco de dados PostgreSQL para a qual copiar os dados. Se a string estiver vazia, todas as colunas da tabela serão usadas. Para ver um exemplo, consulte Importar um arquivo do Amazon S3 que usa um delimitador personalizado.

Opções do

Uma string de texto necessária contendo argumentos para o comando PostgreSQL `COPY`. Esses argumentos especificam como os dados devem ser copiados para a tabela do PostgreSQL. Para obter mais detalhes, consulte Documentação de COPY do PostgreSQL.

s3_info

Um tipo composto `aws_commons._s3_uri_1` contendo as seguintes informações sobre o objeto do S3:

- `bucket` – O nome do Amazon S3 bucket que contém o arquivo.
- `file_path` – o nome do Amazon S3 arquivo, incluindo o caminho do arquivo.
- `region`: a região da AWS em que o arquivo se encontra. Para obter uma listagem de nomes de regiões da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

credenciais

Um tipo composto `aws_commons._aws_credentials_1` contendo as seguintes credenciais a serem usadas para a operação de importação:

- Chave de acesso
- Chave secreta
- Token de sessão

Para obter informações sobre como criar uma estrutura `aws_commons._aws_credentials_1` composta, consulte aws_commons.create_aws_credentials.

Sintaxe alternativa

Para ajudar nos testes, você pode usar um conjunto expandido de parâmetros em vez dos parâmetros `s3_info` e `credentials`. Veja a seguir as variações acionais da sintaxe da função `aws_s3.table_import_from_s3`.

- Em vez de usar o parâmetro `s3_info` para identificar um arquivo do Amazon S3, use a combinação dos parâmetros `bucket`, `file_path` e `region`. Com essa forma da função, o acesso ao Amazon S3 é fornecido por uma função do IAM na instância de banco de dados do PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- Em vez de usar o parâmetro `credentials` para especificar o acesso ao Amazon S3, use a combinação dos parâmetros `access_key`, `session_key` e `session_token`.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text,
   access_key text,
   secret_key text,
   session_token text
)
```

Parâmetros alternativos

Bucket de

Uma string de texto contendo o nome do bucket do Amazon S3 que contém o arquivo.

file_path

Uma string de texto contendo o nome do arquivo do Amazon S3, incluindo o caminho do arquivo.

region

Uma string de texto que identifica a Região da AWS do arquivo. Para obter uma listagem de nomes de Região da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

access_key

Uma string de texto contendo a chave de acesso a ser usada para a operação de importação. O padrão é NULL.

secret_key

Uma string de texto contendo a chave secreta a ser usada para a operação de importação. O padrão é NULL.

session_token

(Opcional) Uma string de texto contendo a chave de sessão a ser usada para a operação de importação. O padrão é NULL.

aws_commons.create_s3_uri

Cria uma estrutura `aws_commons._s3_uri_1` para comportar informações do arquivo do Amazon S3. Use os resultados da função `aws_commons.create_s3_uri` no parâmetro `s3_info` da função aws_s3.table_import_from_s3.

Sintaxe

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parâmetros

Bucket de

Uma string de texto necessária contendo o nome do bucket do Amazon S3 para o arquivo.

file_path

Uma string de texto necessária contendo o nome do arquivo do Amazon S3, incluindo o caminho do arquivo.

region

Uma string de texto necessária que contém a Região da AWS na qual o arquivo se encontra. Para obter uma listagem de nomes de Região da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

aws_commons.create_aws_credentials

Define uma chave de acesso e uma chave secreta em uma estrutura `aws_commons._aws_credentials_1`. Use os resultados da função `aws_commons.create_aws_credentials` no parâmetro `credentials` da função aws_s3.table_import_from_s3.

Sintaxe

```
aws_commons.create_aws_credentials(
   access_key text,
```

```
    secret_key text,
    session_token text
)
```

Parâmetros

access_key

Uma string de texto necessária contendo a chave de acesso a ser usada para importar um arquivo do Amazon S3. O padrão é NULL.

secret_key

Uma string de texto necessária contendo a chave secreta a ser usada para importar um arquivo do Amazon S3. O padrão é NULL.

session_token

Uma string de texto opcional contendo o token de sessão a ser usado para importar um arquivo do Amazon S3. O padrão é NULL. Se você fornecer um `session_token` opcional, poderá usar credenciais temporárias.

Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3

Consulte dados em um Aurora PostgreSQLcluster de banco de dados e exporte-os diretamente para arquivos armazenados em um bucket do Amazon S3. Para fazer isso, primeiro instale a extensão `aws_s3` do Aurora PostgreSQL . Essa extensão fornece as funções usadas para exportar os resultados de consultas para o Amazon S3. A seguir, é possível descobrir como instalar a extensão e exportar dados para o Amazon S3.

Você pode exportar de uma instância de banco de dados provisionada ou de uma instância de banco de dados do Aurora Serverless v2. Essas etapas não são compatíveis com o Aurora Serverless v1.

> **Note**
>
> A exportação entre contas não é compatível com o Amazon S3.

Todas as versões do Aurora PostgreSQL disponíveis no momento são compatíveis com a exportação de dados do Amazon Simple Storage Service. Para obter informações detalhadas sobre

as versões, consulte Atualizações do Amazon Aurora PostgreSQL nas Notas de lançamento do Aurora PostgreSQL.

Se você não tiver um bucket configurado para sua exportação, consulte os tópicos a seguir no Guia do usuário do Amazon Simple Storage Service.

- Configuração do Amazon S3
- Como criar um bucket

Por padrão, os dados exportados do Aurora PostgreSQL ao Amazon S3 usam criptografia do lado do servidor com uma Chave gerenciada pela AWS. Também é possível usar uma chave gerenciada pelo cliente que você já tenha criado. Se estiver usando a criptografia do bucket, o bucket do Amazon S3 deverá ser criptografado com uma chave do AWS Key Management Service (AWS KMS) (SSE-KMS). No momento, os buckets criptografados com chaves gerenciadas pelo Amazon S3 (SSE-S3) não são compatíveis.

> **Note**
>
> É possível salvar o banco de dados e os dados de snapshot do cluster de banco de dados no Amazon S3 usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS. Para obter mais informações, consulte Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.

Tópicos

- Instalar a extensão aws_s3
- Visão geral da exportação de dados para o Amazon S3
- Especificar o caminho do arquivo do Amazon S3 para o qual exportar
- Configurar o acesso a um bucket do Amazon S3
- Exportar dados de consulta usando a função aws_s3.query_export_to_s3
- Referência de funções
- Solução de problemas de acesso ao Amazon S3

Instalar a extensão aws_s3

Antes de usar o Amazon Simple Storage Service com o seu cluster de banco de dados do Aurora PostgreSQL, você precisa instalar a extensão `aws_s3`. Essa extensão fornece funções para exportar dados da instância gravadora de um cluster de banco de dados do Aurora PostgreSQL para um bucket do Amazon S3. Também fornece as funções para importar dados do Amazon S3. Para obter mais informações, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL. A extensão `aws_s3` depende de algumas das funções auxiliares da extensão `aws_commons`, que é instalada automaticamente quando necessária.

Como instalar a extensão **`aws_s3`**

1. Use psql (ou pgAdmin) para se conectar à instância gravadora de seu cluster de banco de dados do Aurora PostgreSQL como um usuário que tem privilégios `rds_superuser`. Se você manteve o nome padrão durante o processo de configuração, se conectará como `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Para instalar a extensão, execute o comando a seguir.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. Para verificar se a extensão está instalada, você pode usar o metacomando psql `\dx`.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

As funções para importar dados do Amazon S3 e exportar dados para o Amazon S3 agora estão disponíveis para uso.

Confira se a sua versão do Aurora PostgreSQL permite exportações para o Amazon S3.

Você pode verificar se sua versão do Aurora PostgreSQL oferece suporte à exportação para o Amazon S3 usando o comando `describe-db-engine-versions`. O exemplo a seguir verifica se a versão 10.14 pode ser exportada para o Amazon S3.

```
aws rds describe-db-engine-versions --region us-east-1 \
--engine aurora-postgresql --engine-version 10.14 | grep s3Export
```

Se a saída incluir a string `"s3Export"`, o mecanismo é compatível com exportações do Amazon S3. Caso contrário, não há compatibilidade.

Visão geral da exportação de dados para o Amazon S3

Para exportar dados armazenados em um banco de dados Aurora PostgreSQL para um bucket do Amazon S3, use o procedimento a seguir.

Para exportar dados do Aurora PostgreSQL para o S3

1. Identifique um caminho de arquivo do Amazon S3 a ser usado para exportar dados. Para obter detalhes sobre esse processo, consulte Especificar o caminho do arquivo do Amazon S3 para o qual exportar.
2. Conceda permissão para acessar o bucket do Amazon S3.

   Para exportar dados para um arquivo do Amazon S3, forneça ao cluster de bancos de dados Aurora PostgreSQL permissão para acessar o bucket do Amazon S3 a ser usado para armazenamento pela exportação. Isso inclui as seguintes etapas:

   1. Crie uma política do IAM que forneça acesso a um bucket do Amazon S3 para o qual você deseja exportar.
   2. Crie uma função do IAM.
   3. Anexe a política criada à função criada.
   4. Adicione esse perfil do IAM ao cluster de banco de dados.

   Para obter detalhes sobre esse processo, consulte Configurar o acesso a um bucket do Amazon S3.
3. Identifique uma consulta de banco de dados para obter os dados. Exporte os dados da consulta chamando a função `aws_s3.query_export_to_s3`.

Após concluir as tarefas de preparação anteriores, use a função aws_s3.query_export_to_s3 para exportar os resultados da consulta para o Amazon S3. Para obter detalhes sobre esse processo, consulte Exportar dados de consulta usando a função aws_s3.query_export_to_s3.

Especificar o caminho do arquivo do Amazon S3 para o qual exportar

Especifique as seguintes informações para identificar o local no Amazon S3 para o qual deseja exportar dados:

- Nome do bucket – um bucket é um contêiner para objetos ou arquivos do Amazon S3.

  Para ter mais informações sobre como armazenar dados com o Amazon S3, consulte Criação de um bucket e Trabalhar com objetos no Guia do usuário do Amazon Simple Storage Service.
- Caminho do arquivo – o caminho do arquivo identifica onde a exportação é armazenada no bucket do Amazon S3. O caminho do arquivo consiste no seguinte:
  - Um prefixo de caminho opcional que identifica um caminho de pasta virtual.
  - Um prefixo de arquivo que identifica um ou mais arquivos a serem armazenados. Exportações maiores são armazenadas em vários arquivos, cada um com um tamanho máximo de aproximadamente 6 GB. Os nomes de arquivos adicionais têm o mesmo prefixo de arquivo, mas com o acréscimo de `_part`*XX*. O *XX* representa 2, depois 3 e assim por diante.

  Por exemplo, um caminho de arquivo com uma pasta `exports` e um prefixo de arquivo `query-1-export` é `/exports/query-1-export`.
- Região da AWS (opcional): a região da AWS onde o bucket do Amazon S3 está localizado. Se você não especificar um valor de região da AWS, o Aurora salvará os arquivos no Amazon S3, na mesma região da AWS que a instância de banco de dados de exportação.

> **Note**
>
> Atualmente, a região AWS deve ser a mesma que a região de exportação do cluster de banco de dados.

Para obter uma listagem de nomes de regiões da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

Para manter as informações do arquivo do Amazon S3 sobre onde a exportação deve ser armazenada, você pode usar a função aws_commons.create_s3_uri para criar uma estrutura `aws_commons._s3_uri_1` composta da seguinte forma.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Posteriormente, você fornece esse valor de `s3_uri_1` como um parâmetro na chamada para a função aws_s3.query_export_to_s3. Para ver exemplos, consulte Exportar dados de consulta usando a função aws_s3.query_export_to_s3.

Configurar o acesso a um bucket do Amazon S3

Para exportar dados ao Amazon S3, forneça ao cluster de banco de dados PostgreSQL permissão para acessar o bucket do Amazon S3 para o qual os arquivos serão enviados.

Para fazer isso, use o procedimento a seguir.

Como conceder acesso a um cluster de banco de dados PostgreSQL ao Amazon S3 por meio de um perfil do IAM

1. Crie uma política do IAM.

   Essa política concede ao bucket e ao objeto as permissões para que o cluster de banco de dados PostgreSQL acesse o Amazon S3.

   Como parte da criação dessa política, execute as seguintes etapas:

   a. Inclua na política as seguintes ações necessárias para permitir a transferência de arquivos do cluster de banco de dados PostgreSQL para um bucket do Amazon S3:

      - `s3:PutObject`
      - `s3:AbortMultipartUpload`

   b. Inclua o nome do recurso da Amazon (ARN) que identifica o bucket do Amazon S3 e os objetos no bucket. O formato do ARN para acessar o Amazon S3 é: `arn:aws:s3:::amzn-s3-demo-bucket/*`

Para obter mais informações sobre como criar uma política do IAM para o Aurora PostgreSQL, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM. Consulte também Tutorial: criar e anexar sua primeira política gerenciada pelo cliente no Manual do usuário do IAM.

O comando da AWS CLI a seguir cria uma política do IAM denominada `rds-s3-export-policy` com essas opções. Ele concede acesso a um bucket denominado *`amzn-s3-demo-bucket`*.

> **Warning**
>
> Recomendamos configurar o banco de dados em uma VPC privada que tenha políticas de endpoint configuradas para acessar buckets específicos. Para obter mais informações, consulte Usar políticas de endpoint para o Amazon S3 no Manual do usuário da Amazon VPC.
> É altamente recomendável não criar uma política com acesso a todos os recursos. Esse acesso pode representar uma ameaça à segurança dos dados. Se você criar uma política que forneça o acesso `S3:PutObject` a todos os recursos usando `"Resource":"*"`, um usuário com privilégios de exportação poderá exportar dados para todos os buckets em sua conta. Além disso, o usuário poderá exportar dados para qualquer bucket gravável publicamente dentro de sua região daAWS.

Depois de criar a política, observe o nome do recurso da Amazon (ARN) da política. O ARN será necessário para uma etapa posterior, quando você anexar a política a um perfil do IAM.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
```

```
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }'
```

2. Crie uma função do IAM.

   Faça isso para que o do Aurora PostgreSQL possa assumir esse perfil do IAM em seu nome para acessar os buckets do Amazon S3. Para ter mais informações, consulte Criar um perfil para delegar permissões a um usuário do IAM no Guia do usuário do IAM.

   Convém usar as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas baseadas em recursos para limitar as permissões do serviço a um recurso específico. Essa é a maneira mais eficiente de se proteger contra o problema "confused deputy".

   Se você utilizar ambas as chaves de contexto de condição global, e o valor `aws:SourceArn` contiver o ID da conta, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão utilizar o mesmo ID de conta quando utilizados na mesma instrução de política.

   - Use `aws:SourceArn` se quiser acesso entre serviços para um único recurso.
   - Use `aws:SourceAccount` se você quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

   Na política, certifique-se de usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. O exemplo a seguir mostra como fazer isso utilizando o comando da AWS CLI para criar uma função chamada `rds-s3-export-role`.

   Example

   Para Linux, macOS ou Unix:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
```

```
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

Para Windows:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

3. Anexe a política do IAM que você criou à função do IAM que você criou.

O comando da AWS CLI a seguir anexa a política criada anteriormente à função chamada `rds-s3-export-role.`. Substitua *`your-policy-arn`* pelo ARN da política que você anotou na etapa anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-export-role
```

4. Adicione o perfil do IAM ao cluster de banco de dados. Faça isso usando o Console de gerenciamento da AWS ou a AWS CLI, conforme descrito a seguir.

Console

Para adicionar um perfil do IAM a um cluster de banco de dados do PostgreSQL usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha o nome da de cluster de banco de dados do PostgreSQL para exibir os detalhes.
3. Na guia Connectivity & security (Conectividade e segurança), na seção Manage IAM roles (Gerenciar perfis do IAM), escolha o perfil a ser adicionado em Add IAM roles to this instance (Adicionar perfis do IAM a essa instância).
4. Em Feature (Recurso), escolha s3Export.
5. Escolha Add role (adicionar função).

AWS CLI

Para adicionar um perfil do IAM a um cluster de banco de dados do PostgreSQL usando a CLI

- Use o comando a seguir para adicionar a função ao cluster de banco de dados do PostgreSQL chamado `my-db-cluster`. Substitua*`your-role-arn`* pelo ARN da função que você anotou em uma etapa anterior. Use `s3Export` para o valor da opção `--feature-name`.

  Example

  Para Linux, macOS ou Unix:

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
```

```
    --feature-name s3Export \
    --role-arn your-role-arn   \
    --region your-region
```

Para Windows:

```
aws rds add-role-to-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --feature-name s3Export ^
    --role-arn your-role-arn ^
    --region your-region
```

Exportar dados de consulta usando a função aws_s3.query_export_to_s3

Exporte os dados do PostgreSQL para o Amazon S3 chamando a função aws_s3.query_export_to_s3.

Tópicos

- Pré-requisitos
- Chamar aws_s3.query_export_to_s3
- Exportar para um arquivo CSV que usa um delimitador personalizado
- Exportar para um arquivo binário com codificação

Pré-requisitos

Antes de usar a função `aws_s3.query_export_to_s3`, verifique se você concluiu os seguintes pré-requisitos:

- Instalar as extensões do PostgreSQL necessárias, conforme descrito em Visão geral da exportação de dados para o Amazon S3.
- Determinar para onde exportar os dados para o Amazon S3, conforme descrito em Especificar o caminho do arquivo do Amazon S3 para o qual exportar.
- Verifique se o cluster de banco de dados tem acesso ao Amazon S3 conforme descrito em Configurar o acesso a um bucket do Amazon S3.

Os exemplos a seguir usam uma tabela de banco de dados chamada `sample_table`. Esses exemplos exportam os dados para um bucket chamado *`amzn-s3-demo-bucket`*. A tabela e os dados de exemplo são criados com as seguintes instruções SQL no psql.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

Chamar aws_s3.query_export_to_s3

Veja a seguir as formas básicas de chamar a função aws_s3.query_export_to_s3.

Esses exemplos usam a variável `s3_uri_1` para identificar uma estrutura que contém as informações que identificam o arquivo do Amazon S3. Use a função aws_commons.create_s3_uri para criar a estrutura.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Embora os parâmetros variem para as duas chamadas de função `aws_s3.query_export_to_s3` a seguir, os resultados são os mesmos para esses exemplos. Todas as linhas da tabela `sample_table` são exportadas para um bucket chamado *`amzn-s3-demo-bucket`*.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

Os parâmetros são descritos da seguinte forma:

- `'SELECT * FROM sample_table'` – o primeiro parâmetro é uma string de texto necessária que contém uma consulta SQL. O mecanismo PostgreSQL executa essa consulta. Os resultados da consulta são copiados no bucket do S3 identificado em outros parâmetros.
- `:'s3_uri_1'` – esse parâmetro é uma estrutura que identifica o arquivo do Amazon S3. Esse exemplo usa uma variável para identificar a estrutura criada anteriormente. Em vez disso, você

pode criar a estrutura incluindo a chamada de função `aws_commons.create_s3_uri` em linha dentro da chamada de função `aws_s3.query_export_to_s3` da seguinte forma.

```
SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
);
```

- `options :='format text'` – o parâmetro `options` é uma string de texto opcional que contém argumentos `COPY` do PostgreSQL. O processo de cópia usa os argumentos e o formato do comando COPY PostgreSQL.

Se o arquivo especificado não existir no bucket do Amazon S3, ele será criado. Se o arquivo já existir, ele será substituído. A sintaxe para acessar os dados exportados no Amazon S3 é a seguinte.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

Exportações maiores são armazenadas em vários arquivos, cada um com um tamanho máximo de aproximadamente 6 GB. Os nomes de arquivos adicionais têm o mesmo prefixo de arquivo, mas com o acréscimo de `_part`*XX*. O *XX* representa 2, depois 3 e assim por diante. Por exemplo, suponha que você especifique o caminho onde armazena arquivos de dados como o seguinte.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

Se a exportação precisar criar três arquivos de dados, o bucket do Amazon S3 conterá os seguintes arquivos de dados.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

Para obter a referência completa para esta função e as formas adicionais de chamá-la, consulte aws_s3.query_export_to_s3. Para obter mais informações sobre como acessar arquivos no Amazon S3, consulte Visualização de um objeto no Guia do usuário do Amazon Simple Storage Service.

Exportar para um arquivo CSV que usa um delimitador personalizado

O exemplo a seguir mostra como chamar a função aws_s3.query_export_to_s3 para exportar dados para um arquivo que usa um delimitador personalizado. O exemplo usa argumentos do

comando COPY do PostgreSQL para especificar o formato de valor separado por vírgula (CSV) e um delimitador de dois pontos (:).

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

Exportar para um arquivo binário com codificação

O exemplo a seguir mostra como chamar a função aws_s3.query_export_to_s3 para exportar dados para um arquivo binário que tenha a codificação Windows-1253.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

Referência de funções

Funções

- aws_s3.query_export_to_s3
- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

Exporta um resultado de consulta do PostgreSQL para um bucket do Amazon S3. A extensão `aws_s3` fornece a função `aws_s3.query_export_to_s3`.

Os dois parâmetros necessários são `query` e `s3_info`. Eles definem a consulta a ser exportada e identificam o bucket do Amazon S3 para o qual exportar. Um parâmetro opcional chamado `options` fornece a definição de vários parâmetros de exportação. Para obter exemplos de como usar a função `aws_s3.query_export_to_s3`, consulte Exportar dados de consulta usando a função aws_s3.query_export_to_s3.

Sintaxe

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
```

```
)
```

Parâmetros de entrada

query

Uma string de texto necessária que contém uma consulta SQL executada pelo mecanismo do PostgreSQL. Os resultados dessa consulta são copiados em um bucket do S3 identificado no parâmetro `s3_info`.

s3_info

Um tipo composto `aws_commons._s3_uri_1` contendo as seguintes informações sobre o objeto do S3:

- `bucket` – o nome do bucket do Amazon S3 que deve conter o arquivo.
- `file_path` – o nome e o caminho do arquivo do Amazon S3.
- `region`: a região da AWS na qual o bucket se encontra. Para obter uma listagem de nomes de regiões da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

  Atualmente, esse valor deve ser a mesma região da AWS que de exportação do cluster de banco de dados. O padrão é a região da AWS para exportação do cluster de banco de dados.

Para criar uma estrutura `aws_commons._s3_uri_1` composta, consulte a função aws_commons.create_s3_uri.

options

Uma string de texto opcional que contém argumentos para o comando `COPY` do PostgreSQL. Esses argumentos especificam como os dados devem ser copiados quando exportados. Para obter mais detalhes, consulte Documentação de COPY do PostgreSQL.

kms_key text

Uma string de texto opcional contendo a chave do KMS gerenciada pelo cliente do bucket do S3 para o qual exportar os dados.

Parâmetros de entrada alternativos

Para ajudar nos testes, você pode usar um conjunto expandido de parâmetros em vez do parâmetro `s3_info`. Veja a seguir as variações adicionais de sintaxe da função `aws_s3.query_export_to_s3`.

Em vez de usar o parâmetro `s3_info` para identificar um arquivo do Amazon S3, use a combinação dos parâmetros `bucket`, `file_path` e `region`.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query

Uma string de texto necessária que contém uma consulta SQL executada pelo mecanismo do PostgreSQL. Os resultados dessa consulta são copiados em um bucket do S3 identificado no parâmetro `s3_info`.

bucket

Uma string de texto necessária que contém o nome do bucket do Amazon S3 que contém o arquivo.

file_path

Uma string de texto necessária contendo o nome do arquivo do Amazon S3, incluindo o caminho do arquivo.

region

Uma string de texto opcional que contém a região da AWS na qual o bucket se contra. Para obter uma listagem de nomes de regiões da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

Atualmente, esse valor deve ser a mesma região da AWS que de exportação do cluster de banco de dados. O padrão é a região da AWS para exportação do cluster de banco de dados.

options

Uma string de texto opcional que contém argumentos para o comando `COPY` do PostgreSQL. Esses argumentos especificam como os dados devem ser copiados quando exportados. Para obter mais detalhes, consulte Documentação de COPY do PostgreSQL.

kms_key text

Uma string de texto opcional contendo a chave do KMS gerenciada pelo cliente do bucket do S3 para o qual exportar os dados.

Parâmetros de saída

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

O número de linhas da tabela que foram carregadas com êxito no Amazon S3 para a determinada consulta.

files_uploaded

O número de arquivos carregados no Amazon S3. Os arquivos são criados em tamanhos de aproximadamente 6 GB. Cada arquivo adicional criado tem `_part`*XX* acrescentado ao nome. O *XX* representa 2, depois 3 e assim por diante, conforme necessário.

bytes_uploaded

O número total de bytes carregados no Amazon S3.

Exemplos

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Cria uma estrutura `aws_commons._s3_uri_1` para comportar informações do arquivo do Amazon S3. Use os resultados da função `aws_commons.create_s3_uri` no parâmetro

`s3_info` da função aws_s3.query_export_to_s3. Para obter um exemplo de uso da função `aws_commons.create_s3_uri`, consulte Especificar o caminho do arquivo do Amazon S3 para o qual exportar.

Sintaxe

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parâmetros de entrada

Bucket de

Uma string de texto necessária contendo o nome do bucket do Amazon S3 para o arquivo.

file_path

Uma string de texto necessária contendo o nome do arquivo do Amazon S3, incluindo o caminho do arquivo.

region

Uma string de texto necessária que contém a região da AWS na qual o arquivo se encontra. Para obter uma listagem de nomes de regiões da AWS e os valores associados, consulte Regiões e zonas de disponibilidade.

Solução de problemas de acesso ao Amazon S3

Se você encontrar problemas de conexão ao tentar exportar dados para o Amazon S3, primeiro confirme se as regras de acesso de saída para o grupo de segurança da VPC associado à sua instância de banco de dados permitem conectividade de rede. Especificamente, o grupo de segurança deve ter uma regra que permita que a instância de banco de dados envie tráfego TCP para a porta 443 e para todos os endereços IPv4 (0.0.0.0/0). Para obter mais informações, consulte Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança.

Veja também as seguintes recomendações:

- Solução de problemas de identidade e acesso do Amazon Aurora
- Solução de problemas do Amazon S3 no Guia do usuário do Amazon Simple Storage Service

- Troubleshooting Amazon S3 and IAM (Solucionar problemas no Amazon S3 e no IAM) no Guia do usuário do IAM

Invocar uma função do AWS Lambda de um cluster de banco de dados do Aurora PostgreSQL

AWS LambdaO é um serviço computacional orientado a eventos que permite executar código sem o provisionamento ou gerenciamento de servidores. Está disponível para uso com muitos serviços da AWS, incluindo o Aurora PostgreSQL. Por exemplo, você pode usar funções Lambda para processar notificações de eventos de um banco de dados ou para carregar dados de arquivos sempre que um novo arquivo é carregado para o Simple Storage Service (Amazon S3). Para saber mais sobre o Lambda, consulte O que é o AWS Lambda? no Guia do desenvolvedor do AWS Lambda.

> **Note**
>
> A chamada de funções do AWS Lambda é compatível no Aurora PostgreSQL 11.9 e em versões posteriores (incluindo Aurora Serverless v2).

Configurar o Aurora PostgreSQL para trabalhar com funções Lambda é um processo de várias etapas envolvendo o AWS Lambda, o IAM, sua VPC e seu cluster de bancos de dados Aurora PostgreSQL. A seguir, você pode encontrar resumos das etapas necessárias.

Para obter mais informações sobre como criar uma função Lambda, consulte Conceitos básicos do Lambda e Tópicos essenciais do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

Tópicos

- Etapa 1: configurar o cluster de bancos de dados do Aurora PostgreSQL para conexões de saída para o AWS Lambda
- Etapa 2: configurar o IAM para o cluster de bancos de dados do Aurora PostgreSQL e o AWS Lambda
- Etapa 3: instalar a extensão aws_lambda para um cluster de bancos de dados do Aurora PostgreSQL
- Etapa 4: usar as funções Lambda auxiliares com o cluster de bancos de dados Aurora PostgreSQL (opcional)
- Etapa 5: invocar uma função Lambda do seu cluster de bancos de dados Aurora PostgreSQL

- Etapa 6: Conceder a outros usuários permissão para invocar funções do Lambda
- Exemplos: Invocar uma função do Lambda do cluster de banco de dados do Aurora PostgreSQL
- Mensagens de erro da função Lambda
- Função do AWS Lambda e referência de parâmetros

Etapa 1: configurar o cluster de bancos de dados do Aurora PostgreSQL para conexões de saída para o AWS Lambda

As funções do Lambda sempre são executadas dentro de uma Amazon VPC de propriedade do serviço AWS Lambda. O Lambda aplica regras de segurança e acesso à rede a essa VPC e mantém e monitora a VPC automaticamente. Seu cluster de bancos de dados do Aurora PostgreSQL envia tráfego de rede para a VPC do serviço Lambda. Como você configura isso depende se sua instância de banco de dados primária do cluster de bancos de dados Aurora é pública ou privada.

- Cluster de banco de dados público do Aurora PostgreSQL: a instância de banco de dados primária de um cluster de banco de dados é pública se está localizada em uma sub-rede pública na VPC e se a propriedade “PubliclyAccessible” da instância é `true`. Para encontrar o valor dessa propriedade, você pode usar o comando da AWS CLI describe-db-instances. Ou, você pode usar o Console de gerenciamento da AWS para abrir a guia Connectivity & security (Conectividade e segurança) e verificar se Publicly accessible (Publicamente acessível) está definido como Yes (Sim). Para verificar se a instância está na sub-rede pública da VPC, use o Console de gerenciamento da AWS ou a AWS CLI.

  Para configurar o acesso ao Lambda, use o Console de gerenciamento da AWS ou a AWS CLI para criar uma regra de saída no grupo de segurança da VPC. A regra de saída especifica que o TCP pode usar a porta 443 para enviar pacotes para qualquer endereço IPv4 (0.0.0.0/0).
- Cluster de banco de dados privado do Aurora PostgreSQL: nesse caso, a propriedade “PubliclyAccessible” da instância é `false` ou está em uma sub-rede privada. Para permitir que a instância funcione com o Lambda, use um gateway de conversão de endereços de rede (NAT). Para obter mais informações, consulte Gateways de NAT. Ou configure sua VPC com um endpoint da VPC para o Lambda. Para obter mais informações, consulte Endpoints da VPC no Guia do usuário da Amazon VPC. O endpoint responde às chamadas feitas pelo cluster de bancos de dados do Aurora PostgreSQL para suas funções do Lambda.

Agora, sua VPC pode interagir com o VPC do AWS Lambda no nível da rede. Depois, configure as permissões usando o IAM.

Etapa 2: configurar o IAM para o cluster de bancos de dados do Aurora PostgreSQL e o AWS Lambda

Invocar funções Lambda do seu cluster de bancos de dados Aurora PostgreSQL requer certos privilégios. Para configurar os privilégios necessários, recomendamos que você crie uma política do IAM que permita chamar funções Lambda, atribuir essa política a uma função e, em seguida, aplicar a função ao cluster de banco de dados. Essa abordagem dá ao cluster de banco de dados privilégios para invocar a função Lambda especificada em seu nome. As etapas a seguir mostram como fazer isso usando a AWS CLI.

Para configurar permissões do IAM para usar seu cluster com o Lambda

1. Use o comando create-policy da AWS CLI para criar uma política do IAM que permita que o seu cluster de bancos de dados do Aurora PostgreSQL invoque a função do Lambda especificada. (O ID da instrução (Sid) é uma descrição opcional para sua instrução de política e não tem efeito sobre o uso.) Esta política fornece ao seu cluster de bancos de dados Aurora as permissões mínimas necessárias para invocar a função Lambda especificada.

   ```
   aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Sid": "AllowAccessToExampleFunction",
           "Effect": "Allow",
           "Action": "lambda:InvokeFunction",
           "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
           }
       ]
   }'
   ```

   Você também pode usar a política `AWSLambdaRole` predefinida que permite invocar qualquer uma das suas funções Lambda. Para obter mais informações, consulte Políticas do IAM baseadas em identidade para o Lambda.

2. Use o comando da AWS CLI create-role para criar uma função do IAM que a política possa assumir em tempo de execução.

   ```
   aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
       "Version": "2012-10-17",
       "Statement": [
   ```

```
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Anexe a política à função usando o comando da AWS CLI attach-role-policy.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Aplique o perfil ao seu cluster de bancos de dados do Aurora PostgreSQL usando o comando add-role-to-db-cluster AWS CLI. Esta última etapa permite que seus usuários de banco de dados de clusters de banco de dados invoquem funções Lambda.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Com a VPC e as configurações do IAM concluídas, agora você pode instalar a extensão `aws_lambda`. (Observe que você pode instalar a extensão a qualquer momento, mas até configurar o suporte à VPC e os privilégios do IAM corretos, a extensão `aws_lambda` não adiciona nada às capacidades do cluster de bancos de dados Aurora PostgreSQL.)

Etapa 3: instalar a extensão **`aws_lambda`** para um cluster de bancos de dados do Aurora PostgreSQL

Para usar o AWS Lambda com o cluster de banco de dados do Aurora PostgreSQL, adicione a extensão `aws_lambda` do PostgreSQL ao cluster de banco de dados do Aurora PostgreSQL. Esta extensão fornece o cluster de bancos de dados Aurora PostgreSQL com a capacidade de chamar funções Lambda do PostgreSQL.

Como instalar a extensão **aws_lambda** em um cluster de bancos de dados do Aurora PostgreSQL

Use a linha de comando psql do PostgreSQL ou a ferramenta pgAdmin para se conectar ao seu cluster de bancos de dados Aurora PostgreSQL.

1. Conecte-se ao seu cluster de bancos de dados Aurora PostgreSQL como usuário com privilégios rds_superuser. O usuário padrão postgres é mostrado no exemplo.

```
psql -h cluster-instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p
 5432
```

2. Instale a extensão aws_lambda. A extensão aws_commons também é necessária. Ela fornece funções auxiliares para aws_lambda e muitas outras extensões do Aurora para PostgreSQL. Se ainda não estiver no seu cluster do Aurora PostgreSQL, ela é instalada com aws_lambda como mostrado a seguir.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

A extensão aws_lambda está instalada em sua instância de banco de dados primária do cluster de bancos de dados Aurora PostgreSQL. Agora você pode criar estruturas de conveniência para chamar suas funções Lambda.

Etapa 4: usar as funções Lambda auxiliares com o cluster de bancos de dados Aurora PostgreSQL (opcional)

Você pode usar as funções auxiliares na extensão aws_commons para preparar entidades que você pode invocar mais facilmente a partir do PostgreSQL. Para isso, você precisa ter as seguintes informações sobre suas funções Lambda:

- Nome da função – O nome, nome do recurso da Amazon (ARN), versão ou apelido da função Lambda. A política do IAM criada em Etapa 2: configurar o IAM para a instância e o Lambda requer o ARN, portanto, recomendamos que você use o ARN da sua função.
- Região da AWS – (Opcional) A região da AWS onde a função Lambda está localizada se não estiver na mesma região que seu cluster de bancos de dados Aurora PostgreSQL.

Para manter as informações do nome da função Lambda, você pode usar a função aws_commons.create_lambda_function_arn. Esta função auxiliar cria uma estrutura composta `aws_commons._lambda_function_arn_1` com os detalhes necessários para a função de invocação. A seguir, você pode encontrar três abordagens alternativas para configurar essa estrutura composta.

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

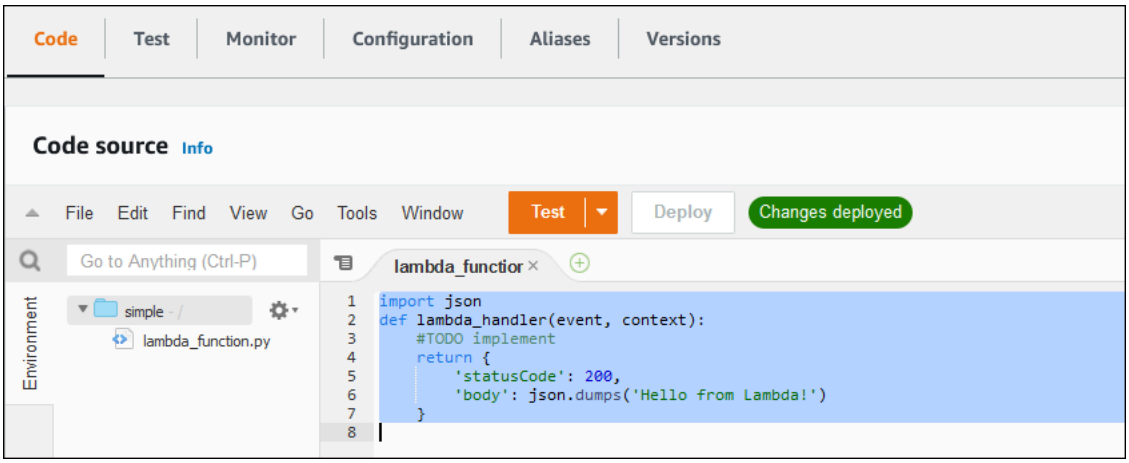
Qualquer um desses valores pode ser usado em chamadas para a função aws_lambda.invoke. Para obter exemplos, consulte Etapa 5: invocar uma função Lambda do seu cluster de bancos de dados Aurora PostgreSQL.

Etapa 5: invocar uma função Lambda do seu cluster de bancos de dados Aurora PostgreSQL

A função `aws_lambda.invoke` se comporta de forma síncrona ou assíncrona, dependendo do `invocation_type`. As duas alternativas para esse parâmetro são `RequestResponse` (o padrão) e `Event`, da seguinte forma:

- **`RequestResponse`** – Este tipo de invocação é síncrono. É o comportamento padrão quando a chamada é feita sem especificar um tipo de invocação. A carga útil da resposta inclui os resultados da função `aws_lambda.invoke`. Use esse tipo de invocação de quando seu fluxo de trabalho exigir o recebimento de resultados da função Lambda antes de continuar.
- **`Event`** – Este tipo de invocação é assíncrono. A resposta não inclui uma carga útil contendo resultados. Use esse tipo de invocação quando o fluxo de trabalho não precisar de um resultado da função Lambda para continuar o processamento.

Como um teste simples de sua configuração, você pode se conectar à sua instância de banco de dados usando `psql` e chamar uma função de exemplo a partir da linha de comando. Suponha que você tenha uma das funções básicas configuradas em seu serviço Lambda, como a função Python simples mostrada na captura de tela a seguir.



Para invocar uma função de exemplo

1. Conecte-se à sua instância de banco de dados primária usando `psql` ou pgAdmin.

   ```
   psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. Invoque a função usando seu ARN.

   ```
   SELECT * from
    aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
   region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
    Postgres!"}'::json );
   ```

   A resposta se parece com esta a seguir.

   ```
   status_code |                         payload                          |
    executed_version | log_result
   -------------+----------------------------------------------------------
   +------------------+------------
            200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
        |
   (1 row)
   ```

Se sua tentativa de invocação não for bem-sucedida, consulte Mensagens de erro da função Lambda.

Etapa 6: Conceder a outros usuários permissão para invocar funções do Lambda

Neste ponto dos procedimentos, apenas você como `rds_superuser` pode invocar suas funções do Lambda. Para permitir que outros usuários invoquem quaisquer funções criadas por você, é necessário conceder permissão a eles.

Como conceder a outros permissão para invocar funções do Lambda

1. Conecte-se à sua instância de banco de dados primário usando `psql` ou pgAdmin.

```
psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. Execute os seguintes comandos SQL:

```
postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
```

Exemplos: Invocar uma função do Lambda do cluster de banco de dados do Aurora PostgreSQL

A seguir, você pode encontrar alguns exemplos de como chamar a função aws_lambda.invoke. A maioria dos exemplos usa a estrutura composta `aws_lambda_arn_1` que você cria em Etapa 4: usar as funções Lambda auxiliares com o cluster de bancos de dados Aurora PostgreSQL (opcional) para simplificar a passagem dos detalhes da função. Para obter um exemplo de invocação assíncrona, consulte Exemplo: invocação assíncrona (evento) de funções Lambda. Todos os outros exemplos listados usam invocação síncrona.

Para saber mais sobre os tipos de invocação do Lambda, consulte Invocação de funções Lambda no Guia do desenvolvedor do AWS Lambda. Para obter mais informações sobre o `aws_lambda_arn_1`, consulte aws_commons.create_lambda_function_arn.

Lista de exemplos

- Exemplo: invocação síncrona (RequestResponse) de funções Lambda
- Exemplo: invocação assíncrona (evento) de funções Lambda
- Exemplo: capturar o log de execução do Lambda em uma resposta de função

- Exemplo: incluir o contexto do cliente em uma função Lambda
- Exemplo: invocar uma versão específica de uma função Lambda

Exemplo: invocação síncrona (RequestResponse) de funções Lambda

A seguir estão dois exemplos de uma invocação síncrona de função Lambda. Os resultados dessas chamadas de funções `aws_lambda.invoke` são os mesmos.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

Os parâmetros são descritos da seguinte forma:

- `:'aws_lambda_arn_1'` – Este parâmetro identifica a estrutura composta criada em Etapa 4: usar as funções Lambda auxiliares com o cluster de bancos de dados Aurora PostgreSQL (opcional), com a função auxiliar `aws_commons.create_lambda_function_arn`. Você também pode criar essa estrutura em linha dentro da sua chamada `aws_lambda.invoke` da seguinte forma:

  ```
  SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
   'aws-region'),
  '{"body": "Hello from Postgres!"}'::json
  );
  ```

- `'{"body": "Hello from PostgreSQL!"}'::json` – A carga útil JSON para passar para a função Lambda.
- `'RequestResponse'` – O Lambda tipo de invocação.

Exemplo: invocação assíncrona (evento) de funções Lambda

Segue-se um exemplo de uma invocação de função Lambda assíncrona. O tipo de invocação `Event` agenda a invocação de função Lambda com a carga de entrada específica e retorna imediatamente. Use o tipo de invocação de `Event` em determinados fluxos de trabalho que não dependem dos resultados da função Lambda.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

Exemplo: capturar o log de execução do Lambda em uma resposta de função

Você pode incluir os últimos 4 KB do log de execução na resposta da função usando o parâmetro `log_type` em sua chamada de função `aws_lambda.invoke`. Por padrão, esse parâmetro é definido como `None`, mas você pode especificar `Tail` para capturar os resultados do log de execução do Lambda na resposta, conforme mostrado a seguir.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Defina o parâmetro aws_lambda.invoke da função `log_type` para `Tail` incluir o log de execução na resposta. O valor padrão para o parâmetro `log_type` é `None`.

O `log_result` que é retornado é uma string codificada de `base64`. Você pode decodificar os conteúdos usando uma combinação das funções PostgreSQL `decode` e `convert_from`.

Para obter mais informações sobre o `log_type`, consulte aws_lambda.invoke.

Exemplo: incluir o contexto do cliente em uma função Lambda

A função `aws_lambda.invoke` tem um parâmetro `context` que você pode usar para passar informações separadas da carga útil, como mostrado a seguir.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Para incluir o contexto do cliente, use um objeto JSON para o parâmetro aws_lambda.invoke da função`context`.

Para obter mais informações sobre os parâmetros do `context`, consulte a referência aws_lambda.invoke.

Exemplo: invocar uma versão específica de uma função Lambda

Você pode especificar uma determinada versão de uma função Lambda incluindo o parâmetro `qualifier` com a chamada `aws_lambda.invoke`. A seguir, você pode encontrar um exemplo que faz isso usando '*custom_version*' como um alias para a versão.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

Você também pode fornecer um qualificador de função Lambda com as informações de nome da função da forma a seguir.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

Para obter mais informações sobre `qualifier` e outros parâmetros, consulte a referência aws_lambda.invoke.

Mensagens de erro da função Lambda

Na lista a seguir, você pode encontrar informações sobre mensagens de erro, com possíveis causas e soluções.

- Problemas de configuração da VPC

  Problemas de configuração da VPC podem gerar as seguintes mensagens de erro ao tentar se conectar:

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  Uma causa comum para esse erro é o grupo de segurança da VPC configurado incorretamente. É necessário ter uma regra de saída para TCP aberta na porta 443 para que o grupo de segurança de sua VPC possa se conectar à VPC do Lambda.
- Falta de permissões necessárias para invocar funções do Lambda

  Se você vir uma das seguintes mensagens de erro, isso significa que o usuário (função) que está invocando a função não tem as permissões adequadas.

```
ERROR:  permission denied for schema aws_lambda
```

```
ERROR:  permission denied for function invoke
```

Um usuário (função) deve receber concessões específicas para invocar funções do Lambda. Para obter mais informações, consulte Etapa 6: Conceder a outros usuários permissão para invocar funções do Lambda.

- Tratamento inadequado de erros em suas funções do Lambda

  Se uma função Lambda lança uma exceção durante o processamento da solicitação, `aws_lambda.invoke` terá um erro do PostgreSQL, como o seguinte.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
ERROR:  lambda invocation failed
DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
 "Unhandled", details: "<Error details string>".
```

  Certifique-se de lidar com erros nas funções Lambda ou na aplicação PostgreSQL.

Função do AWS Lambda e referência de parâmetros

Veja a seguir a referência para as funções e os parâmetros a serem usados para invocar o Lambda com o Aurora PostgreSQL .

Funções e parâmetros

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- Parâmetros aws_lambda

aws_lambda.invoke

Executa uma função do Lambda para um cluster de bancos de dados Aurora PostgreSQL .

Para obter mais detalhes sobre invocar funções Lambda, consulte também Invocar no Guia do desenvolvedor do AWS Lambda.

Sintaxe

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
```

```
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

Parâmetros de entrada

function_name

O nome de identificação da função Lambda. O valor pode ser o nome da função, um ARN ou um ARN parcial. Para obter uma lista de formatos possíveis, consulte Formatos de nome da função Lambda no Guia do desenvolvedor do AWS Lambda.

payload

A entrada para a função Lambda. O formato pode ser JSON ou JSONB. Para obter mais informações, consulte Tipos JSON na documentação do PostgreSQL.

região

(Opcional) A Lambda Região da função. Por padrão, o Aurora resolve a região da AWS a partir do ARN completo na `function_name` ou usa a região da instância de banco de dados Aurora PostgreSQL. Se este valor de região entrar em conflito com o fornecido no ARN de `function_name`, um erro será gerado.

invocation_type

O tipo de invocação da função do Lambda. O valor diferencia letras maiúsculas de minúsculas. Os valores possíveis incluem o seguinte:

- `RequestResponse` – O padrão. Esse tipo de invocação para uma função Lambda é síncrona e retorna uma carga útil de resposta no resultado. Use o tipo de invocação de

`RequestResponse` quando seu fluxo de trabalho depende de receber o resultado da função Lambda imediatamente.

- `Event` – Esse tipo de invocação para uma função Lambda é assíncrono e retorna imediatamente sem uma carga devolvida. Use o tipo de invocação de`Event` quando não precisar de resultados da função Lambda antes de seu fluxo de trabalho prosseguir.
- `DryRun` – Este tipo de invocação testa o acesso sem executar a função Lambda.

log_type

O tipo de log Lambda a ser retornado no parâmetro de saída de `log_result`. O valor diferencia letras maiúsculas de minúsculas. Os valores possíveis incluem o seguinte:

- Cauda – O parâmetro de saída `log_result` devolvido incluirá os últimos 4 KB do log de execução.
- Nenhuma – Nenhuma informação de log Lambda é devolvida.

context

Contexto do cliente no formato JSON ou JSONB. Os campos a serem usados incluem que `custom` e `env`.

qualifier

Um qualificador que identifica a versão de uma função Lambda a ser invocada. Se esse valor entrar em conflito com um fornecido no ARN do `function_name`, gera um erro.

Parâmetros de saída

status_code

Um código de resposta de status HTTP. Para obter mais informações, consulte Elementos de resposta de invocação do Lambda no Guia do desenvolvedor do AWS Lambda.

payload

As informações devolvidas da função Lambda executada. O formato está em JSON ou JSONB.

executed_version

A versão da função Lambda executada.

log_result

As informações do log de execução devolvidas se o valor `log_type` é `Tail` quando a função Lambda for invocada. O resultado contém os últimos 4 KB do log de execução codificado em Base64.

aws_commons.create_lambda_function_arn

Cria uma estrutura `aws_commons._lambda_function_arn_1` para manter as informações do nome da função Lambda. Você pode usar os resultados da função `aws_commons.create_lambda_function_arn` no parâmetro `function_name` da função aws_lambda.invoke.

Sintaxe

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

Parâmetros de entrada

function_name

Uma string de texto necessária que contém o nome da função Lambda. O valor pode ser um nome de função, um ARN parcial ou um ARN completo.

região

Uma string de texto opcional que contém a região da AWS na qual a função Lambda está inserida. Para obter uma listagem de nomes de regiões da e os valores associados, consulte Regiões e zonas de disponibilidade.

Parâmetros aws_lambda

Na tabela, é possível encontrar parâmetros associados à função do `aws_lambda`.

Parâmetro	Descrição
`aws_lambda.connect_timeout_ms`	É um parâmetro dinâmico e define o tempo máximo de espera durante a conexão com o AWS Lambda. O valor padrão é `1000`. Os valores permitidos para esse parâmetro são de 1 a 900000.
`aws_lambda.request_timeout_ms`	É um parâmetro dinâmico e define o tempo máximo de espera enquanto aguarda a resposta do AWS Lambda. O valor padrão é `3000`. Os valores permitidos para esse parâmetro são de 1 a 900000.
`aws_lambda.endpoint_override`	Especifica o endpoint que pode ser usado para se conectar ao AWS Lambda. Uma string vazia seleciona o endpoint padrão do AWS Lambda para a região. É necessário reiniciar o banco de dados para esse parâmetro ter efeito.

Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs

É possível configurar seu cluster de bancos de dados do Aurora PostgreSQL para exportar dados de log para o Amazon CloudWatch Logs regularmente. Quando você faz isso, os eventos do log do PostgreSQL do seu cluster de banco de dados do Aurora PostgreSQL são automaticamente publicados no Amazon CloudWatch, na forma de logs do Amazon CloudWatch. No CloudWatch, você pode encontrar os dados de logs exportados em um grupo de logs para o seu cluster de banco de dados do Aurora PostgreSQL. O grupo de logs contém um ou mais fluxos de logs que contêm os eventos do log do PostgreSQL de cada instância no cluster.

A publicação dos logs no CloudWatch Logs permite que você mantenha os registros de log do PostgreSQL do seu cluster em armazenamento altamente durável. Com os dados de log disponíveis no CloudWatch Logs, você pode avaliar e melhorar as operações do seu cluster. Você também pode usar o CloudWatch para criar alarmes e visualizar métricas. Para saber mais, consulte Monitorar eventos de log no Amazon CloudWatch.

> **Note**
>
> A publicação de logs do PostgreSQL no CloudWatch Logs consome armazenamento, e você incorre em cobranças por esse armazenamento. Se não precisar mais de algum log do CloudWatch, lembre-se de excluí-lo.

Desativar a opção de log de exportação para um cluster de banco de dados do Aurora PostgreSQL existente não afetará nenhum dado que já esteja armazenado no CloudWatch Logs. Os logs existentes permanecem disponíveis no CloudWatch Logs com base nas suas configurações de retenção de logs. Para saber mais sobre o CloudWatch Logs, consulte O que é o Amazon CloudWatch Logs?

O Aurora PostgreSQL oferece suporte à publicação de logs no CloudWatch Logs para as seguintes versões.

- 16.1 e todas as versões posteriores
- Versão 15.2 e versões 15 posteriores
- 14.3 e versões 14 posteriores
- versões 13.3 e posteriores à 13
- 12.8 e versões 12 posteriores
- versões 11.9 e posteriores à 11

Para ter informações sobre como ativar a opção de publicar logs no CloudWatch Logs, monitorar eventos de log no CloudWatch Logs e analisar logs usando o CloudWatch Logs Insights, consulte os tópicos a seguir.

Tópicos

- Ativar a opção de publicação de logs no Amazon CloudWatch
- Monitorar eventos de log no Amazon CloudWatch
- Analisar logs do PostgreSQL utilizando o CloudWatch Logs Insights

Ativar a opção de publicação de logs no Amazon CloudWatch

Para publicar um log do PostgreSQL do seu cluster de banco de dados do Aurora PostgreSQL no CloudWatch Logs, escolha a opção Log export (Exportação de logs) para o cluster. Você

pode escolher a configuração Log export (Exportação de logs) ao criar um cluster de banco de dados do Aurora PostgreSQL. Ou você pode modificar o cluster posteriormente. Quando você modifica um cluster existente, seus logs do PostgreSQL de cada instância são publicados no cluster do CloudWatch a partir desse momento. Para o Aurora PostgreSQL, o log do PostgreSQL (`postgresql.log`) é o único log que é publicado no Amazon CloudWatch.

Você pode usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para ativar o recurso de exportação de logs para seu cluster de banco de dados do Aurora PostgreSQL.

Console

Você escolhe a opção de exportação de logs para começar a publicar os logs do PostgreSQL do seu cluster de banco de dados do Aurora PostgreSQL no CloudWatch Logs.

Como ativar o recurso de exportação de logs pelo console

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione o cluster de banco de dados do Aurora PostgreSQL cujos dados de log você deseja publicar no CloudWatch Logs.
4. Selecione Modify.
5. Na seção Log exports (Exportações de log), selecione PostgreSQL log (Log do PostgreSQL).
6. Escolha Continue (Continuar) e Modify cluster (Modificar cluster) na página de resumo.

AWS CLI

Você pode ativar a opção de exportação de logs para começar a publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs com a AWS CLI. Para isso, execute o comando modify-db-cluster da AWS CLI com as seguintes opções:

- `--db-cluster-identifier`—O identificador de cluster de banco de dados.
- `--cloudwatch-logs-export-configuration` – a definição de configuração para os tipos de log a serem definidos para exportação no CloudWatch Logs para cluster de banco de dados.

Também é possível publicar logs do Aurora PostgreSQL executando um dos seguintes comandos da AWS CLI:

- create-db-cluster

- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Execute um destes comandos da AWS CLI com as seguintes opções:

- `--db-cluster-identifier`—O identificador de cluster de banco de dados.
- `--engine` — o mecanismo de banco de dados.
- `--enable-cloudwatch-logs-exports` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Outras opções podem ser exigidas dependendo do comando da AWS CLI que você executa.

Example

O comando a seguir cria um cluster de banco de dados Aurora PostgreSQL para publicar arquivos de log no CloudWatch Logs.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --engine aurora-postgresql \
    --enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --engine aurora-postgresql ^
    --enable-cloudwatch-logs-exports postgresql
```

Example

O comando a seguir altera um cluster existente do banco de dados Aurora PostgreSQL para publicar arquivos de log no CloudWatch Logs. O valor `--cloudwatch-logs-export-configuration` é um objeto JSON. A chave para esse objeto é `EnableLogTypes`, e seu valor é `postgresql` e `instance`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --cloudwatch-logs-export-configuration '{\"EnableLogTypes\":[\"postgresql\",
\"instance\"]}'
```

> **Note**
>
> Ao utilizar o prompt de comando do Windows, certifique-se de fazer o escape das aspas duplas (") no código JSON, prefixando-as com uma barra invertida (\).

Example

O exemplo a seguir modifica um cluster de banco de dados Aurora PostgreSQL existente para desativar a publicação de arquivos de log no CloudWatch Logs. O valor `--cloudwatch-logs-export-configuration` é um objeto JSON. A chave para esse objeto é `DisableLogTypes`, e seu valor é `postgresql` e `instance`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":
["postgresql","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"postgresql\",
\"instance\"]}"
```

> **Note**
>
> Ao usar o prompt de comando do Windows, você deve fazer o escape das aspas duplas (") no código JSON, prefixando-as com uma barra invertida (\).

API do RDS

Você pode ativar a opção de exportação de logs para começar a publicar logs do Aurora PostgreSQL com a API do RDS. Para isso, execute a operação ModifyDBCluster com as seguintes opções:

- `DBClusterIdentifier` – o identificador de cluster de banco de dados.
- `CloudwatchLogsExportConfiguration` – a definição de configuração para os tipos de log a serem habilitados para exportação no CloudWatch Logs para o cluster de banco de dados.

Também é possível publicar logs do Aurora PostgreSQL com a API do RDS executando uma das seguintes operações da API do RDS:

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Execute a ação da API do RDS com os seguintes parâmetros:

- `DBClusterIdentifier`—O identificador de cluster de banco de dados.
- `Engine` — o mecanismo de banco de dados.
- `EnableCloudwatchLogsExports` — a definição de configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para cluster de banco de dados.

Outros parâmetros podem ser exigidos dependendo do comando da AWS CLI que você executa.

Monitorar eventos de log no Amazon CloudWatch

Com os eventos de logs do Aurora PostgreSQL publicados e disponíveis no Amazon CloudWatch Logs, você pode visualizar e monitorar os eventos usando o Amazon CloudWatch. Para obter mais

informações sobre monitoramento, consulte Visualizar dados de log enviados para o CloudWatch Logs.

Quando você ativa as exportações de logs, um novo grupo de logs é criado automaticamente usando o prefixo `/aws/rds/cluster/` com o nome do seu Aurora PostgreSQL e o tipo de log, como no padrão a seguir.

```
/aws/rds/cluster/your-cluster-name/postgresql
```

Como exemplo, suponha que um cluster de banco de dados do Aurora PostgreSQL chamado `docs-lab-apg-small` exporta seu log para o Amazon CloudWatch Logs. O nome do grupo de logs no Amazon CloudWatch é mostrado a seguir.

```
/aws/rds/cluster/docs-lab-apg-small/postgresql
```

Se já houver um grupo de logs com o nome especificado, o Aurora usará esse grupo para exportar dados de log para o cluster de banco de dados Aurora. Cada instância de banco de dados no cluster de banco de dados do Aurora PostgreSQL faz upload de seu log do PostgreSQL no grupo de logs como um fluxo de logs distinto. Você pode examinar o grupo de logs e seus fluxos de logs usando as várias ferramentas gráficas e analíticas disponíveis no Amazon CloudWatch.

Por exemplo, é possível pesquisar informações nos eventos de logs do seu cluster de banco de dados do Aurora PostgreSQL e filtrar os eventos usando o console do CloudWatch Logs, a AWS CLI ou a API do CloudWatch Logs. Para obter mais informações, consulte Pesquisar e filtrar dados de logs no Guia do usuário do Amazon CloudWatch Logs.

Por padrão, novos grupos de logs são criados usando a opção Never expire (Nunca expirar) para o período de retenção. Você pode usar o console do CloudWatch Logs, a AWS CLI ou a API do CloudWatch Logs para alterar o período de retenção de log. Para saber mais, consulte Alterar a retenção de dados de log no CloudWatch Logs no Guia do usuário do Amazon CloudWatch Logs.

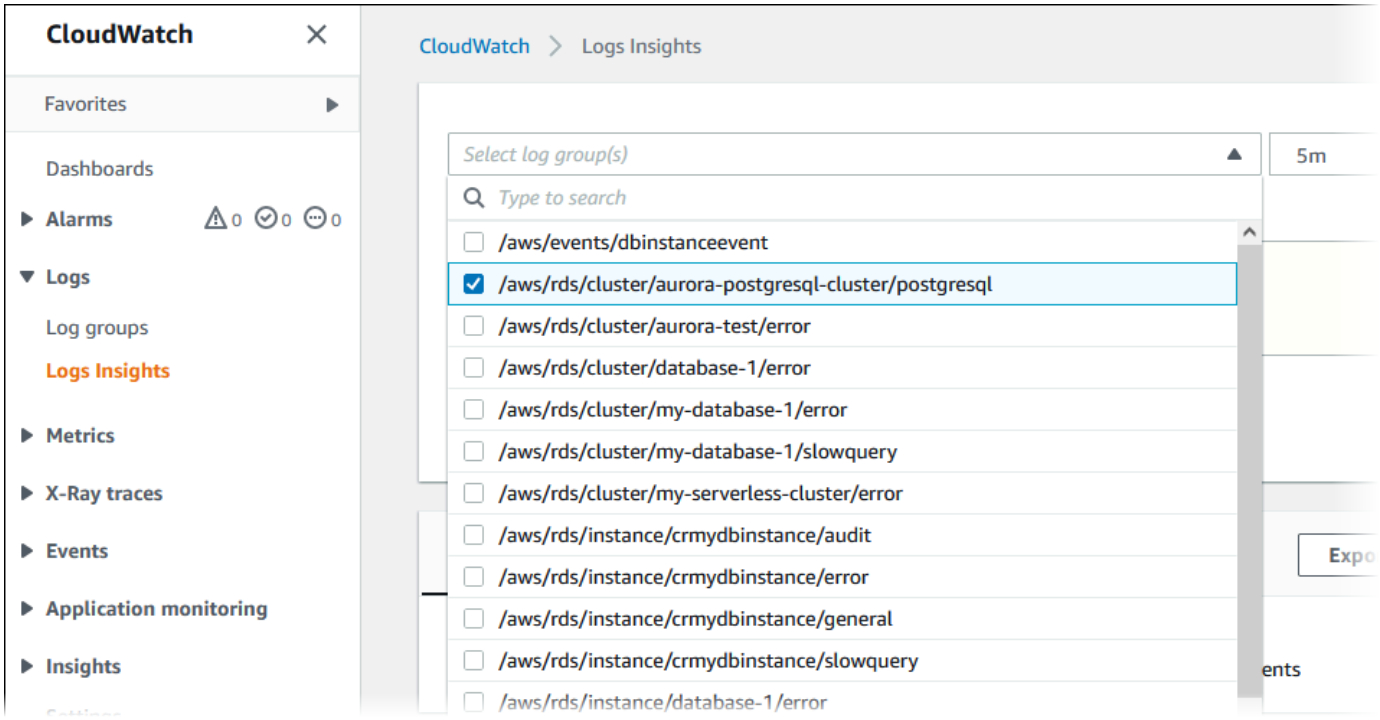
> **Tip**
>
> Você pode usar a configuração automática, como o AWS CloudFormation, para criar grupos de log com períodos de retenção de log predefinidos, filtros de métricas e permissões de acesso.

Analisar logs do PostgreSQL utilizando o CloudWatch Logs Insights

Com os logs do PostgreSQL do seu cluster de banco de dados do Aurora PostgreSQL publicados no CloudWatch Logs, você pode usar o CloudWatch Logs Insights para pesquisar e analisar dados de log de modo interativo no Amazon CloudWatch Logs. O CloudWatch Logs Insights inclui uma linguagem de consulta, exemplos de consultas e outras ferramentas para analisar dados de log com o intuito de identificar possíveis problemas e verificar as correções. Para saber mais, consulte Analisar dados de log no CloudWatch Logs Insights no Guia do usuário do Amazon CloudWatch Logs.

Como analisar logs do PostgreSQL com o CloudWatch Logs Insights

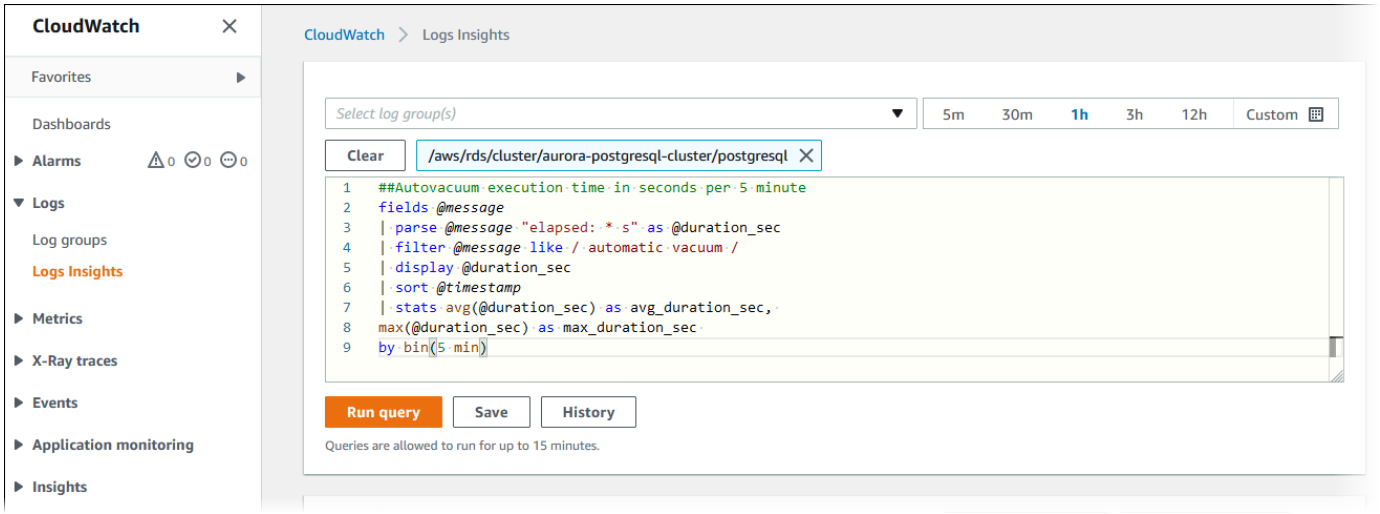
1. Abra o console do CloudWatch, em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, abra Logs e escolha Log insights.
3. Em Select log group(s) (Selecionar grupos de logs), selecione o grupo de logs do seu cluster de banco de dados do Aurora PostgreSQL.



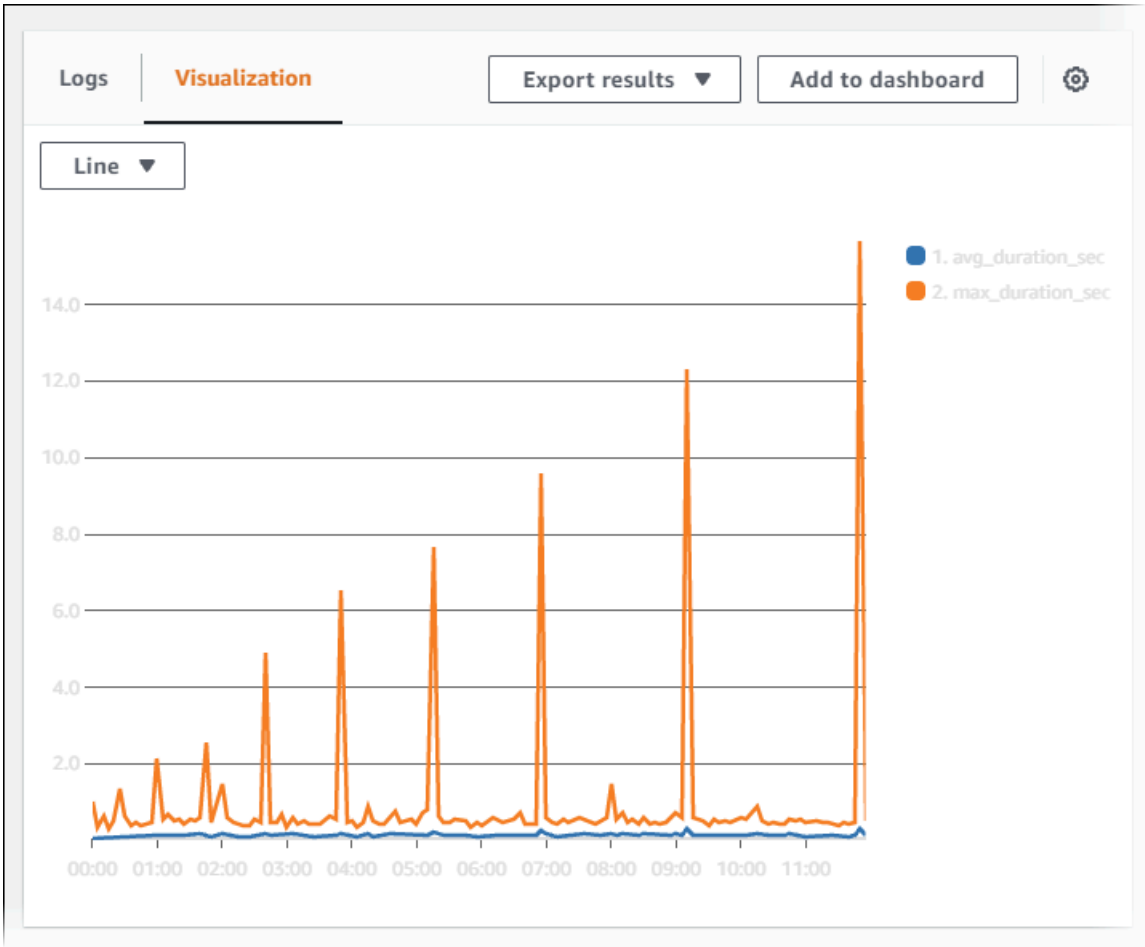
4. No editor de consultas, exclua a consulta atualmente visível, insira o seguinte e escolha Run query (Executar consulta).

```
##Autovacuum execution time in seconds per 5 minute
fields @message
| parse @message "elapsed: * s" as @duration_sec
```

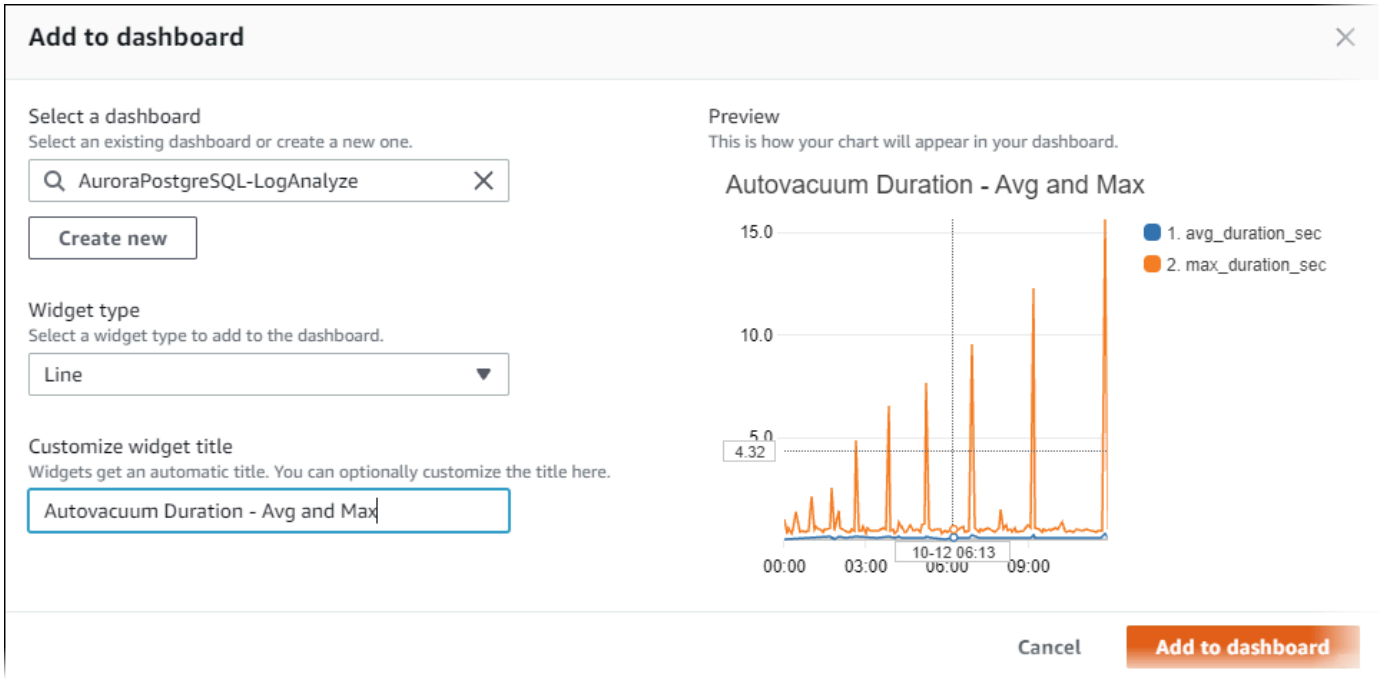
```
| filter @message like / automatic vacuum /
| display @duration_sec
| sort @timestamp
| stats avg(@duration_sec) as avg_duration_sec,
max(@duration_sec) as max_duration_sec
by bin(5 min)
```



5. Escolha a guia Visualization (Visualização).



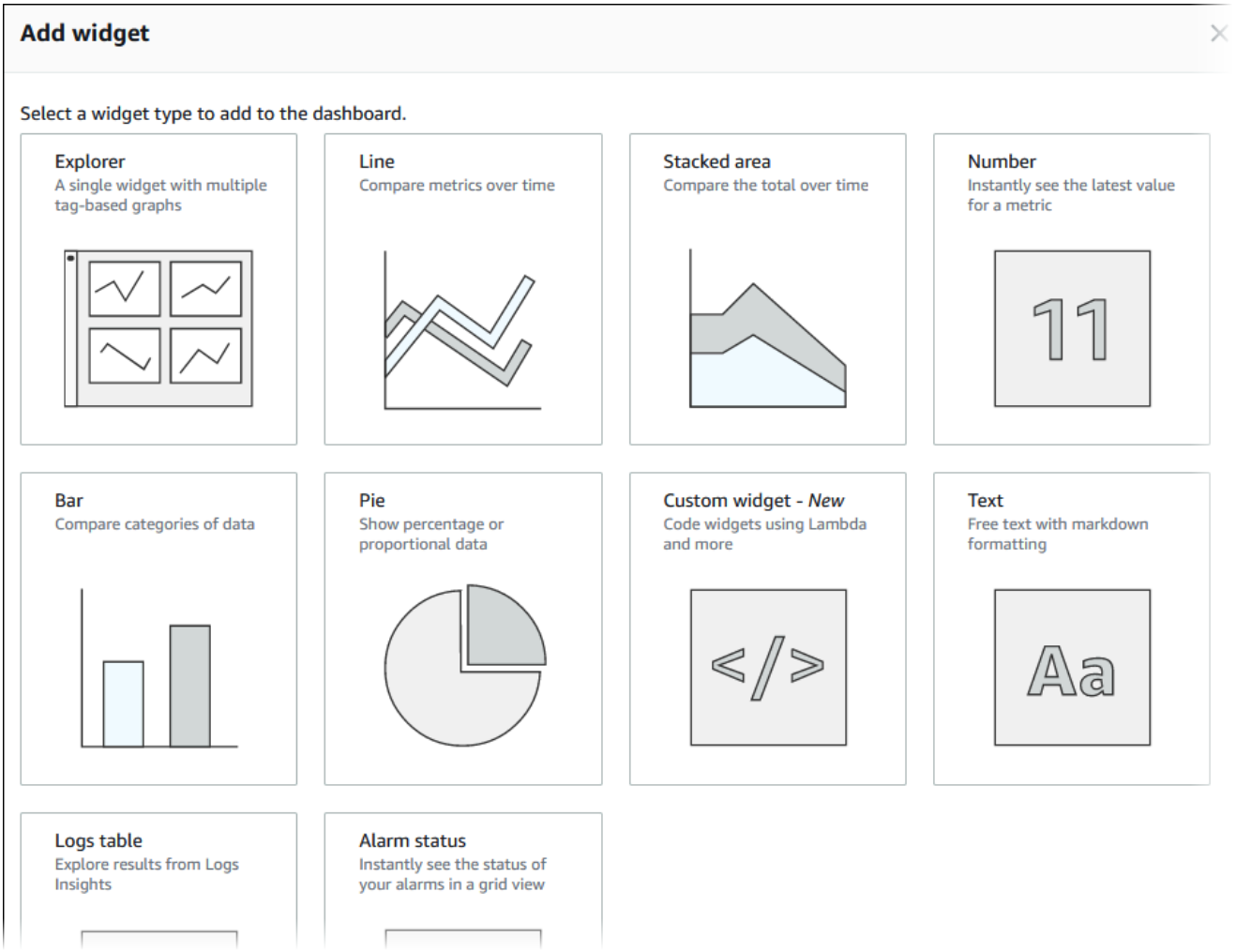
6. Escolha Add to dashboard (Adicionar ao painel).
7. Em Select a dashboard (Selecione um painel), selecione um painel ou insira um nome para criar um novo painel.
8. Em Widget type (Tipo de widget), escolha um tipo de widget para a sua visualização.



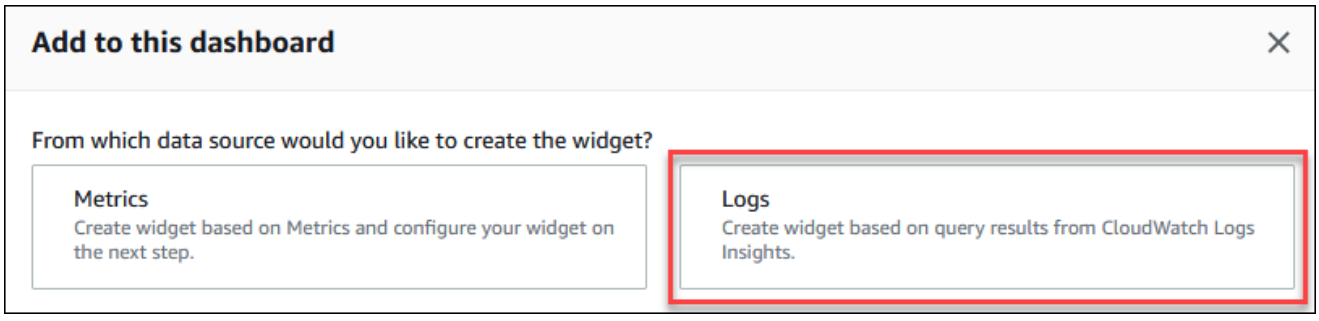
9. (Opcional) Adicione mais widgets com base nos resultados da sua consulta de log.

   a. Escolha Add widget (Adicionar widget).

   b. Escolha um tipo de widget, como Line (Linha).



c. Na janela Add to this dashboard (Adicionar a este painel), escolha Logs.

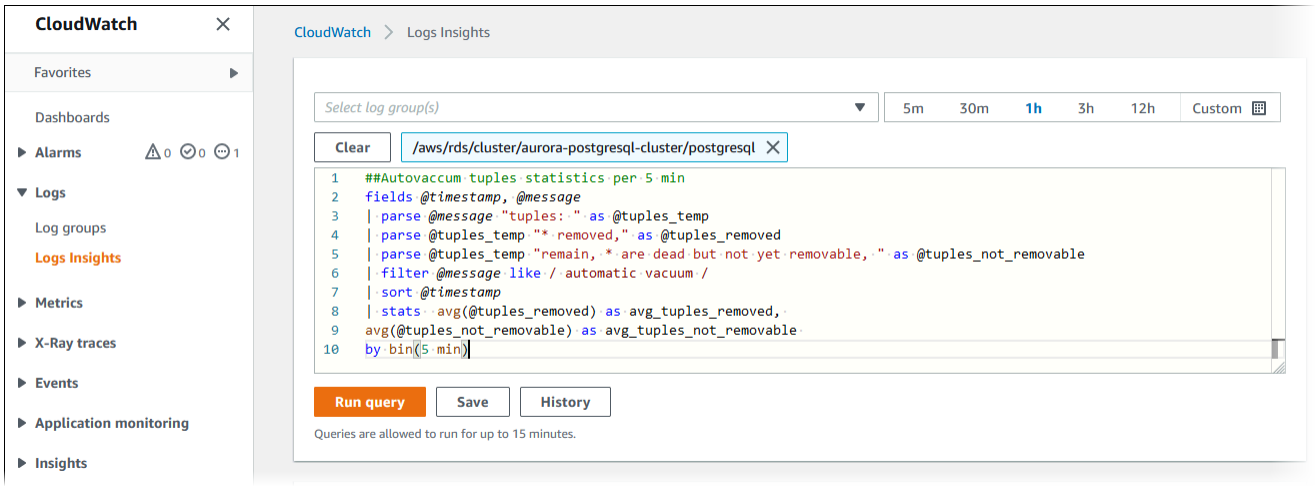


d. Em Select log group(s) (Selecionar grupos de logs), selecione o grupo de logs do seu cluster de banco de dados.

e. No editor de consultas, exclua a consulta atualmente visível, insira o seguinte e escolha Run query (Executar consulta).

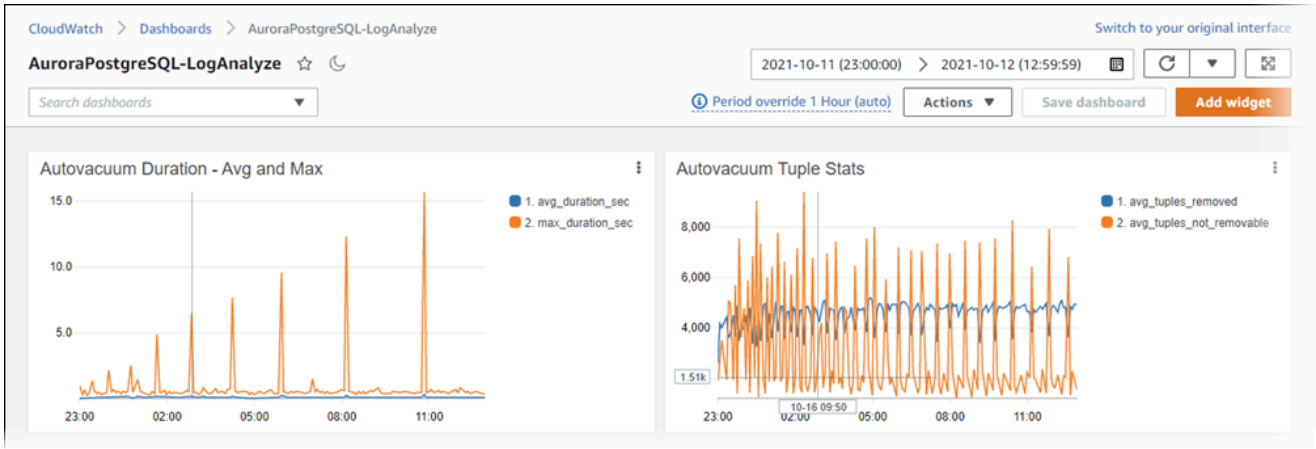
```
##Autovacuum tuples statistics per 5 min
fields @timestamp, @message
```

```
| parse @message "tuples: " as @tuples_temp
| parse @tuples_temp "* removed," as @tuples_removed
| parse @tuples_temp "remain, * are dead but not yet removable, " as
 @tuples_not_removable
| filter @message like / automatic vacuum /
| sort @timestamp
| stats  avg(@tuples_removed) as avg_tuples_removed,
avg(@tuples_not_removable) as avg_tuples_not_removable
by bin(5 min)
```



f. Selecione Create widget (Criar widget).

Seu painel deve ser semelhante à seguinte imagem.



Monitorar planos de execução de consultas e pico de memória do Aurora PostgreSQL

É possível monitorar os planos de execução de consultas na instância de banco de dados do Aurora PostgreSQL para detectar os planos de execução que contribuem para a carga atual do banco de dados e monitorar as estatísticas de performance dos planos de execução ao longo do tempo usando o parâmetro `aurora_compute_plan_id`. Sempre que uma consulta é realizada, o plano de execução usado pela consulta recebe um identificador e o mesmo identificador é usado pelas execuções subsequentes do mesmo plano.

O `aurora_compute_plan_id` é `OFF` por padrão no grupo de parâmetros de banco de dados das versões 14.10, 15.5 e posterior do Aurora PostgreSQL. Para atribuir um identificador de plano, defina `aurora_compute_plan_id` como `ON` no grupo de parâmetros.

Esse identificador de plano é usado em vários utilitários que atendem a uma finalidade diferente.

É possível monitorar o pico de uso da memória de consultas na instância de banco de dados para detectar consultas que contribuem para o alto uso de memória do banco de dados nas seguintes versões:

- 16.3 e todas as versões posteriores
- 15.7 e versões posteriores
- 14.12 e versões posteriores

Sempre que uma consulta é executada, o pico de memória usado pela consulta é monitorado. As consultas geralmente são executadas várias vezes; os valores de uso de memória médio, mínimo e máximo em todas as execuções podem ser visualizados para cada consulta.

Tópicos

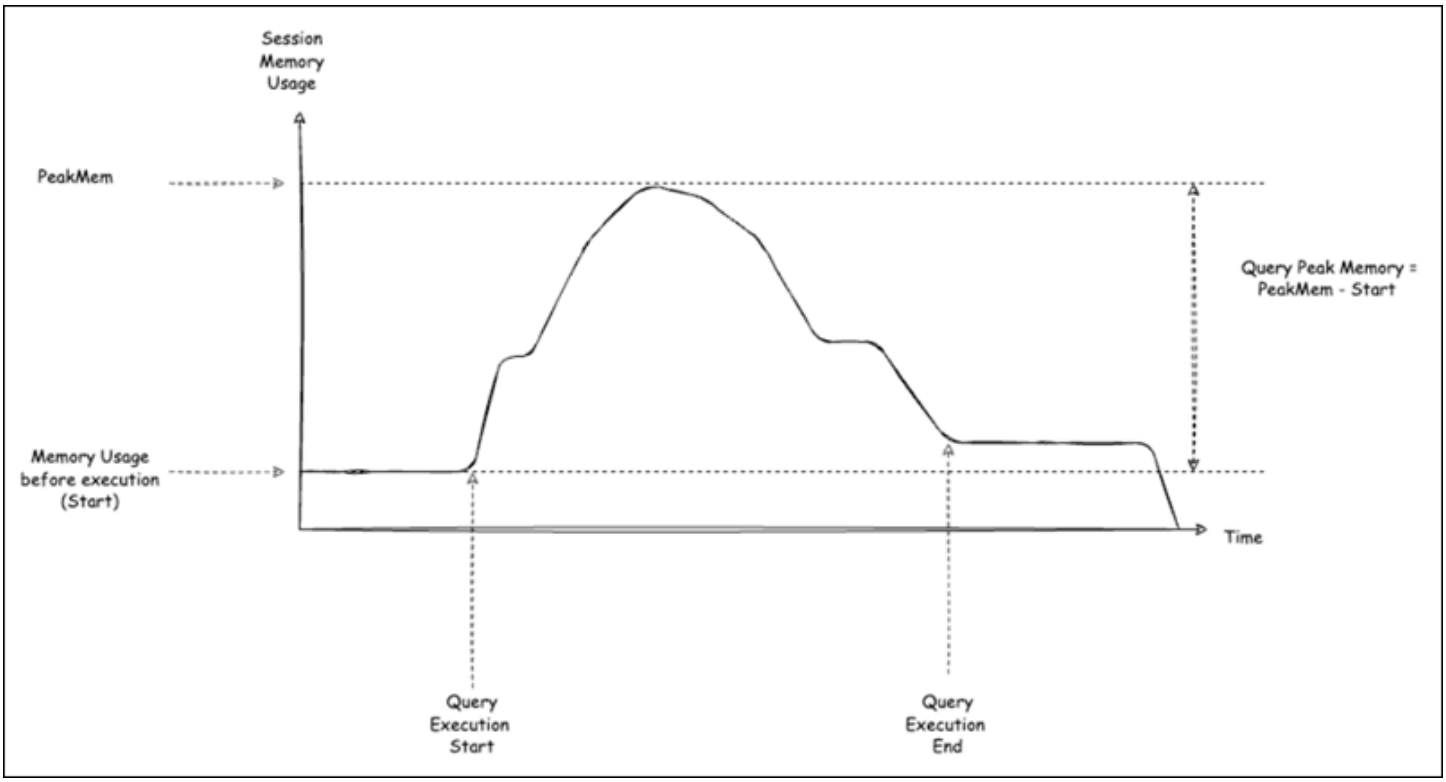
- Acessar planos de execução de consultas e pico de memória usando funções do Aurora
- Referência de parâmetros de planos de execução de consultas do Aurora PostgreSQL

Acessar planos de execução de consultas e pico de memória usando funções do Aurora

Com `aurora_compute_plan_id`, é possível acessar os planos de execução usando as seguintes funções:

- aurora_stat_activity
- aurora_stat_plans

A memória de pico da consulta não inclui a memória alocada antes do início do processamento da consulta. O pico de uso da memória é monitorado e relatado separadamente para as fases de planejamento e execução de cada consulta.



É possível acessar as estatísticas de pico de memória de consultas usando as seguintes funções:

- aurora_stat_statements
- aurora_stat_plans

Para ter mais informações sobre essas funções, consulte Referência de funções do Aurora PostgreSQL.

Referência de parâmetros de planos de execução de consultas do Aurora PostgreSQL

É possível monitorar os planos de execução de consultas usando os parâmetros abaixo em um grupo de parâmetros de banco de dados.

Parâmetros

- aurora_compute_plan_id
- aurora_stat_plans.minutes_until_recapture
- aurora_stat_plans.calls_until_recapture
- aurora_stat_plans.with_costs
- aurora_stat_plans.with_analyze
- aurora_stat_plans.with_timing
- aurora_stat_plans.with_buffers
- aurora_stat_plans.with_wal
- aurora_stat_plans.with_triggers

> **Note**
>
> A configuração dos parâmetros `aurora_stat_plans.with_*` entra em vigor somente para planos recém-capturados.

aurora_compute_plan_id

O `aurora_compute_plan_id` é um parâmetro de configuração que controla se um identificador de plano é atribuído durante a execução da consulta.

Padrão	Valores permitidos	Descrição
desligar	0 (desativado)	Defina como `off` para evitar que um identificador de plano seja atribuído.
	1 (ativado)	Defina como on para atribuir um identificador de plano.

aurora_stat_plans.minutes_until_recapture

O número de minutos que faltam antes que um plano seja recapturado. O padrão é 0, o que desabilitará a recaptura de um plano. Quando o limite de `aurora_stat_plans.calls_until_recapture` é ultrapassado, o plano será recuperado.

Padrão	Valores permitidos	Descrição
0	0-1073741823	Defina o número de minutos para recuperar um plano.

aurora_stat_plans.calls_until_recapture

O número de chamadas para um plano antes que ele seja recapturado. O padrão é 0, o que desativará a recaptura de um plano após várias chamadas. Quando o limite de `aurora_stat_plans.minutes_until_recapture` é ultrapassado, o plano será recuperado.

Padrão	Valores permitidos	Descrição
0	0-1073741823	Defina o número de chamadas antes que um plano seja recapturado.

aurora_stat_plans.with_costs

Captura um plano EXPLAIN com custos estimados. Os valores permitidos são on e `off`. O padrão é on.

Padrão	Valores permitidos	Descrição
ativado	0 (desativado)	Não mostra o custo e as linhas estimados para cada nó do plano.
	1 (ativado)	Mostra o custo estimado e as linhas para cada nó do plano.

aurora_stat_plans.with_analyze

Controla o plano EXPLAIN com ANALYZE. Esse modo só é usado na primeira vez em que um plano é capturado. Os valores permitidos são `on` e `off`. O padrão é `off`.

Padrão	Valores permitidos	Descrição
desligar	0 (desativado)	Não inclui estatísticas reais de tempo de execução do plano.
	1 (ativado)	Inclui estatísticas reais de tempo de execução do plano.

aurora_stat_plans.with_timing

O tempo do plano será capturado na explicação quando ANALYZE for usado. O padrão é `on`.

Padrão	Valores permitidos	Descrição
ativado	0 (desativado)	Não inclui o tempo real de inicialização e o tempo gasto em cada nó do plano.
	1 (ativado)	Inclui o tempo real de inicialização e o tempo gasto em cada nó do plano.

aurora_stat_plans.with_buffers

As estatísticas de uso do buffer do plano será capturado na explicação quando ANALYZE for usado. O padrão é `off`.

Padrão	Valores permitidos	Descrição
desligar	0 (desativado)	Não inclui informações sobre o uso do buffer.
	1 (ativado)	Inclui informações sobre o uso do buffer.

aurora_stat_plans.with_wal

As estatísticas de uso do wal do plano será capturado na explicação quando ANALYZE for usado. O padrão é `off`.

Padrão	Valores permitidos	Descrição
desligar	0 (desativado)	Não inclui informações sobre a geração de registros do WAL.
	1 (ativado)	Inclui informações sobre a geração de registros do WAL.

aurora_stat_plans.with_triggers

As estatísticas de execução do gatilho do plano serão capturadas na explicação quando `ANALYZE` for usado. O padrão é `off`.

Padrão	Valores permitidos	Descrição
desligar	0 (desativado)	Não inclui estatísticas de execução de gatilhos.
	1 (ativado)	Inclui estatísticas de execução de gatilhos.

Gerenciar planos de execução de consultas do Aurora PostgreSQL

O gerenciamento de planos de consulta do Aurora PostgreSQL é um recurso opcional que você pode usar com seu cluster de banco de dados de edição compatível com o Amazon Aurora PostgreSQL. Esse recurso é fornecido como a extensão `apg_plan_mgmt` que você pode instalar em seu cluster de banco de dados do Aurora PostgreSQL. O gerenciamento de planos de consulta permite que você gerencie os planos de execução de consultas gerados pelo otimizador para suas aplicações SQL. A extensão `apg_plan_mgmt` AWS se baseia na funcionalidade nativa de processamento de consultas do mecanismo de banco de dados do PostgreSQL.

A seguir, você encontrará informações sobre os recursos de gerenciamento de planos de consulta do Aurora PostgreSQL, como configurá-lo e como usá-lo com o cluster de banco de dados do Aurora PostgreSQL. Antes de começar, recomendamos que você revise todas as notas de lançamento da versão específica da extensão `apg_plan_mgmt` disponível para sua versão do Aurora PostgreSQL. Para obter mais informações, consulte Aurora PostgreSQL apg_plan_mgmt extension versions (Versões da extensão apg_plan_mgmt do Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

Tópicos

- Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL
- Práticas recomendadas para gerenciamento de planos de consultas do Aurora PostgreSQL
- Gerenciamento de planos de consulta do Aurora PostgreSQL
- Capturar planos de execução do Aurora PostgreSQL
- Usar planos gerenciados do Aurora PostgreSQL
- Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans
- Melhorar os planos de consulta do Aurora PostgreSQL
- Excluir os planos de consulta do Aurora PostgreSQL
- Exportar e importar planos gerenciados do Aurora PostgreSQL
- Referência de parâmetros do gerenciamento de planos de consultas do Aurora PostgreSQL
- Referência de funções do gerenciamento de planos de consultas do Aurora PostgreSQL
- Referência da visualização apg_plan_mgmt.dba_plans da edição compatível do Aurora PostgreSQL
- Atributos avançados do gerenciamento de planos de consultas

Visão geral do gerenciamento de planos de consulta do Aurora PostgreSQL

O gerenciamento de planos de consulta do Aurora PostgreSQL foi projetado para garantir a estabilidade do plano, independentemente das alterações no banco de dados que possam causar a regressão do plano de consulta. A regressão do plano de consulta ocorre quando o otimizador seleciona um plano abaixo do ideal para determinada declaração SQL após alterações no sistema ou no banco de dados. Alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados PostgreSQL podem causar regressão do plano.

Com o gerenciamento de planos de consulta do Aurora PostgreSQL, é possível controlar como e quando planos de execução de consultas mudam. Os benefícios do gerenciamento de planos de consultas do Aurora PostgreSQL incluem o seguinte.

- Melhorar a estabilidade do plano forçando o otimizador a escolher entre um pequeno número de planos bons e conhecidos.
- Otimizar os planos de forma centralizada e distribuir globalmente os melhores planos.
- Identificar índices não utilizados e avaliar o impacto da criação ou remoção de um índice.
- Detectar automaticamente um novo plano de custo mínimo descoberto pelo otimizador.
- Experimentar novos recursos do otimizador com menos risco, pois você pode aprovar apenas as alterações de planos que melhoram a performance.

Você pode usar as ferramentas fornecidas pelo gerenciamento de planos de consulta de forma proativa para especificar o melhor plano para determinadas consultas. Ou você pode usar o gerenciamento de planos de consulta para reagir às mudanças nas circunstâncias e evitar regressões do plano. Para obter mais informações, consulte Práticas recomendadas para gerenciamento de planos de consultas do Aurora PostgreSQL.

Tópicos

- Declarações SQL compatíveis
- Limitações do gerenciamento de planos de consulta
- Terminologia do gerenciamento de planos de consulta
- Versões do gerenciamento de planos de consulta do Aurora PostgreSQL
- Ativar o gerenciamento de planos de consulta do Aurora PostgreSQL
- Atualizar o gerenciamento de planos de consultas do Aurora PostgreSQL

- Desativar o gerenciamento de planos de consulta do Aurora PostgreSQL

Declarações SQL compatíveis

O gerenciamento de planos de consulta é compatível com os tipos de declaração SQL a seguir.

- Qualquer declaração SELECT, INSERT, UPDATE ou DELETE, independentemente da complexidade.
- Instruções preparadas. Para obter mais informações, consulte PREPARE na documentação do PostgreSQL.
- Declarações dinâmicas, como as executadas no modo imediato. Para obter mais informações, consulte Dynamic SQL (SQL dinâmico) e EXECUTE IMMEDIATE na documentação do PostgreSQL.
- Comandos e declarações SQL incorporados. Para obter mais informações, consulte Embedded SQL Commands (Comandos SQL incorporados) na documentação do PostgreSQL.
- Declarações dentro de funções nomeadas. Para obter mais informações, consulte CREATE FUNCTION na documentação do PostgreSQL.
- Declarações com tabelas temporárias.
- Declarações em procedimentos e bloqueios de DO.

Você pode usar o gerenciamento de planos de consulta com `EXPLAIN` no modo manual para capturar um plano sem realmente executá-lo. Para obter mais informações, consulte Analisar o plano selecionado pelo otimizador. Para saber mais sobre os modos de gerenciamento de planos de consulta (manual, automático), consulte Capturar planos de execução do Aurora PostgreSQL.

O gerenciamento de planos de consulta do Aurora PostgreSQL é compatível com todos os recursos da linguagem PostgreSQL, inclusive tabelas particionadas, herança, segurança em nível de linha e Common Table Expressions (CTEs, Expressões de tabelas comuns) recursivas. Para saber mais sobre esses recursos da linguagem PostgreSQL, consulte Table Partitioning (Particionamento de tabelas), Row Security Policies (Políticas de segurança de linha) e WITH Queries (Common Table Expressions) [Consultas WITH (expressões de tabela comuns)], bem como outros tópicos na documentação do PostgreSQL.

Para obter informações sobre diferentes versões do recurso de gerenciamento de planos de consulta do Aurora PostgreSQL, consulte Aurora PostgreSQL apg_plan_mgmt extension versions (Versões da

extensão apg_plan_mgmt do Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

Limitações do gerenciamento de planos de consulta

A versão atual do gerenciamento de planos de consulta do Aurora PostgreSQL tem as limitações a seguir.

- Os planos não são capturados para declarações que fazem referência às relações do sistema: declarações que fazem referência às relações do sistema, como `pg_class`, não são capturadas. Isso ocorre naturalmente, para evitar que um grande número de planos gerados pelo sistema que são usados internamente seja capturado. Isso também se aplica a tabelas do sistema dentro de visualizações.
- Pode ser necessária uma classe de instância de banco de dados maior para seu cluster de banco de dados do Aurora PostgreSQL: dependendo da workload, o gerenciamento de planos de consulta pode precisar de uma classe de instância de banco de dados que tenha mais de duas vCPUs. O número de `max_worker_processes` é limitado pelo tamanho da classe da instância de banco de dados. O número de `max_worker_processes` fornecidos por uma classe de instância de banco de dados de duas vCPU (db.t3.medium, por exemplo) pode não ser suficiente para uma determinada workload. Recomendamos que você selecione uma classe de instância de banco de dados com mais de duas vCPUs para seu cluster de banco de dados do Aurora PostgreSQL se você usar gerenciamento de planos de consulta.

  Quando a classe da instância de banco de dados não for compatível com a workload, o gerenciamento de planos de consulta gerará uma mensagem de erro como a seguinte.

  ```
  WARNING: could not register plan insert background process
  HINT: You may need to increase max_worker_processes.
  ```

  Nesse caso, você deve aumentar a escala verticalmente de seu cluster de banco de dados do Aurora PostgreSQL para um tamanho de classe de instância de banco de dados com mais memória. Para obter mais informações, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.
- Os planos já armazenados nas sessões não são afetados: o gerenciamento do plano de consulta é uma forma de influenciar os planos de consulta sem alterar o código da aplicação. No entanto, quando um plano genérico já estiver armazenado em uma sessão existente e você quiser alterar o respectivo plano de consulta, primeiro defina `plan_cache_mode` como `force_custom_plan` no grupo de parâmetros do cluster de banco de dados.

- `queryid` em `apg_plan_mgmt.dba_plans` e `pg_stat_statements` pode divergir quando:
  - Os objetos são descartados e recriados após serem armazenados em apg_plan_mgmt.dba_plans.
  - A tabela `apg_plan_mgmt.plans` é importada de outro cluster.

Para obter informações sobre diferentes versões do recurso de gerenciamento de planos de consulta do Aurora PostgreSQL, consulte Aurora PostgreSQL apg_plan_mgmt extension versions (Versões da extensão apg_plan_mgmt do Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

Terminologia do gerenciamento de planos de consulta

Os seguintes termos são utilizados neste tópico:

declaração gerenciada

Uma declaração SQL capturada pelo otimizador no gerenciamento de planos de consulta. Uma declaração gerenciada tem um ou mais planos de execução de consultas armazenados na visualização `apg_plan_mgmt.dba_plans`.

linha de base do plano

O conjunto de planos aprovados para determinada declaração gerenciada. Ou seja, todos os planos para a declaração gerenciada que têm “Aprovado” em sua coluna `status` na visualização `dba_plan`.

histórico de planos

O conjunto de todos os planos capturados para determinada declaração gerenciada. O histórico de planos contém todos os planos capturados para a declaração, independentemente do status.

regressão de planos de consultas

O caso em que o otimizador seleciona um plano abaixo do ideal do que o anterior a determinada alteração no ambiente do banco de dados, como uma nova versão do PostgreSQL ou alterações nas estatísticas.

Versões do gerenciamento de planos de consulta do Aurora PostgreSQL

O gerenciamento de planos de consulta é compatível com todas as versões do Aurora PostgreSQL atualmente disponíveis. Para obter informações, consulte a lista de Amazon Aurora PostgreSQL

updates (Atualizações do Amazon Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

A funcionalidade de gerenciamento de planos de consulta é adicionada ao seu cluster de banco de dados do Aurora PostgreSQL quando você instala a extensão `apg_plan_mgmt`. Versões diferentes do Aurora PostgreSQL são compatíveis com diferentes versões da extensão `apg_plan_mgmt`. Recomendamos que você atualize a extensão de gerenciamento de planos de consulta para a versão mais recente de sua versão do Aurora PostgreSQL.

> **Note**
>
> Para ver as notas de cada versão da extensão `apg_plan_mgmt`, consulte Aurora PostgreSQL apg_plan_mgmt extension versions (Versões da extensão apg_plan_mgmt do Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

Você pode identificar a versão em execução em seu cluster conectando-se a uma instância com `psql` e usando o metacomando \dx para listar extensões, conforme mostrado a seguir.

```
labdb=> \dx
                       List of installed extensions
     Name      | Version |    Schema     |                            Description
---------------+---------+---------------
+------------------------------------------------------------------------
 apg_plan_mgmt | 1.0     | apg_plan_mgmt | Amazon Aurora with PostgreSQL compatibility
 Query Plan Management
 plpgsql       | 1.0     | pg_catalog    | PL/pgSQL procedural language
(2 rows)
```

A saída mostra que esse cluster está usando a versão 1.0 da extensão. Somente determinadas versões de `apg_plan_mgmt` estão disponíveis para determinada versão do Aurora PostgreSQL. Em alguns casos, talvez seja necessário atualizar o cluster de banco de dados do Aurora PostgreSQL para uma nova versão secundária ou aplicar um patch para poder realizar a atualização para a versão mais recente do gerenciamento de planos de consulta. O `apg_plan_mgmt` versão 1.0 mostrado na saída é de um cluster de banco de dados do Aurora PostgreSQL versão 10.17, que não tem uma versão mais recente do `apg_plan_mgmt` disponível. Nesse caso, o cluster de banco de dados do Aurora PostgreSQL deve ser atualizado para uma versão mais recente do PostgreSQL.

Para obter mais informações sobre a atualização de um cluster de banco de dados do Aurora PostgreSQL para uma nova versão do PostgreSQL, consulte Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL.

Para saber como atualizar a extensão `apg_plan_mgmt`, consulte Atualizar o gerenciamento de planos de consultas do Aurora PostgreSQL.

Ativar o gerenciamento de planos de consulta do Aurora PostgreSQL

Configurar o gerenciamento de planos de consulta para seu cluster de banco de dados do Aurora PostgreSQL envolve a instalação de uma extensão e a alteração de várias configurações de parâmetros do cluster de banco de dados. Você precisa de permissões `rds_superuser` para instalar a extensão `apg_plan_mgmt` e ativar o recurso para o cluster de banco de dados do Aurora PostgreSQL.

A instalação da extensão cria uma função, `apg_plan_mgmt`. Essa função permite que os usuários do banco de dados visualizem, gerenciem e mantenham planos de consulta. Como administrador com privilégios `rds_superuser`, não deixe de conceder a função `apg_plan_mgmt` aos usuários do banco de dados conforme necessário.

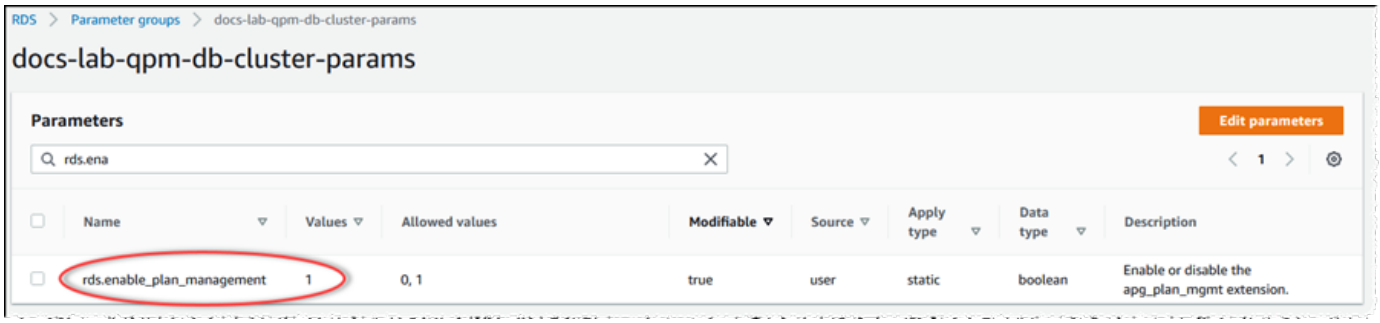
Somente os usuários com a função `rds_superuser` podem concluir o seguinte procedimento. `rds_superuser` é necessário para criar a extensão `apg_plan_mgmt` e sua função `apg_plan_mgmt`. Os usuários devem receber a função `apg_plan_mgmt` para administrar a extensão `apg_plan_mgmt`.

Como ativar o gerenciamento de planos de consulta para o cluster de banco de dados do Aurora PostgreSQL

As etapas a seguir ativam o gerenciamento de planos de consulta para todas as declarações SQL enviadas ao cluster de banco de dados do Aurora PostgreSQL. Isso é conhecido como modo automático. Para saber mais sobre a diferença entre os modos, consulte Capturar planos de execução do Aurora PostgreSQL.

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Crie um grupo de parâmetros de cluster de banco de dados personalizado para o cluster de bancos de dados do Aurora PostgreSQL. Você precisa alterar determinados parâmetros para ativar o gerenciamento de planos de consulta e definir seu comportamento. Para obter mais informações, consulte Criar um grupo de parâmetros de banco de dados no Amazon Aurora.

3. Abra o grupo de parâmetros de cluster de banco de dados personalizado e defina o parâmetro `rds.enable_plan_management` como 1, como mostrado na imagem a seguir.



   Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

4. Crie um grupo de parâmetros de banco de dados personalizado que você possa usar para definir os parâmetros do plano de consulta em nível de instância. Para obter mais informações, consulte Criar um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

5. Modifique a instância do gravador do cluster de banco de dados do Aurora PostgreSQL para usar o grupo de parâmetros de banco de dados personalizado. Para obter mais informações, consulte Modificar uma instância de banco de dados em um cluster de banco de dados.

6. Modifique o cluster de banco de dados do Aurora PostgreSQL para usar o grupo de parâmetros de cluster de banco de dados personalizado. Para obter mais informações, consulte Modificar o cluster de banco de dados usando o console, a CLI e a API.

7. Reinicialize a instância de banco de dados para habilitar as configurações do grupo de parâmetros personalizado.

8. Conecte-se ao endpoint da instância de banco de dados de seu cluster de banco de dados do Aurora PostgreSQL usando `psql` ou `pgAdmin`. O exemplo a seguir usa a conta `postgres` padrão para a função `rds_superuser`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=my-db
```

9. Crie a extensão `apg_plan_mgmt` para a instância de banco de dados, conforme mostrado a seguir.

```
labdb=> CREATE EXTENSION apg_plan_mgmt;
CREATE EXTENSION
```

> **Tip**
>
> Instale a extensão `apg_plan_mgmt` no banco de dados de modelo de sua aplicação. O banco de dados de modelo padrão é chamado de `template1`. Para saber mais, consulte Template Databases (Bancos de dados de modelos) na documentação do PostgreSQL.

10. Altere o parâmetro `apg_plan_mgmt.capture_plan_baselines` para `automatic`. Essa configuração faz com que o otimizador gere planos para cada declaração SQL planejada ou executada duas ou mais vezes.

> **Note**
>
> O gerenciamento de planos de consulta também tem um modo manual que você pode usar para declarações SQL específicas. Para saber mais, consulte Capturar planos de execução do Aurora PostgreSQL.

11. Altere o valor do parâmetro `apg_plan_mgmt.use_plan_baselines` para “on” (ativado). Esse parâmetro faz com que o otimizador selecione um plano para a declaração na linha de base do plano. Para saber mais, consulte Usar planos gerenciados do Aurora PostgreSQL.

> **Note**
>
> Você pode modificar o valor de qualquer um desses parâmetros dinâmicos para a sessão sem precisar reinicializar a instância.

Quando a configuração do gerenciamento de planos de consulta estiver concluída, conceda a função `apg_plan_mgmt` a todos os usuários do banco de dados que precisem visualizar, gerenciar ou manter planos de consulta.

Atualizar o gerenciamento de planos de consultas do Aurora PostgreSQL

Recomendamos que você atualize a extensão de gerenciamento de planos de consulta para a versão mais recente de sua versão do Aurora PostgreSQL.

1. Conecte-se à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL como um usuário com privilégios `rds_superuser`. Se você manteve o nome padrão ao

configurar a instância, se conectará como `postgres` Este exemplo mostra como usar `psql`, mas você também pode usar pgAdmin, se preferir.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Execute a consulta a seguir para atualizar a extensão.

```
ALTER EXTENSION apg_plan_mgmt UPDATE TO '2.1';
```

3. Use a função apg_plan_mgmt.validate_plans para atualizar os hashes de todos os planos. O otimizador valida todos os planos Aprovados, Não aprovados e Rejeitados para garantir que eles ainda sejam planos viáveis para a nova versão da extensão.

```
SELECT apg_plan_mgmt.validate_plans('update_plan_hash');
```

   Para saber mais sobre o uso dessa função, consulte Validar planos.

4. Use a função apg_plan_mgmt.reload para atualizar todos os planos na memória compartilhada com os planos validados na visualização dba_plans.

```
SELECT apg_plan_mgmt.reload();
```

Para saber mais sobre todas as funções disponíveis para o gerenciamento de planos de consulta, consulte Referência de funções do gerenciamento de planos de consultas do Aurora PostgreSQL.

Desativar o gerenciamento de planos de consulta do Aurora PostgreSQL

Você pode desativar o gerenciamento de planos de consulta a qualquer momento, desativando as `apg_plan_mgmt.use_plan_baselines` e as `apg_plan_mgmt.capture_plan_baselines`.

```
labdb=> SET apg_plan_mgmt.use_plan_baselines = off;

labdb=> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Práticas recomendadas para gerenciamento de planos de consultas do Aurora PostgreSQL

Com o gerenciamento de planos de consultas, você pode controlar como e quando os planos de execução de consultas mudam. Como DBA, suas principais metas ao usar o QPM incluem evitar regressões quando há alterações no banco de dados e controlar se o otimizador pode ou não usar um novo plano. A seguir, você pode encontrar algumas práticas recomendadas para o uso do gerenciamento de planos de consultas. As abordagens de gerenciamento de planos proativo e reativo diferem em relação a como e quando novos planos são aprovados para uso.

Sumário

- Gerenciamento de planos proativo para ajudar a evitar a regressão de performance
  - Garantir a estabilidade do plano após a atualização da versão principal
- Gerenciamento de planos reativo para detectar e reparar regressões de performance

Gerenciamento de planos proativo para ajudar a evitar a regressão de performance

Para evitar que ocorram regressões de performance do plano, você evolui as linhas de base do plano executando um procedimento que compara a performance dos planos recém-descobertos com a performance da linha de base existente dos planos aprovados e, depois, aprova automaticamente o conjunto mais rápido de planos como a nova linha de base. Dessa forma, a linha de base dos planos melhora com o tempo, à medida que planos mais rápidos são descobertos.

1. Em um ambiente de desenvolvimento, identifique as instruções SQL que tenham o maior impacto sobre a performance ou a produtividade do sistema. Em seguida, capture os planos dessas instruções conforme descrito em Capturar manualmente planos para instruções SQL específicas e Capturar planos automaticamente.
2. Exporte os planos capturados do ambiente de desenvolvimento e os importe para o ambiente de produção. Para obter mais informações, consulte Exportar e importar planos gerenciados do Aurora PostgreSQL.
3. Na produção, execute o aplicativo e imponha o uso de planos gerenciados aprovados. Para obter mais informações, consulte Usar planos gerenciados do Aurora PostgreSQL. Enquanto o aplicativo estiver em execução, adicione também novos planos enquanto o otimizador os descobre. Para obter mais informações, consulte Capturar planos automaticamente.
4. Analise os planos não aprovados e confirme os que apresentam boa performance. Para obter mais informações, consulte Avaliar performance do plano.

5. Enquanto o aplicativo continua sendo executado, o otimizador começa a usar os novos planos conforme apropriado.

Garantir a estabilidade do plano após a atualização da versão principal

Toda versão principal do PostgreSQL inclui melhorias e alterações no otimizador de consulta projetadas para melhorar a performance. No entanto, os planos de execução de consultas gerados pelo otimizador em versões anteriores podem causar regressões de performance em versões atualizadas mais recentes. Você pode usar o gerenciador de planos de consulta para resolver esses problemas de performance e garantir a estabilidade do plano após a atualização da versão principal.

O otimizador sempre usa o plano aprovado de custo mínimo, mesmo que haja mais de um plano aprovado para a mesma instrução. Depois de uma atualização, o otimizador pode descobrir novos planos, mas eles serão salvos como planos não aprovados. Esses planos serão executados somente se aprovados usando o estilo reativo de gerenciamento do plano com o parâmetro unapproved_plan_execution_threshold. Você pode maximizar a estabilidade do plano usando o estilo proativo de gerenciamento do plano com o parâmetro evolve_plan_baselines. Isso compara a performance dos novos planos com os planos antigos e aprova ou rejeita planos que são pelo menos 10% mais rápidos do que o próximo melhor plano.

Após a atualização, você pode usar a função `evolve_plan_baselines` para comparar a performance do plano antes e depois da atualização usando suas associações de parâmetros de consulta. As etapas a seguir pressupõem que você tenha usado planos gerenciados aprovados em seu ambiente de produção, conforme detalhado em Usar planos gerenciados do Aurora PostgreSQL.

1. Antes de atualizar, execute sua aplicação com o gerenciador de planos de consulta em execução. Enquanto a aplicação estiver em execução, adicione novos planos à medida que o otimizador os descobrir. Para obter mais informações, consulte Capturar planos automaticamente.
2. Avalie a performance de cada plano. Para obter mais informações, consulte Avaliar performance do plano.
3. Após a atualização, analise seus planos aprovados novamente usando a função `evolve_plan_baselines`. Compare a performance antes e depois de usar as associações de parâmetros de consulta. Se o novo plano for rápido, você poderá adicioná-lo aos planos aprovados. Se for mais rápido do que outro plano para as mesmas associações de parâmetros, você poderá marcar o plano mais lento como Rejected (Rejeitado).

   Para obter mais informações, consulte Aprovar planos melhores. Para obter informações de referência sobre essa função, consulte apg_plan_mgmt.evolve_plan_baselines.

Para obter mais informações, consulte Garantir uma performance consistente após atualizações de versões principais com o gerenciamento de planos de consultas do Amazon Aurora edição compatível com PostgreSQL.

> **Note**
>
> Ao realizar uma atualização de versão principal usando a replicação lógica ou AWS DMS, replique o esquema `apg_plan_mgmt` para garantir que os planos existentes sejam copiados na instância atualizada. Para obter mais informações sobre replicação lógica, consulte Usar replicação lógica para realizar uma atualização de versão principal do Aurora PostgreSQL.

Gerenciamento de planos reativo para detectar e reparar regressões de performance

Ao monitorar a aplicação enquanto ela é executada, é possível detectar planos que causam regressões de performance. Ao detectar regressões, rejeite ou corrija manualmente os planos inadequados seguindo estas etapas:

1. Enquanto o aplicativo estiver em execução, imponha o uso de planos gerenciados e adicione automaticamente planos recém-descobertos como não aprovados. Para obter mais informações, consulte Usar planos gerenciados do Aurora PostgreSQL e Capturar planos automaticamente.
2. Monitore o aplicativo em execução em busca de regressões de performance.
3. Ao descobrir uma regressão de plano, defina o status do plano como `rejected`. Na próxima vez em que executar a instrução SQL, o otimizador vai ignorar automaticamente o plano rejeitado e usar um plano aprovado diferente. Para obter mais informações, consulte Rejeitar ou desabilitar planos mais lentos.

   Em alguns casos, convém corrigir um plano inválido, em vez de rejeitar, desabilitar ou excluí-lo. Use a extensão `pg_hint_plan` para testar melhorando um plano. Com `pg_hint_plan`, você usa comentários especiais para informar o otimizador como ele normalmente cria um plano. Para obter mais informações, consulte Corrigir planos usando pg_hint_plan.

Gerenciamento de planos de consulta do Aurora PostgreSQL

Com o gerenciamento de planos de consulta ativado para seu cluster de banco de dados do Aurora PostgreSQL, o otimizador gera e armazena planos de execução de consultas para qualquer declaração SQL processada mais de uma vez. O otimizador sempre define o status do primeiro

plano gerado de uma declaração gerenciada como `Approved` e o armazena na visualização `dba_plans`.

O conjunto de planos aprovados salvos para uma declaração gerenciada é conhecido como linha de base de planos. Enquanto a aplicação é executada, o otimizador pode gerar planos adicionais para as declarações gerenciadas. O otimizador define planos capturados adicionais para um status de `Unapproved`.

Posteriormente, será possível decidir se os planos `Unapproved` apresentam boa performance e alterá-los para `Approved`, `Rejected` ou `Preferred`. Para fazer isso, você usa a função `apg_plan_mgmt.evolve_plan_baselines` ou `apg_plan_mgmt.set_plan_status`.

Quando o otimizador gera um plano para uma instrução SQL, o gerenciamento de planos de consulta salva o plano na tabela `apg_plan_mgmt.plans`. Os usuários do banco de dados que receberam a função `apg_plan_mgmt` podem ver os detalhes do plano consultando a visualização `apg_plan_mgmt.dba_plans`. Por exemplo, a consulta a seguir lista os detalhes dos planos atualmente na visualização de um cluster de banco de dados do Aurora PostgreSQL que não esteja em produção.

- `sql_hash`: um identificador da instrução SQL que é o valor de hash para o texto normalizado da instrução SQL.
- `plan_hash`: um identificador exclusivo para o plano que é uma combinação do `sql_hash` e um hash do plano.
- `status` – O status do plano. O otimizador pode executar um plano aprovado.
- `enabled`: indica se o plano está pronto para uso (verdadeiro) ou não (falso).
- `plan_outline`: uma representação do plano utilizada para recriar o plano de execução real. Os operadores na estrutura em árvore são mapeados para operadores na saída EXPLAIN.

A visualização `apg_plan_mgmt.dba_plans` tem muitas outras colunas que contêm todos os detalhes do plano, como quando ele foi utilizado pela última vez. Para obter detalhes completos, consulte Referência da visualização apg_plan_mgmt.dba_plans da edição compatível do Aurora PostgreSQL.

Normalização e o hash SQL

Na visualização `apg_plan_mgmt.dba_plans`, identifique uma declaração gerenciada com um valor de hash SQL. O hash SQL é calculado com base em uma representação normalizada da declaração SQL que remove algumas diferenças, como os valores literais.

O processo de normalização de cada declaração SQL preserva espaço e maiúsculas e minúsculas, para que você ainda possa ler e entender a essência da declaração SQL. A normalização remove ou substitui os itens a seguir.

- Principais comentários do bloco
- A palavra-chave EXPLAIN e as opções EXPLAIN e EXPLAIN ANALYZE
- Espaços à direita
- Todos os literais

Por exemplo, utilize a declaração a seguir.

```
/*Leading comment*/ EXPLAIN SELECT /* Query 1 */ * FROM t WHERE x > 7 AND y = 1;
```

O gerenciamento de planos de consulta normaliza essa declaração da seguinte maneira:

```
SELECT /* Query 1 */ * FROM t WHERE x > CONST AND y = CONST;
```

A normalização permite que o mesmo hash SQL seja utilizado para declarações SQL semelhantes que só podem ter diferenças nos valores literais ou de parâmetro. Em outras palavras, podem existir vários planos para o mesmo hash SQL, com um plano diferente que é ideal em condições diferentes.

> **Note**
>
> Uma única declaração SQL utilizada com esquemas diferentes tem planos diferentes porque está vinculada ao esquema específico em tempo de execução. O planejador usa as estatísticas para vinculação do esquema para selecionar o plano ideal.

Para saber mais sobre como o otimizador seleciona um plano, consulte Usar planos gerenciados do Aurora PostgreSQL. Nessa seção, você pode aprender a usar `EXPLAIN` e `EXPLAIN ANALYZE` e visualizar um plano antes que ele seja realmente utilizado. Para obter detalhes, consulte Analisar o plano selecionado pelo otimizador. Para obter uma imagem que descreve o processo de seleção de um plano, consulte Como o otimizador escolhe que plano executar..

Capturar planos de execução do Aurora PostgreSQL

O gerenciamento de planos de consulta do Aurora PostgreSQL oferece dois modos diferentes para capturar planos de execução de consultas: automático ou manual. Você seleciona o modo definindo

o valor de `apg_plan_mgmt.capture_plans_baselines` como `automatic` ou `manual`. Você pode capturar planos de execução para instruções SQL específicas usando a captura de plano manual. Como alternativa, você pode capturar todos (ou os mais lentos) os planos executados duas ou mais vezes durante as execuções do aplicativo usando a captura de plano automática.

Ao capturar planos, o otimizador define o status do primeiro plano capturado de uma instrução gerenciada como `approved`. O otimizador define o status de todos os planos adicionais capturados para uma instrução gerenciada como `unapproved`. No entanto, mais de um plano pode, ocasionalmente, ser salvo com o status de `approved`. Isso pode ocorrer quando vários planos são criados para uma instrução em paralelo e antes da confirmação do primeiro plano da instrução.

Para controlar o número máximo de planos que podem ser capturados e armazenados na visualização `dba_plans`, defina o parâmetro `apg_plan_mgmt.max_plans` no grupo de parâmetros no nível de instância de banco de dados. Uma alteração feita no parâmetro `apg_plan_mgmt.max_plans` exige a reinicialização de uma instância de banco de dados para que um novo valor entre em vigor. Para obter mais informações, consulte o parâmetro apg_plan_mgmt.max_plans.

Capturar manualmente planos para instruções SQL específicas

Caso você tenha um conjunto conhecido de instruções SQL a serem gerenciadas, coloque as instruções em um arquivo SQL e capture manualmente os planos. A seguir, um exemplo de psql de como capturar planos de consulta manualmente para um conjunto de instruções SQL.

```
psql> SET apg_plan_mgmt.capture_plan_baselines = manual;
psql> \i my-statements.sql
psql> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Depois de capturar um plano para cada instrução SQL, o otimizador adicionará uma nova linha à visualização `apg_plan_mgmt.dba_plans`.

Recomendamos usar instruções EXPLAIN ou EXPLAIN EXECUTE no arquivo script do SQL. Garanta a inclusão de variações suficientes em valores de parâmetros para capturar todos os planos de interesse.

Se você conhecer um plano melhor do que o plano de custo mínimo do otimizador, poderá forçar o otimizador a usar o plano melhor. Para isso, especifique uma ou mais dicas do otimizador. Para obter mais informações, consulte Corrigir planos usando pg_hint_plan. Para comparar a performance dos planos `unapproved` e `approved` e aprovar, rejeitar ou excluí-los, consulte Avaliar performance do plano.

Capturar planos automaticamente

Use a captura de planos automática em situações como a seguinte:

- Você não sabe as instruções SQL específicas que deseja gerenciar.
- Você tem centenas ou milhares de instruções SQL a serem gerenciadas.
- O aplicativo usa uma API cliente. Por exemplo, o JDBC usa instruções preparadas não nomeadas ou instruções de modo em lote que não podem ser expressadas em psql.

Para capturar planos automaticamente

1. Ative a captura de planos automática definindo `apg_plan_mgmt.capture_plan_baselines` como `automatic` no grupo de parâmetros no nível da instância de banco de dados. Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.
2. Quando o aplicativo é executado, o otimizador captura planos de todas as instruções SQL executadas pelo menos duas vezes.

   Quando o aplicativo é executado com as configurações do parâmetro de gerenciamento de planos de consultas padrão, o otimizador captura planos de todas as instruções SQL executadas pelo menos duas vezes. Capturar todos os planos usando os padrões tem pouquíssimos custos indiretos no tempo de execução e pode ser habilitado na produção.

Para desativar a captura de planos automática

- Defina o parâmetro `apg_plan_mgmt.capture_plan_baselines` como `off` do grupo de parâmetros no nível de instância de banco de dados.

Para avaliar a performance dos planos não aprovados e aprovar, rejeitar ou excluí-los, consulte Avaliar performance do plano.

Usar planos gerenciados do Aurora PostgreSQL

Para fazer o otimizador usar planos capturados para as instruções gerenciadas, defina o parâmetro `apg_plan_mgmt.use_plan_baselines` como `true`. Veja uma instância local de exemplo.

```
SET apg_plan_mgmt.use_plan_baselines = true;
```

Enquanto a aplicação estiver em execução, essa configuração fará com que o otimizador use o plano de custo mínimo, preferido ou aprovado que seja válido e esteja habilitado em cada declaração gerenciada.

Analisar o plano selecionado pelo otimizador

Quando o parâmetro `apg_plan_mgmt.use_plan_baselines` é definido como `true`, use instruções EXPLAIN ANALYZE SQL para fazer o otimizador mostrar o plano que ele usaria se fosse executar a instrução. Veja um exemplo a seguir.

```
EXPLAIN ANALYZE EXECUTE rangeQuery (1,10000);
```

```
                                                        QUERY PLAN
-----------------------------------------------------------------------------
 Aggregate  (cost=393.29..393.30 rows=1 width=8) (actual time=7.251..7.251 rows=1
 loops=1)
   ->  Index Only Scan using t1_pkey on t1 t  (cost=0.29..368.29 rows=10000 width=0)
 (actual time=0.061..4.859 rows=10000 loops=1)
Index Cond: ((id >= 1) AND (id <= 10000))
         Heap Fetches: 10000
 Planning time: 1.408 ms
 Execution time: 7.291 ms
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1984047223, Plan Hash: 512153379
```

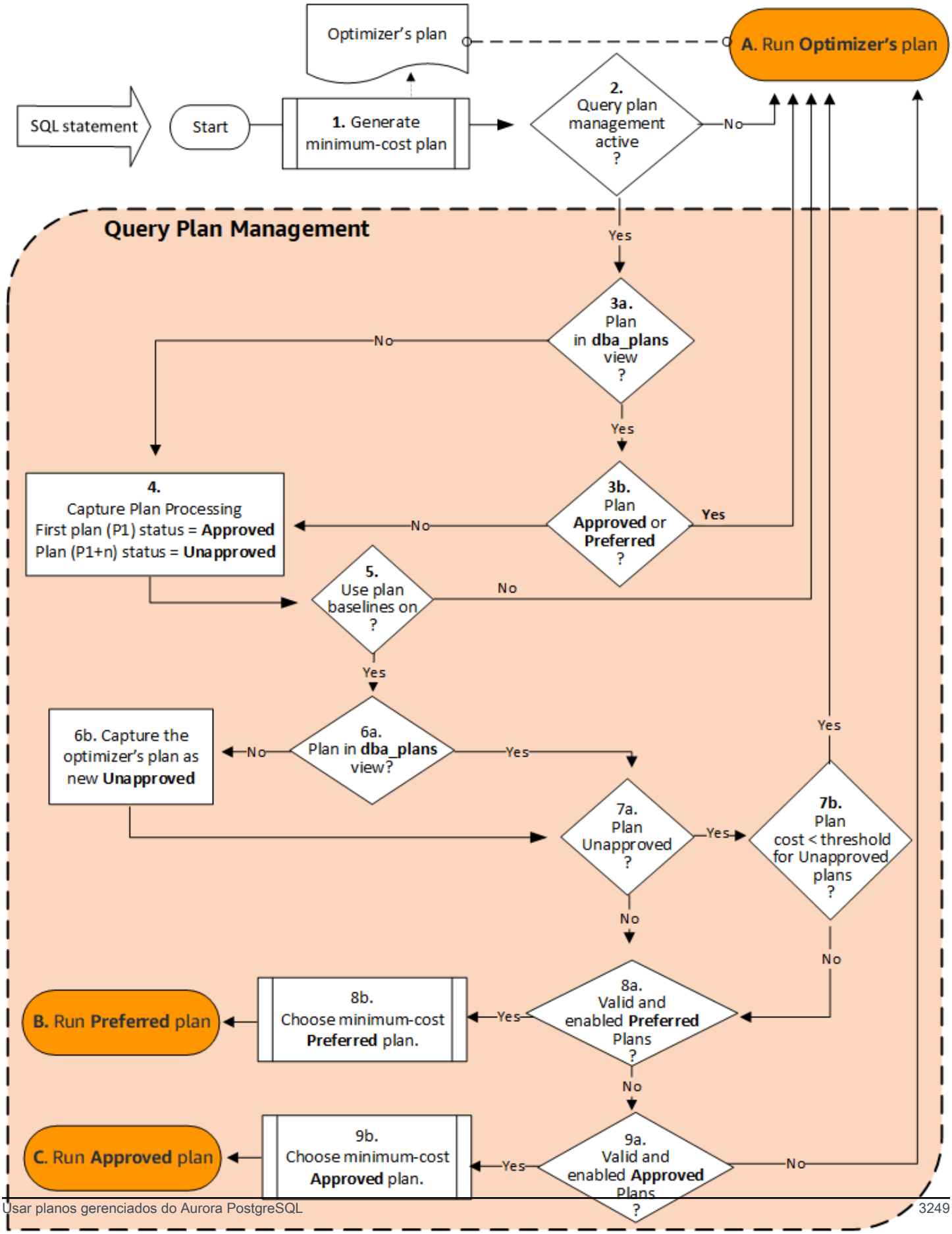
O resultado mostra o plano aprovado que seria executado a partir da linha de base. No entanto, o resultado também mostra que ele encontrou um plano de custo mais baixo. Neste caso, você captura esse novo plano de custo mínimo ativando a captura de planos automáticos conforme descrito em Capturar planos automaticamente.

Novos planos são sempre capturados pelo otimizador como `Unapproved`. Use a função `apg_plan_mgmt.evolve_plan_baselines` para comparar planos e alterá-los para aprovados, rejeitados ou desabilitados. Para obter mais informações, consulte Avaliar performance do plano.

Como o otimizador escolhe que plano executar.

O custo de um plano de execução é uma estimativa feita pelo otimizador para comparar planos diferentes. Ao calcular o custo de um plano, o otimizador inclui fatores como operações de CPU e E/S exigidas por esse plano. Para saber mais sobre as estimativas de custo do planejador de consultas PostgreSQL, consulte Planejamento de consultas na documentação do PostgreSQL.

A imagem a seguir mostra como um plano é selecionado para determinada declaração SQL quando o gerenciamento de planos de consulta está ativo e quando não está.


Optimizer's plan
A. Run Optimizer's plan
SQL statement
Start
1. Generate minimum-cost plan
2. Query plan management active ?
No
Query Plan Management
Yes
3a. Plan in dba_plans view ?
No
Yes
4. Capture Plan Processing
First plan (P1) status = Approved
Plan (P1+n) status = Unapproved
3b. Plan Approved or Preferred ?
No
Yes
5. Use plan baselines on ?
No
Yes
6a. Plan in dba_plans view?
No
6b. Capture the optimizer's plan as new Unapproved
Yes
7a. Plan Unapproved ?
Yes
7b. Plan cost < threshold for Unapproved plans ?
Yes
No
No
8a. Valid and enabled Preferred Plans ?
Yes
8b. Choose minimum-cost Preferred plan.
B. Run Preferred plan
No
9a. Valid and enabled Approved Plans ?
Yes
9b. Choose minimum-cost Approved plan.
C. Run Approved plan
No

O fluxo é o seguinte:

1. O otimizador gera um plano de custo mínimo para a instrução SQL.
2. Se o gerenciamento do plano de consulta não estiver ativo, o plano do otimizador será executado imediatamente (A. Executar o plano do otimizador). O gerenciamento de planos de consultas está inativo quando os parâmetros `apg_plan_mgmt.capture_plan_baselines` e `apg_plan_mgmt.use_plan_baselines` estão em suas configurações padrão (“off” e “false”, respectivamente).

   Caso contrário, o gerenciamento de planos de consultas estará ativo. Nesse caso, a instrução SQL e o plano do otimizador para ela são avaliados em mais detalhes antes que um plano seja escolhido.

   > **Tip**
   >
   > Usuários do banco de dados com o perfil `apg_plan_mgmt` podem comparar planos de maneira proativa, alterar o status dos planos e forçar o uso de planos específicos, conforme necessário. Para obter mais informações, consulte Melhorar os planos de consulta do Aurora PostgreSQL.

3. A instrução SQL pode já ter planos que foram armazenados pelo gerenciamento de planos de consultas no passado. Os planos são armazenados em `apg_plan_mgmt.dba_plans`, junto com informações sobre as instruções SQL que foram usadas para criá-los. As informações sobre um plano incluem seu status. O status de um plano pode determinar se ele é usado ou não, conforme mostrado a seguir.
   a. Se o plano não estiver entre os planos armazenados para a instrução SQL, significa que é a primeira vez que esse plano específico foi gerado pelo otimizador para a instrução SQL fornecida. O plano é enviado para o processamento de captura de plano (4).
   b. Se o plano estiver entre os planos armazenados e seu status for Aprovado ou Preferencial, o plano será executado (A. Executar o plano do otimizador).

      Se o plano estiver entre os planos armazenados, mas não for Aprovado nem Preferencial, o plano será enviado para o processamento de captura de plano (4).
4. Quando um plano é capturado pela primeira vez para determinada instrução SQL, o status do plano é sempre definido como Aprovado (P1). Se o otimizador gerar posteriormente o mesmo plano para a mesma instrução SQL, o status desse plano será alterado para Não aprovado (P1+n).

Com o plano capturado e seu status atualizado, a avaliação continua na próxima etapa (5).

5. A linha de base de um plano consiste no histórico da instrução SQL e em seus planos em vários estados. O gerenciamento de planos de consultas pode levar em consideração a linha de base ao escolher um plano, dependendo se a opção de usar linhas de base do plano está ativada ou não, conforme mostrado a seguir.
   - A opção de usar linhas de base do plano está “off” quando o parâmetro `apg_plan_mgmt.use_plan_baselines` está definido com o valor padrão (`false`). O plano não é comparado com a linha de base antes de ser executado (A. Executar plano do otimizador).
   - A opção de usar linhas de base do plano está “on” quando o parâmetro `apg_plan_mgmt.use_plan_baselines` está definido como `true`. O plano é avaliado em mais detalhes usando a linha de base (6).
6. O plano é comparado a outros planos para a instrução na linha de base.
   a. Se o plano do otimizador estiver entre os planos na linha de base, seu status será verificado (7a).
   b. Se o plano do otimizador não estiver entre os planos na linha de base, o plano será adicionado aos planos da instrução como um novo plano `Unapproved`.
7. O status do plano é verificado somente para determinar se ele é Não aprovado.
   a. Se o status do plano for Não aprovado, o custo estimado do plano será comparado com a estimativa de custo especificada para o limite do plano de execução não aprovado.
      - Se o custo estimado do plano estiver abaixo do limite, o otimizador o usará mesmo sendo um plano Não aprovado (A. Executar o plano do otimizador). Geralmente, o otimizador não executa um plano Não aprovado. No entanto, quando o parâmetro `apg_plan_mgmt.unapproved_plan_execution_threshold` especifica um valor de limite de custo, o otimizador compara o custo do plano Não aprovado com o limite. Se o custo estimado for menor que o limite, o otimizador executará o plano. Para obter mais informações, consulte apg_plan_mgmt.unapproved_plan_execution_threshold.
      - Se o custo estimado do plano não estiver abaixo do limite, os outros atributos do plano serão verificados (8a).
   b. Se o status do plano for diferente de Não aprovado, seus outros atributos serão verificados (8a).
8. O otimizador não usará um plano desativado. Ou seja, um plano com o atributo `enable` definido como “f” (false). O otimizador também não usará um plano com status Rejeitado.

O otimizador não pode usar nenhum plano que não seja válido. Os planos podem se tornar inválidos ao longo do tempo quando os objetos dos quais dependem, como índices e partições de tabelas, são removidos ou excluídos.

a. Se a instrução tiver planos habilitados e válidos com status Preferencial, o otimizador escolherá o plano de custo mínimo dentre os planos com status Preferencial armazenados para essa instrução SQL. Depois, o otimizador executa o plano Preferencial de custo mínimo.

b. Se a instrução não tiver nenhum plano Preferencial habilitado e válido, será avaliada na próxima etapa (9).

9. Se a instrução tiver planos habilitados e válidos com status Aprovado, o otimizador escolherá o plano de custo mínimo dentre os planos com status Aprovado armazenados para essa instrução SQL. Depois, o otimizador executa o plano Aprovado de custo mínimo.

   Se a instrução não tiver nenhum plano Aprovado habilitado e válido, o otimizador usará o plano de custo mínimo (A. Executar o plano do otimizador).

Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans

Os usuários e os administradores do banco de dados que receberam a função `apg_plan_mgmt` podem visualizar e gerenciar os planos armazenados em `apg_plan_mgmt.dba_plans`. O administrador de um cluster de banco de dados do Aurora PostgreSQL (alguém com permissões `rds_superuser`) deve conceder explicitamente essa função aos usuários do banco de dados que precisam trabalhar com o gerenciamento de planos de consulta.

A visualização `apg_plan_mgmt` contém o histórico dos planos de todas as declarações SQL gerenciadas para cada banco de dados na instância do gravador do cluster de banco de dados do Aurora PostgreSQL. Essa visualização permite examinar os planos, seu estado, quando foram utilizados pela última vez e todos os outros detalhes relevantes.

Conforme discutido em Normalização e o hash SQL, cada plano gerenciado é identificado pela combinação de um valor de hash SQL e de um valor de hash do plano. Com esses identificadores, use ferramentas como o Amazon RDS Performance Insights para rastrear a performance do plano individual. Para obter mais informações sobre o Performance Insights, consulte Usar o Amazon RDS Performance Insights.

Listar planos gerenciados

Para listar os planos gerenciados, use uma instrução SELECT na visualização `apg_plan_mgmt.dba_plans`. O exemplo a seguir exibe algumas colunas na visualização `dba_plans`, como o `status`, que identifica os planos aprovados e não aprovados.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;

 sql_hash   | plan_hash |   status   | enabled | stmt_name
------------+-----------+------------+---------+------------
 1984047223 | 512153379 | Approved   | t       | rangequery
 1984047223 | 512284451 | Unapproved | t       | rangequery
 (2 rows)
```

Para facilitar a leitura, a consulta e a saída mostradas listam apenas algumas das colunas da visualização `dba_plans`. Para obter informações completas, consulte Referência da visualização apg_plan_mgmt.dba_plans da edição compatível do Aurora PostgreSQL.

Melhorar os planos de consulta do Aurora PostgreSQL

Melhore o gerenciamento dos planos de consulta avaliando a performance e corrigindo os planos. Para ter mais informações sobre como melhorar planos de consulta, confira os tópicos a seguir.

Tópicos

- Avaliar performance do plano
- Corrigir planos usando pg_hint_plan

Avaliar performance do plano

Depois que o otimizador capturar planos como não aprovados, use a função `apg_plan_mgmt.evolve_plan_baselines` para comparar planos com base na performance real. Dependendo do resultado dos experimentos de performance, altere o status de um plano de não aprovado para aprovado ou rejeitado. Em vez disso, opte por usar a função `apg_plan_mgmt.evolve_plan_baselines` para desabilitar temporariamente um plano caso ele não atenda aos requisitos.

Aprovar planos melhores

O exemplo a seguir demonstra como alterar o status de planos gerenciados para aprovados usando a função `apg_plan_mgmt.evolve_plan_baselines`.

```
SELECT apg_plan_mgmt.evolve_plan_baselines (
   sql_hash,
   plan_hash,
   min_speedup_factor := 1.0,
   action := 'approve'
)
FROM apg_plan_mgmt.dba_plans WHERE status = 'Unapproved';
```

```
NOTICE:     rangequery (1,10000)
NOTICE:     Baseline   [ Planning time 0.761 ms, Execution time 13.261 ms]
NOTICE:     Baseline+1 [ Planning time 0.204 ms, Execution time 8.956 ms]
NOTICE:     Total time benefit: 4.862 ms, Execution time benefit: 4.305 ms
NOTICE:     Unapproved -> Approved
evolve_plan_baselines
----------------------
0
(1 row)
```

A saída mostra um relatório de performance da instrução `rangequery` com associações de parâmetro 1 e 10.000. O novo plano não aprovado (`Baseline+1`) é melhor do que o melhor plano aprovado anteriormente (`Baseline`). Para confirmar se o novo plano já está `Approved`, verifique a visualização `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
sql_hash  | plan_hash |  status  | enabled | stmt_name
------------+-----------+----------+---------+------------
1984047223 | 512153379 | Approved | t       | rangequery
1984047223 | 512284451 | Approved | t       | rangequery
(2 rows)
```

O plano gerenciado já inclui dois planos aprovados que são a linha de base do plano da instrução. Também é possível chamar a função `apg_plan_mgmt.set_plan_status` para definir

diretamente o campo de status de um plano como `'Approved'`, `'Rejected'`, `'Unapproved'` ou `'Preferred'`.

Rejeitar ou desabilitar planos mais lentos

Para rejeitar ou desabilitar planos, passe `'reject'` ou `'disable'` como o parâmetro de ação para a função `apg_plan_mgmt.evolve_plan_baselines`. Esse exemplo desativa todos os planos `Unapproved` capturados que sejam pelo menos 10% mais lentos do que o melhor plano `Approved` para a instrução.

```
SELECT apg_plan_mgmt.evolve_plan_baselines(
sql_hash,  -- The managed statement ID
plan_hash, -- The plan ID
1.1,       -- number of times faster the plan must be
'disable'  -- The action to take. This sets the enabled field to false.
)
FROM apg_plan_mgmt.dba_plans
WHERE status = 'Unapproved' AND   -- plan is Unapproved
origin = 'Automatic';       -- plan was auto-captured
```

Também defina diretamente um plano como rejeitado ou desabilitado. Para definir diretamente o campo habilitado de um plano como `true`, ou `false`, chame a função `apg_plan_mgmt.set_plan_enabled`. Para definir diretamente o campo de status de um plano como `'Approved'`, `'Rejected'`, `'Unapproved'` ou `'Preferred'`, chame a função `apg_plan_mgmt.set_plan_status`.

Para excluir planos que não são válidos e que você espera que permaneçam inválidos, use a função `apg_plan_mgmt.validate_plans`. Essa função permite excluir ou desabilitar planos inválidos. Para obter mais informações, consulte Validar planos.

Corrigir planos usando pg_hint_plan

O otimizador de consulta foi bem projetado para encontrar um plano ideal para todas as instruções e, na maioria dos casos, o otimizador encontra um plano bom. No entanto, às vezes, talvez você saiba que existe um plano muito melhor do que o gerado pelo otimizador. Duas maneiras recomendadas para fazer o otimizador gerar um plano desejado incluem usar a extensão `pg_hint_plan` ou definir variáveis GUC em PostgreSQL:

- Extensão `pg_hint_plan` – especifique uma "dica" para modificar como o planejador funciona usando a extensão `pg_hint_plan` PostgreSQL. Para instalar e saber mais sobre com usar a extensão `pg_hint_plan`, consulte a documentação pg_hint_plan.
- Variáveis GUC – Substitua um ou mais parâmetros de modelo de custo ou outros parâmetros de otimizador, como o `from_collapse_limit` ou o `GEQO_threshold`.

Ao usar uma dessas técnicas para forçar otimizador de consultas a usar um plano, também use o gerenciamento de planos de consultas para capturar impor o uso do novo plano.

Use a extensão `pg_hint_plan` para alterar a ordem da junção, os métodos da junção ou os caminhos de acesso de uma instrução SQL. Use um comentário SQL com sintaxe `pg_hint_plan` especial para modificar como o otimizador cria um plano. Por exemplo, suponhamos que a instrução SQL problemática tenha uma junção bidirecional.

```
SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Depois, suponhamos que o otimizador selecione a ordem de junção (t1, t2), embora você saiba que a ordem de junção (t2, t1) seja mais rápida. A dica a seguir força o otimizador a usar a ordem de junção mais rápida, (t2, t1). Inclua EXPLAIN, de maneira que o otimizador gere um plano para a declaração SQL, mas sem executar a declaração. (Saída não mostrada.)

```
/*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

As etapas a seguir mostram como usar `pg_hint_plan`.

Para modificar o plano gerado do otimizador e capturar o plano usando pg_hint_plan

1. Ative o modo de captura manual.

   ```
   SET apg_plan_mgmt.capture_plan_baselines = manual;
   ```

2. Especifique uma dica para a instrução SQL de interesse.

   ```
   /*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
   ```

```
FROM t1, t2
WHERE t1.id = t2.id;
```

Depois dessa execução, o otimizador captura o plano na visualização `apg_plan_mgmt.dba_plans`. O plano capturado não inclui a sintaxe do comentário especial `pg_hint_plan` porque o gerenciamento de planos de consultas normaliza a instrução removendo os principais comentários.

3. Veja os planos gerenciados usando a visualização `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, sql_text, plan_outline
FROM apg_plan_mgmt.dba_plans;
```

4. Defina o status do plano como `Preferred`. Isso garante que o otimizador optará por executá-lo, em vez de selecioná-lo no conjunto de planos aprovados quando o plano de custo mínimo ainda não for `Approved` ou `Preferred`.

```
SELECT apg_plan_mgmt.set_plan_status(sql-hash, plan-hash, 'preferred' );
```

5. Desative a captura de plano manual e imponha o uso de planos gerenciados.

```
SET apg_plan_mgmt.capture_plan_baselines = false;
SET apg_plan_mgmt.use_plan_baselines = true;
```

Agora, quando a instrução SQL original for executada, o otimizador escolherá um plano `Approved` ou `Preferred`. Se o plano de custo mínimo não for `Approved` nem `Preferred`, o otimizador escolherá o plano `Preferred`.

Excluir os planos de consulta do Aurora PostgreSQL

Exclua os planos de execução que você não está usando ou que não são válidos. Para ter mais informações sobre exclusão de planos, consulte as seções a seguir.

Tópicos

- Excluir planos
- Validar planos

Excluir planos

Os planos serão excluídos automaticamente se não forem utilizados em mais de um mês, especificamente, 32 dias. Essa é a configuração padrão do parâmetro `apg_plan_mgmt.plan_retention_period`. Você pode alterar o período de retenção do plano para um período mais longo ou mais curto, a partir do valor 1. A determinação do número de dias desde o último uso de um plano é calculada subtraindo a data `last_used` da data atual. A data `last_used` é a mais recente em que o otimizador selecionou um plano como plano de custo mínimo ou em que o plano foi executado. A data é armazenada para o plano na visualização `apg_plan_mgmt.dba_plans`.

Recomendamos que você exclua planos que não tiverem sido usados por muito tempo ou que não tiverem sido úteis. Cada plano tem uma data `last_used` atualizada pelo otimizador sempre que ele executa um plano ou o seleciona como o plano de custo mínimo para uma declaração. Confira as últimas datas `last_used` para identificar os planos que você pode excluir com segurança.

A consulta a seguir retorna uma tabela de três colunas com a contagem do número total de planos, dos planos que não foram excluídos e dos planos que foram excluídos. Há uma consulta aninhada que é um exemplo de como usar a função `apg_plan_mgmt.delete_plan` para excluir todos os planos que não tiverem sido selecionados como o plano de custo mínimo nos últimos 31 dias e o status não seja `Rejected`.

```
SELECT (SELECT COUNT(*) from apg_plan_mgmt.dba_plans) total_plans,
       COUNT(*) FILTER (WHERE result = -1) failed_to_delete,
       COUNT(*) FILTER (WHERE result = 0) successfully_deleted
       FROM (
            SELECT apg_plan_mgmt.delete_plan(sql_hash, plan_hash) as result
            FROM apg_plan_mgmt.dba_plans
            WHERE last_used < (current_date - interval '31 days')
            AND status <> 'Rejected'
            ) as dba_plans ;
```

```
 total_plans | failed_to_delete | successfully_deleted
-------------+------------------+----------------------
           3 |                0 |                    2
```

Para obter mais informações, consulte apg_plan_mgmt.delete_plan.

Para excluir planos que não são válidos e que você espera que permaneçam inválidos, use a função `apg_plan_mgmt.validate_plans`. Essa função permite excluir ou desabilitar planos inválidos. Para obter mais informações, consulte Validar planos.

> **Important**
>
> Caso você não exclua planos divergentes, poderá acabar ficando sem memória compartilhada separada para o gerenciamento de planos de consultas. Para controlar a memória disponível para planos gerenciados, use o parâmetro `apg_plan_mgmt.max_plans`. Defina esse parâmetro no grupo de parâmetros de banco de dados personalizado e reinicie a instância de banco de dados para que as alterações tenham efeito. Para obter mais informações, consulte o parâmetro apg_plan_mgmt.max_plans.

Validar planos

Use a função `apg_plan_mgmt.validate_plans` para excluir ou desabilitar planos que sejam inválidos.

Os planos podem se tornar inválidos ou obsoletos quando objetos dependentes são removidos, como um índice ou uma tabela. No entanto, um plano só poderá ser inválido temporariamente se o objeto removido for recriado. Caso um plano inválido se torne válido depois, convém optar por desabilitar um plano inválido ou não fazer nada, em vez de excluí-lo.

Para encontrar e excluir todos os planos que sejam inválidos e que não tenham sido usados na semana passada, use a função `apg_plan_mgmt.validate_plans` da maneira a seguir.

```
SELECT apg_plan_mgmt.validate_plans(sql_hash, plan_hash, 'delete')
FROM apg_plan_mgmt.dba_plans
WHERE last_used < (current_date - interval '7 days');
```

Para habilitar ou desabilitar diretamente um plano, use a função `apg_plan_mgmt.set_plan_enabled`.

Exportar e importar planos gerenciados do Aurora PostgreSQL

Exporte os planos gerenciados e os importe para outra instância de banco de dados.

Para exportar planos gerenciados

Um usuário autorizado pode copiar qualquer subconjunto da tabela `apg_plan_mgmt.plans` para outra tabela e salvá-lo usando o comando `pg_dump`. Veja um exemplo a seguir.

```
CREATE TABLE plans_copy AS SELECT *
FROM apg_plan_mgmt.plans [ WHERE predicates ] ;
```

```
% pg_dump --table apg_plan_mgmt.plans_copy -Ft mysourcedatabase > plans_copy.tar
```

```
DROP TABLE apg_plan_mgmt.plans_copy;
```

Para importar planos gerenciados

1. Copie o arquivo .tar dos planos gerenciados exportados para o sistema onde os planos precisam ser restaurados.
2. Use o comando `pg_restore` a fim de copiar o arquivo tar para uma nova tabela.

   ```
   % pg_restore --dbname mytargetdatabase -Ft plans_copy.tar
   ```
3. Mescle a tabela `plans_copy` com a tabela `apg_plan_mgmt.plans`, conforme mostrado no exemplo a seguir.

   > **Note**
   >
   > Em alguns casos, você pode descarregar de uma versão da extensão `apg_plan_mgmt` e restaurar para outra versão. Nesses casos, as colunas na tabela de planos podem ser diferentes. Se for diferente, nomeie as colunas explicitamente, em vez de usar SELECT *.

   ```
   INSERT INTO apg_plan_mgmt.plans SELECT * FROM plans_copy
    ON CONFLICT ON CONSTRAINT plans_pkey
    DO UPDATE SET
    status = EXCLUDED.status,
    enabled = EXCLUDED.enabled,
    -- Save the most recent last_used date
    --
    last_used = CASE WHEN EXCLUDED.last_used > plans.last_used
    THEN EXCLUDED.last_used ELSE plans.last_used END,
   ```

```
-- Save statistics gathered by evolve_plan_baselines, if it ran:
--
estimated_startup_cost = EXCLUDED.estimated_startup_cost,
estimated_total_cost = EXCLUDED.estimated_total_cost,
planning_time_ms = EXCLUDED.planning_time_ms,
execution_time_ms = EXCLUDED.execution_time_ms,
total_time_benefit_ms = EXCLUDED.total_time_benefit_ms,
execution_time_benefit_ms = EXCLUDED.execution_time_benefit_ms;
```

4. Recarregue os planos gerenciados na memória compartilhada e remova a tabela de planos temporária.

```
SELECT apg_plan_mgmt.reload(); -- refresh shared memory
DROP TABLE plans_copy;
```

Referência de parâmetros do gerenciamento de planos de consultas do Aurora PostgreSQL

Você pode definir suas preferências para a extensão `apg_plan_mgmt` usando os parâmetros listados nesta seção. Eles estão disponíveis no parâmetro de cluster de banco de dados personalizado e no grupo de parâmetros de banco de dados associado ao seu cluster de banco de dados do Aurora PostgreSQL. Esses parâmetros controlam o comportamento do recurso de gerenciamento de planos de consultas e como ele afeta o otimizador. Para obter informações sobre como configurar o gerenciamento de planos de consultas, consulte Ativar o gerenciamento de planos de consulta do Aurora PostgreSQL. A alteração dos parâmetros a seguir não terá efeito se a extensão `apg_plan_mgmt` não for configurada conforme detalhado nessa seção. Para obter informações sobre como modificar parâmetros, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora e Grupos de parâmetros de banco de dados para instâncias de banco de dados do Amazon Aurora.

Parâmetros

- apg_plan_mgmt.capture_plan_baselines
- apg_plan_mgmt.plan_capture_threshold
- apg_plan_mgmt.explain_hashes
- apg_plan_mgmt.log_plan_enforcement_result
- apg_plan_mgmt.max_databases
- apg_plan_mgmt.max_plans

- apg_plan_mgmt.plan_hash_version
- apg_plan_mgmt.plan_retention_period
- apg_plan_mgmt.unapproved_plan_execution_threshold
- apg_plan_mgmt.use_plan_baselines
- auto_explain.hashes

apg_plan_mgmt.capture_plan_baselines

Captura os planos de execução de consultas gerados pelo otimizador para cada instrução SQL e os armazena na exibição `dba_plans`. Por padrão, o número máximo de planos que podem ser armazenados é 10.000, conforme especificado pelo parâmetro `apg_plan_mgmt.max_plans`. Para obter informações de referência, consulte apg_plan_mgmt.max_plans.

É possível definir esse parâmetro no grupo de parâmetros do cluster de banco de dados personalizado ou no grupo de parâmetros do banco de dados personalizado. Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
desligar	automatic	Aplique essa configuração no nível da sessão ou em um grupo de parâmetros para capturar planos que são usados duas ou mais vezes.
	manual	Aplique essa configuração no nível da sessão ou em um grupo de parâmetros para capturar planos que são usados uma ou mais vezes.
	desligar	Desabilita a captura de planos.

Para obter mais informações, consulte Capturar planos de execução do Aurora PostgreSQL.

apg_plan_mgmt.plan_capture_threshold

Especifica um limite para que, se o custo total do plano de execução da consulta estiver abaixo do limite, o plano não seja capturado na exibição. `apg_plan_mgmt.dba_plans`

Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
0	0 - 1.79769e+308	Define o limite do custo total de execução do plano de consulta `apg_plan_mgmt` para captura de planos.

Para obter mais informações, consulte Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans.

apg_plan_mgmt.explain_hashes

Especifica se `EXPLAIN [ANALYZE]` mostra `sql_hash` e `plan_hash` no final de sua saída. Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
0	0 (desativado)	EXPLAIN não mostra sql_hash e plan_hash sem a opção hashes true.
	1 (ativado)	EXPLAIN mostra sql_hash e plan_hash sem a opção hashes true.

apg_plan_mgmt.log_plan_enforcement_result

Especifica se os resultados precisam ser registrados para verificar se os planos gerenciados pelo QPM são usados corretamente. Quando um plano genérico armazenado é usado, não haverá registros gravados nos arquivos de log. Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
nenhuma	nenhuma	Não mostra nenhum resultado de aplicação do plano nos arquivos de log.
	on_error	Mostra apenas o resultado de aplicação do plano em arquivos de log quando o QPM não usa planos gerenciados.

Padrão	Valores permitidos	Descrição
	todas	Mostra todos os resultados de aplicação do plano em arquivos de log, incluindo êxitos e falhas.

apg_plan_mgmt.max_databases

Especifica o número máximo de bancos de dados na instância Writer do seu cluster de banco de dados do Aurora PostgreSQL que podem usar o gerenciamento de planos de consultas. Por padrão, até 10 bancos de dados podem usar o gerenciamento de planos de consultas. Se você tiver mais de 10 bancos de dados na instância, poderá alterar o valor dessa configuração. Para descobrir quantos bancos de dados existem em determinada instância, conecte-se à instância usando `psql`. Depois, use o metacomando psql, `\l`, para listar os bancos de dados.

Alterar o valor desse parâmetro exige que você reinicialize a instância para que a configuração entre em vigor.

Padrão	Valores permitidos	Descrição
10	10-2147483647	Número máximo de bancos de dados que podem usar o gerenciamento de planos de consultas na instância.

É possível definir esse parâmetro no grupo de parâmetros do cluster de banco de dados personalizado ou no grupo de parâmetros do banco de dados personalizado.

apg_plan_mgmt.max_plans

Defina o número máximo de instruções SQL que o gerenciador de planos de consulta pode manter na visualização `apg_plan_mgmt.dba_plans`. Recomendamos definir esse parâmetro como `10000` ou superior para todas as versões do Aurora PostgreSQL.

É possível definir esse parâmetro no grupo de parâmetros do cluster de banco de dados personalizado ou no grupo de parâmetros do banco de dados personalizado. Alterar o valor desse parâmetro exige que você reinicialize a instância para que a configuração entre em vigor.

Padrão	Valores permitidos	Descrição
10000	10-2147483647	Número máximo de planos que podem ser armazenados na exibição `apg_plan_mgmt.dba_plans` . O padrão para o Aurora PostgreSQL versão 10 e anteriores é 1000.

Para obter mais informações, consulte Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans.

apg_plan_mgmt.plan_hash_version

Especifica os casos de uso que o cálculo de plan_hash foi desenvolvido para atender. Uma versão superior de `apg_plan_mgmt.plan_hash_version` contém todas as funcionalidades da versão inferior. Por exemplo, a versão 3 atende aos casos de uso compatíveis com a versão 2.

A alteração do valor desse parâmetro deve ser seguida de uma chamada para `apg_plan_mgmt.validate_plans('update_plan_hash')`. Ela atualiza os valores de plan_hash em cada banco de dados com apg_plan_mgmt instalado e entradas na tabela de planos. Para obter mais informações, consulte . Validar planos

Padrão	Valores permitidos	Descrição
1	1	Cálculo padrão de plan_hash.
	2	Cálculo de plan_hash modificado para oferecer suporte a vários esquemas.
	3	Cálculo de plan_hash modificado para oferecer suporte a vários esquemas e a tabelas particionadas.
	4	Cálculo de plan_hash modificado para oferecer suporte a nós de materialização e a operadores paralelos.

apg_plan_mgmt.plan_retention_period

Especifica o número de dias em que os planos serão mantidos na exibição `apg_plan_mgmt.dba_plans`; serão excluídos automaticamente depois disso. Por padrão, um plano é excluído quando tiverem decorrido 32 dias desde o último uso do plano (a coluna `last_used` na exibição `apg_plan_mgmt.dba_plans`). Você pode alterar essa configuração para qualquer número, 1 ou mais.

Alterar o valor desse parâmetro exige que você reinicialize a instância para que a configuração entre em vigor.

Padrão	Valores permitidos	Descrição
32	1-2147483647	Número máximo de dias desde o último uso de um plano antes que seja excluído.

Para obter mais informações, consulte Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans.

apg_plan_mgmt.unapproved_plan_execution_threshold

Especifica um limite de custo abaixo do qual um plano Não aprovado pode ser usado pelo otimizador. Por padrão, o limite é 0, portanto o otimizador não executa planos com status Não aprovado. Definir esse parâmetro como um limite de custo trivialmente baixo, como 100, evita a sobrecarga de fiscalização do plano em planos triviais. Você também pode definir esse parâmetro como um valor extremamente grande, como 10000000, usando o estilo reativo do gerenciamento do plano. Isso permite que o otimizador use todos os planos escolhidos sem sobrecarga de fiscalização do plano. Mas, quando um plano ruim é encontrado, você pode marcá-lo manualmente como “rejeitado” para que não seja usado na próxima vez.

O valor desse parâmetro representa uma estimativa de custo para execução de determinado plano. Se um plano Não aprovado estiver abaixo desse custo estimado, o otimizador o usará para a instrução SQL. Você pode ver os planos capturados e seu status (Aprovado, Não aprovado) na exibição `dba_plans`. Para saber mais, consulte Examinar planos de consulta do Aurora PostgreSQL na exibição dba_plans.

Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
0	0-2147483647	Estimativa de custo do plano abaixo da qual um plano Não aprovado é usado.

Para obter mais informações, consulte Usar planos gerenciados do Aurora PostgreSQL.

apg_plan_mgmt.use_plan_baselines

Especifica que o otimizador deve usar um dos planos capturados e armazenados com status Aprovado na exibição `apg_plan_mgmt.dba_plans`. Por padrão, esse parâmetro está desativado (false), fazendo com que o otimizador use o plano de custo mínimo que ele gera sem qualquer avaliação adicional. Ativar esse parâmetro (configurá-lo como true) força o otimizador a escolher um plano de execução de consulta para a instrução a partir da linha de base do plano. Para obter mais informações, consulte Usar planos gerenciados do Aurora PostgreSQL. Para encontrar uma imagem que detalha esse processo, consulte Como o otimizador escolhe que plano executar..

É possível definir esse parâmetro no grupo de parâmetros do cluster de banco de dados personalizado ou no grupo de parâmetros do banco de dados personalizado. Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
false	verdadeiro	Use um plano Aprovado, Preferencial ou Não aprovado de `apg_plan_mgmt.dba_plans` . Se nenhum deles atender a todos os critérios de avaliação do otimizador, ele poderá usar seu próprio plano de custo mínimo gerado. Para obter mais informações, consulte Como o otimizador escolhe que plano executar..
	false	Use o plano de custo mínimo gerado pelo otimizador.

Você pode avaliar os tempos de resposta de diferentes planos capturados e alterar o status do plano, conforme necessário. Para obter mais informações, consulte Melhorar os planos de consulta do Aurora PostgreSQL.

auto_explain.hashes

Especifica se a saída de auto_explain mostra sql_hash e plan_hash. Alterar o valor desse parâmetro não requer uma reinicialização.

Padrão	Valores permitidos	Descrição
0 (desativado)	0 (desativado)	O resultado de `auto_explain` não mostra `sql_hash` nem `plan_hash` .
	1 (ativado)	O resultado de `auto_explain` mostra `sql_hash` e `plan_hash` .

Referência de funções do gerenciamento de planos de consultas do Aurora PostgreSQL

A extensão `apg_plan_mgmt` fornece as funções a seguir.

Funções

- apg_plan_mgmt.copy_outline
- apg_plan_mgmt.delete_plan
- apg_plan_mgmt.evolve_plan_baselines
- apg_plan_mgmt.get_explain_plan
- apg_plan_mgmt.plan_last_used
- apg_plan_mgmt.reload
- apg_plan_mgmt.set_plan_enabled
- apg_plan_mgmt.set_plan_status
- apg_plan_mgmt.update_plans_last_used
- apg_plan_mgmt.validate_plans

apg_plan_mgmt.copy_outline

Copie um determinado hash do plano SQL e o esboço do plano para um hash e um esboço do plano SQL de destino, substituindo assim o hash e o esboço do plano de destino. Essa função está disponível no `apg_plan_mgmt` 2.3 e superiores.

Sintaxe

```
apg_plan_mgmt.copy_outline(
    source_sql_hash,
    source_plan_hash,
    target_sql_hash,
    target_plan_hash,
    force_update_target_plan_hash
)
```

Valor de retorno

Retorna 0 quando a cópia é bem-sucedida. Gera exceções para entradas inválidas.

Parâmetros

Parâmetro	Descrição
`source_sql_hash`	O ID `sql_hash` associado ao `plan_hash` a ser copiado na consulta de destino.
`source_plan_hash`	O ID `plan_hash` a ser copiado na consulta de destino.
`target_sql_hash`	O ID `sql_hash` da consulta a ser atualizada com o hash e o esboço do plano de origem.
`target_plan_hash`	O ID `plan_hash` da consulta a ser atualizada com o hash e o esboço do plano de origem.
`force_update_target_plan_hash`	(Opcional) O ID `target_plan_hash` da consulta é atualizado mesmo que o plano de origem não seja reproduzível para o `target_sql_hash` . Quando definida como verdadeira, a função pode ser usada para copiar planos em esquemas em que os nomes e as colunas das relações são consistentes.

Observações de uso

Essa função permite que você copie um hash do plano e um esboço do plano que usam dicas para outras declarações semelhantes e, assim, evita que você precise usar instruções de dica em linha em cada ocorrência nas declarações de destino. Se a consulta de destino atualizada ocasionar um plano inválido, essa função gerará um erro e reverterá a tentativa de atualização.

apg_plan_mgmt.delete_plan

Exclua um plano gerenciado.

Sintaxe

```
apg_plan_mgmt.delete_plan(
    sql_hash,
    plan_hash
)
```

Valor de retorno

Retorna 0 caso a exclusão tenha sido bem-sucedida ou -1 em caso de falha na exclusão.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado.

apg_plan_mgmt.evolve_plan_baselines

Verifica se um plano já aprovado é mais rápido ou se um plano identificado pelo otimizador de consultas como um plano de custo mínimo é mais rápido.

Sintaxe

```
apg_plan_mgmt.evolve_plan_baselines(
    sql_hash,
    plan_hash,
```

```
    min_speedup_factor,
    action
)
```

Valor de retorno

O número de planos que não eram mais rápidos do que o melhor plano aprovado.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado. Use NULL como média de todos os planos que tenham o mesmo valor de ID `sql_hash`.
`min_speed up_factor`	O fator de agilização mínimo pode ser o número de vezes mais rápido que um plano deve ser em relação ao melhor dos planos já aprovados para aprová-lo. Como alternativa, esse fator pode ser o número de vezes mais lento que um plano deve ser para rejeitá-lo ou desabilitá-lo. Trata-se de um valor flutuante positivo.
`action`	A ação que a função deve realizar. Entre os valores válidos estão os seguintes. O uso de maiúsculas ou minúsculas não importa. • `'disable'` – desabilite cada plano correspondente que não atenda ao fator de agilização mínimo. • `'approve'` – habilite cada plano correspondente que atenda ao fator de agilização mínimo e defina seu status como `approved`. • `'reject'` – para cada plano correspondente que não atenda ao fator de agilização mínimo, defina o status como `rejected`. • NULL – a função simplesmente retorna o número de planos sem benefícios quanto à performance porque eles não atendem ao fator de agilização mínimo.

Observações de uso

Defina planos especificados como aprovados, rejeitados ou desabilitados com base na velocidade do planejamento mais o tempo de execução ser maior ou não que o melhor plano aprovado por um fator definido por você. O parâmetro de ação pode ser definido como `'approve'` ou `'reject'` para aprovar ou rejeitar automaticamente um plano que atenda aos critérios de performance. Como alternativa, ele pode ser definido como '' (string vazia) para realizar o experimento de performance e produzir um relatório, mas não realizar ação.

Você pode evitar a reexecução infundada da função `apg_plan_mgmt.evolve_plan_baselines` de um plano no qual ela foi executada recentemente. Para isso, restrinja os planos a apenas os planos não aprovados criados recentemente. Também é possível evitar executar a função `apg_plan_mgmt.evolve_plan_baselines` em qualquer plano aprovado que tenha um timestamp `last_verified` recente.

Realize um experimento de performance para comparar o tempo de planejamento mais execução de cada plano relativo aos outros planos na linha de base. Em alguns casos, talvez haja somente um plano para uma instrução e o plano seja aprovado. Nesse caso, compare o tempo de planejamento mais execução do plano com o tempo de planejamento mais execução de nenhum plano usado.

O benefício incremental (ou desvantagem) de cada plano é registrado na visualização `apg_plan_mgmt.dba_plans` na coluna `total_time_benefit_ms`. Quando esse valor é positivo, há uma vantagem de performance mensurável em incluir esse plano na linha de base.

Além de coletar o tempo de planejamento e execução de cada plano candidato, a coluna `last_verified` da visualização `apg_plan_mgmt.dba_plans` é atualizada com o `current_timestamp`. O time stamp `last_verified` pode ser usado para evitar a reexecução dessa função em um plano que recentemente teve a verificação da performance.

apg_plan_mgmt.get_explain_plan

Gera o texto de uma instrução `EXPLAIN` para a instrução SQL especificada.

Sintaxe

```
apg_plan_mgmt.get_explain_plan(
    sql_hash,
    plan_hash,
    [explainOptionList]
)
```

Valor de retorno

Retorna estatísticas de tempo de execução para as instruções SQL especificadas. Use sem `explainOptionList` para retornar um plano `EXPLAIN` simples.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado.
`explainOptionList`	Uma lista separada por vírgulas com opções de explicação. Os valores válidos incluem `'analyze'` , `'verbose'` , `'buffers'` , `'hashes'` e `'format json'`. Se a lista de `explainOptionList` for NULL (NULA) ou uma string vazia (''), essa função gerará uma instrução `EXPLAIN` sem nenhuma estatística.

Observações de uso

Para a `explainOptionList`, você pode usar qualquer uma das mesmas opções que você usaria com uma instrução `EXPLAIN`. O otimizador do Aurora PostgreSQL concatena a lista de opções que você fornece à instrução `EXPLAIN`.

apg_plan_mgmt.plan_last_used

Retorna a data `last_used` do plano especificado da memória compartilhada.

> **Note**
>
> O valor na memória compartilhada é sempre atual na instância de banco de dados primária do cluster de banco de dados. Esse valor é apenas periodicamente liberado na coluna `last_used` da visualização `apg_plan_mgmt.dba_plans`.

Sintaxe

```
apg_plan_mgmt.plan_last_used(
    sql_hash,
    plan_hash
)
```

Valor de retorno

Retorna a data `last_used`.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado.

apg_plan_mgmt.reload

Recarregue planos na memória compartilhada da visualização `apg_plan_mgmt.dba_plans`.

Sintaxe

```
apg_plan_mgmt.reload()
```

Valor de retorno

Nenhum.

Parâmetros

Nenhum.

Observações de uso

Chame `reload` para as seguintes situações:

- Use-o para atualizar a memória compartilhada de uma réplica somente leitura imediatamente, em vez de aguardar a propagação de novos planos para a réplica.

- Use-o após a importação de planos gerenciados.

apg_plan_mgmt.set_plan_enabled

Habilite ou desabilite um plano gerenciado.

Sintaxe

```
apg_plan_mgmt.set_plan_enabled(
    sql_hash,
    plan_hash,
    [true | false]
)
```

Valor de retorno

Retorna 0 caso a definição tenha sido bem-sucedida ou -1 em caso de falha na definição.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado.
`enabled`	Valores boolianos de verdadeiro ou falso: • Um valor de `true` habilita o plano. • Um valor de `false` desabilita o plano.

apg_plan_mgmt.set_plan_status

Defina o status de um plano gerenciado como `Approved`, `Unapproved`, `Rejected`, ou `Preferred`.

Sintaxe

```
apg_plan_mgmt.set_plan_status(
    sql_hash,
    plan_hash,
    status
)
```

Valor de retorno

Retorna 0 caso a definição tenha sido bem-sucedida ou -1 em caso de falha na definição.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado.
`status`	Uma string com um dos seguintes valores: • `'Approved'` • `'Unapproved'` • `'Rejected'` • `'Preferred'` A formatação de maiúsculas/minúsculas usada não importa. No entanto, o valor do status é definido com maiúsculas iniciais na visualização `apg_plan_mgmt.dba_plans`. Para obter mais informações sobre esses valores, consulte `status` em Referência da visualização apg_plan_mgmt.dba_plans da edição compatível do Aurora PostgreSQL.

apg_plan_mgmt.update_plans_last_used

Atualiza imediatamente a tabela de planos com a data de `last_used` armazenada na memória compartilhada.

Sintaxe

```
apg_plan_mgmt.update_plans_last_used()
```

Valor de retorno

Nenhum.

Parâmetros

Nenhum.

Observações de uso

Chame `update_plans_last_used` para garantir que as consultas na coluna `dba_plans.last_used` usem as informações mais atuais. Se a data `last_used` não for atualizada imediatamente, um processo em segundo plano atualizará a tabela de planos com a data de `last_used` uma vez a cada hora (por padrão).

Por exemplo, se uma instrução com um determinado `sql_hash` começar a ser executada lentamente, será possível determinar quais planos para essa instrução foram executados desde que a regressão de performance começou. Para fazer isso, primeiro descarregue os dados da memória compartilhada no disco para que as datas de `last_used` sejam atuais e, em seguida, consulte todos os planos de `sql_hash` da instrução com a regressão de performance. Na consulta, certifique-se de que a data de `last_used` seja igual ou posterior à data em que a regressão de performance começou. A consulta identifica o plano ou o conjunto de planos que pode ser o responsável pela regressão de performance. Você pode utilizar `apg_plan_mgmt.get_explain_plan` com `explainOptionList` definido como `verbose, hashes`. Também é possível utilizar `apg_plan_mgmt.evolve_plan_baselines` para analisar o plano e quaisquer planos alternativos que possam ter performance melhor.

A função `update_plans_last_used` tem efeito somente na instância de banco de dados primária do cluster de banco de dados.

apg_plan_mgmt.validate_plans

Valide se o otimizador ainda pode recriar planos. O otimizador valida os planos `Approved`, `Unapproved` e `Preferred`, independentemente de o plano estar ativado ou desativado. Planos `Rejected` não são validados. Também é possível usar a função `apg_plan_mgmt.validate_plans` para excluir ou desabilitar planos inválidos.

Sintaxe

```
apg_plan_mgmt.validate_plans(
    sql_hash,
    plan_hash,
    action)

apg_plan_mgmt.validate_plans(
    action)
```

Valor de retorno

O número de planos inválidos.

Parâmetros

Parâmetro	Descrição
`sql_hash`	O ID `sql_hash` da instrução SQL gerenciada do plano.
`plan_hash`	O ID `plan_hash` do plano gerenciado. Use NULL como média de todos os planos para o mesmo valor de ID `sql_hash`.
`action`	A ação que a função deve realizar em planos inválidos. Entre os valores de string válidos estão os seguintes. O uso de maiúsculas ou minúsculas não importa. • `'disable'` – todo plano inválido é desabilitado. • `'delete'` – todo plano inválido é excluído. • `'update_plan_hash'` – Atualiza o `plan_hash` ID para planos que não podem ser reproduzidos de forma exata. Ele também permite que você corrija um plano reescrevendo o SQL. Depois, você pode registrar o plano bom como um plano `Approved` para o SQL original. • NULL – a função simplesmente retorna o número de planos inválidos. Nenhuma outra ação é realizada. • '' – uma string vazia produz uma mensagem que indica o número de planos válidos e inválidos. Todos os outros valores são tratados como strings vazias.

Observações de uso

Use a forma `validate_plans(action)` a fim de validar todos os planos gerenciados para todas as instruções gerenciadas em toda a exibição `apg_plan_mgmt.dba_plans`.

Use a forma `validate_plans(sql_hash, plan_hash, action)` para validar um plano gerenciado especificado com `plan_hash`, para uma instrução gerenciada especificada com `sql_hash`.

Use a forma `validate_plans(sql_hash, NULL, action)` a fim de validar todos os planos gerenciados para a instrução gerenciada especificada com `sql_hash`.

Referência da visualização apg_plan_mgmt.dba_plans da edição compatível do Aurora PostgreSQL

As colunas das informações do plano na visualização `apg_plan_mgmt.dba_plans` incluem o seguinte.

Coluna dba_plans	Descrição
`cardinality_error`	Uma medida do erro entre a cardinalidade estimada em comparação com a cardinalidade real. Cardinalidade é o número de linhas da tabela que o plano deve processar. Se a cardinalidade for grande, aumentará a probabilidade de o plano não ser ideal. Essa coluna é preenchida pela função apg_plan_mgmt.evolve_plan_baselines.
`compatibility_level`	Esse parâmetro mostra quando um plano de consulta foi validado pela última vez. Nas versões 12.19, 13.15, 14.12, 15.7, 16.3 e posteriores do Aurora PostgreSQL, ele exibe o número da versão do Aurora. Para versões anteriores, ele exibe um número de versão específico do recurso. **Note** Mantenha o valor desse parâmetro na configura ção padrão. O Aurora PostgreSQL define e atualiza automaticamente esse valor.

Coluna dba_plans	Descrição
`created_by`	O usuário autenticado (`session_user`) que criou o plano.
`enabled`	Um indicador da habilitação ou de desabilitação do plano. Todos os planos permanecem habilitados por padrão. Desabilite os planos para evitar que eles sejam usados pelo otimizador. Para modificar esse valor, use a função apg_plan_mgmt.set_plan_enabled.
`environment_variables`	Os parâmetros Grand Unified Configuration (GUC – Grande configuração unificada) PostgreSQL e os valores substituídos pelo otimizados no momento em que o plano foi capturado.
`estimated_startup_cost`	O custo de configuração do otimizador estimado antes da entrega de uma tabela pelo otimizador.
`estimated_total_cost`	O custo de entrega do otimizador estimado da linha da tabela final.
`execution_time_benefit_ms`	O benefício do tempo de execução, em milissegundos, da habilitação do plano. Essa coluna é preenchida pela função apg_plan_mgmt.evolve_plan_baselines.
`execution_time_ms`	O tempo estimado, em milissegundos, de execução do plano. Essa coluna é preenchida pela função apg_plan_mgmt.evolve_plan_baselines.
`has_side_effects`	Um valor que indica que a instrução SQL é uma instrução Data Manipulation Language (DML) ou uma instrução SELECT que contém uma função VOLATILE.

Coluna dba_plans	Descrição
`last_used`	Esse valor é atualizado para a data atual sempre que o plano é executado ou quando o plano é o de custo mínimo do otimizador de consultas. Esse valor é armazenado em memória compartilhada e enviado periodicamente para o disco. Para obter o valor mais atualizado, leia a data da memória compartilhada chamando a função `apg_plan_mgmt.plan_last_used(sql_hash, plan_hash)`, em vez de ler o valor `last_used`. Para obter informações adicionais, consulte o parâmetro apg_plan_mgmt.plan_retention_period.
`last_validated`	A data e a hora mais recentes quando se verificou que o plano poderia ser recriado pela função apg_plan_mgmt.validate_plans ou pela função apg_plan_mgmt.evolve_plan_baselines.
`last_verified`	A data e a hora mais recentes quando se verificou que um plano era o de melhor performance para o parâmetro especificado pela função apg_plan_mgmt.evolve_plan_baselines.
`origin`	Como o plano foi capturado com o parâmetro apg_plan_mgmt.capture_plan_baselines. Entre os valores válidos estão os seguintes: `M` – O plano foi capturado com a captura de planos manual. `A` – O plano foi capturado com a captura de planos automática.
`param_list`	Os valores de parâmetro que foram passados para a instrução, caso esta seja uma instrução preparada.
`plan_created`	A data e a hora em que o plano foi criado.
`plan_hash`	O identificador do plano. A combinação de `plan_hash` e `sql_hash` identifica com exclusividade um plano específico.

Coluna dba_plans	Descrição
`plan_outline`	Uma representação do plano usado para recriar o plano de execução real e que seja independentemente em relação ao banco de dados. Os operadores na árvore correspondem aos operadores exibidos na saída EXPLAIN.
`planning_time_ms`	O tempo real de execução do planejador, em milissegundos. Essa coluna é preenchida pela função apg_plan_mgmt.evolve_plan_baselines.
`queryId`	Um hash de instrução, conforme calculado pela extensão `pg_stat_statements` . Não se trata de um identificador estável ou independente de banco de dados porque ele depende de Object Identifiers (OIDs – Identificadores de objetos). O valor será `0` se `compute_query_id` for `off` ao capturar o plano de consulta.
`sql_hash`	Um valor de hash do texto de instrução SQL, normalizado sem literais.
`sql_text`	O texto completo da instrução SQL.

Coluna dba_plans	Descrição
`status`	O status de um plano, que determina como o otimizador usa um plano. Entre os valores válidos estão os seguintes. • `Approved` – um plano utilizável que o otimizador pode optar por executar. O otimizador executa o plano de menor custo com base em um conjunto de planos aprovados da instrução gerenciada (linha de base). Para redefinir um plano a ser aprovado, use a função apg_plan_mgmt.evolve_plan_baselines. • `Unapproved` – um plano capturado cujo uso não foi verificado. Para obter mais informações, consulte Avaliar performance do plano. • `Rejected` – um plano que o otimizador não usará. Para obter mais informações, consulte Rejeitar ou desabilitar planos mais lentos. • `Preferred` – um plano que você determinou como preferido a ser usado em uma instrução gerenciada. Se o plano de custo mínimo do otimizador não for um plano aprovado ou preferencial, você poderá reduzir o custo indireto da imposição do plano. Para fazer isso, crie um subconjunto dos planos aprovados `Preferred`. Quando o custo mínimo do otimizador não for um plano `Approved`, um plano `Preferred` será escolhido antes de um plano `Approved`. Para redefinir um plano como `Preferred`, use a função apg_plan_mgmt.set_plan_status.
`stmt_name`	O nome da instrução SQL dentro de uma instrução PREPARE. Este valor é uma string vazia para uma instrução preparada não nomeada. Este valor é NULL para uma instrução não preparada.

Coluna dba_plans	Descrição
`total_time_benefit_ms`	O benefício do tempo total, em milissegundos, da habilitação desse plano. Este valor considera o tempo de planejamento e o tempo de execução. Caso esse valor seja negativo, há uma desvantagem em habilitar esse plano. Essa coluna é preenchida pela função apg_plan_mgmt.evolve_plan_baselines.

Atributos avançados do gerenciamento de planos de consultas

A seguir, você encontrará informações sobre os atributos avançados do gerenciamento de planos de consulta (QPM) do Aurora PostgreSQL:

Tópicos

- Capturar planos de execução nas réplicas do Aurora PostgreSQL
- Compatibilidade ao particionamento de tabelas

Capturar planos de execução nas réplicas do Aurora PostgreSQL

O QPM (gerenciamento de planos de consulta) permite capturar os planos de consulta gerados pelas réplicas do Aurora e armazená-los na instância de banco de dados primária do cluster de banco de dados do Aurora. Você pode coletar os planos de consulta de todas as réplicas do Aurora e manter um conjunto de planos ideais em uma tabela persistente central na instância primária. Então, você pode aplicar esses planos a outras réplicas quando necessário. Isso ajuda você a manter a estabilidade dos planos de execução e melhorar a performance das consultas nos clusters de banco de dados e nas versões do mecanismo.

Tópicos

- Pré-requisitos
- Gerenciar a captura de planos em réplicas do Aurora
- Solução de problemas

Pré-requisitos

Ativar **`capture_plan_baselines parameter`** na réplica do Aurora: defina o parâmetro `capture_plan_baselines` como automático ou manual para capturar os planos nas réplicas do Aurora. Para ter mais informações, consulte apg_plan_mgmt.capture_plan_baselines.

Instalar a extensão postgres_fdw: você deve instalar a extensão `postgres_fdw` externa do wrapper de dados para capturar os planos nas réplicas do Aurora. Para instalar a extensão, execute o command a seguir o em cada banco de dados.

```
postgres=> CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Gerenciar a captura de planos em réplicas do Aurora

Ativar a captura de planos em réplicas do Aurora

Você deve ter privilégios de `rds_superuser` para criar ou remover a captura de planos nas réplicas do Aurora. Para obter mais informações sobre funções e permissões de usuário, consulte Noções básicas de perfis e permissões do PostgreSQL.

Para capturar planos, chame a função apg_plan_mgmt.create_replica_plan_capture na instância de banco de dados de gravação, conforme mostrado a seguir:

```
postgres=> CALL apg_plan_mgmt.create_replica_plan_capture('endpoint', 'password');
```

- endpoint: o endpoint de gravador ou o cluster_endpoint do Aurora Global Database oferece suporte a failover para captura de planos em réplicas do Aurora.

  Para ter mais informações sobre o endpoint de gravador do Aurora Global Database, consulte Visualizar os endpoints de um Amazon Aurora Global Database.

  Para ter mais informações sobre endpoints do cluster, consulte Endpoints de cluster para Amazon Aurora.
- Senha: siga as diretrizes abaixo ao criar a senha para aumentar a segurança:
  - Deve conter pelo menos oito caracteres.
  - Deve conter pelo menos uma letra maiúscula, uma letra minúscula e um número.
  - Deve ter pelo menos um caractere especial (?, !, #, <, >, * e assim por diante).

> **Note**
>
> Se você alterar o endpoint, a senha ou o número da porta, deverá executar `apg_plan_mgmt.create_replica_plan_capture()` novamente com o endpoint e a senha para reinicializar a captura de plano. Caso contrário, os planos de captura das réplicas do Aurora falharão.

Desativar a captura de planos em réplicas do Aurora

Você pode desativar o parâmetro `capture_plan_baselines` na réplica do Aurora definindo o respectivo valor como `off` no grupo de parâmetros.

Remover a captura de planos em réplicas do Aurora

Você pode remover completamente a captura de planos nas réplicas do Aurora, mas certifique-se antes de fazer isso. Para remover a captura de planos, chame `apg_plan_mgmt.remove_replica_plan_capture` conforme mostrado:

```
postgres=> CALL apg_plan_mgmt.remove_replica_plan_capture();
```

Você deve chamar apg_plan_mgmt.create_replica_plan_capture() novamente para ativar a captura de planos nas réplicas do Aurora com o endpoint e a senha.

Solução de problemas

A seguir vão algumas ideias e soluções alternativas de problemas caso o plano não seja capturado nas réplicas do Aurora conforme o esperado.

- Configurações de parâmetros: verifique se o parâmetro `capture_plan_baselines` está definido com o valor adequado para ativar a captura de planos.
- A extensão **`postgres_fdw`** está instalada: use a consulta a seguir para verificar se `postgres_fdw` está instalada.

  ```
  postgres=> SELECT * FROM pg_extension WHERE extname = 'postgres_fdw'
  ```

- create_replica_plan_capture() é chamado: use o comando a seguir para verificar se o mapeamento do usuário existe. Caso contrário, chame `create_replica_plan_capture()` para inicializar o atributo.

```
postgres=> SELECT * FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- Endpoint e número da porta: verifique se o endpoint e o número da porta são apropriados. Não haverá nenhuma mensagem de erro exibida se esses valores estiverem incorretos.

  Use o comando a seguir para verificar se o endpoint é usado no create() e para verificar em qual banco de dados ele reside:

```
postgres=> SELECT srvoptions FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- reload (): você deve chamar apg_plan_mgmt.reload() depois de chamar apg_plan_mgmt.delete_plan() nas réplicas do Aurora para tornar a função delete efetiva. Isso garante que a mudança seja implementada com sucesso.
- Senha: você deve inserir a senha em create_replica_plan_capture() de acordo com as diretrizes mencionadas. Caso contrário, você receberá uma mensagem de erro. Para obter mais informações, consulte Gerenciar a captura de planos em réplicas do Aurora. Use outra senha que esteja de acordo com os requisitos.
- Conexão entre regiões: a captura de planos nas réplicas do Aurora também é compatível com o banco de dados global do Aurora, onde a instância de gravação e as réplicas do Aurora podem estar em regiões diferentes. Use o endpoint de gravador do Aurora Global Database para manter a conectividade após eventos de failover ou transição. Para ter mais informações sobre os endpoints do Aurora Global Database, consulte Visualizar os endpoints de um Amazon Aurora Global Database. A instância de gravação e a réplica entre regiões devem ser capazes de se comunicar usando o emparelhamento de VPC. Para obter mais informações, consulte Emparelhamento de VPC. Se ocorrer um failover entre regiões, você deverá reconfigurar o endpoint com o novo endpoint primário do cluster de banco de dados.

> **Note**
>
> Ao usar um endpoint de cluster em vez de um endpoint de gravador do Aurora Global Database, você precisará atualizar o endpoint de cluster depois de realizar uma operação de failover ou transição global.

Compatibilidade ao particionamento de tabelas

O gerenciamento de planos de consulta (QPM) do Aurora PostgreSQL comporta o particionamento de tabelas nas seguintes versões:

- 15.3 e versões 15 posteriores
- 14.8 e versões 14 posteriores
- 13.11 e versões 13 posteriores

Para obter mais informações, consulte Particionamento de tabelas.

Tópicos

- Configurar o particionamento de tabelas
- Capturar planos para o particionamento de tabelas
- Aplicar um plano de particionamento de tabelas
- Convenção de nomenclatura

Configurar o particionamento de tabelas

Para configurar o particionamento de tabelas no QPM do Aurora PostgreSQL, faça o seguinte:

1. Defina `apg_plan_mgmt.plan_hash_version` como 3 ou mais no grupo de parâmetros do cluster de banco de dados.
2. Navegue até um banco de dados que usa o Gerenciamento de Planos de Consulta e que tem entradas na visualização de `apg_plan_mgmt.dba_plans`.
3. Ligue para `apg_plan_mgmt.validate_plans('update_plan_hash')` a fim de atualizar o valor de `plan_hash` na tabela de planos.
4. Repita as etapas 2 e 3 para todos os bancos de dados com o Gerenciamento de Planos de Consulta ativado e que tenham entradas na visualização `apg_plan_mgmt.dba_plans`.

Para obter mais informações sobre esses parâmetros, consulte Referência de parâmetros do gerenciamento de planos de consultas do Aurora PostgreSQL.

Capturar planos para o particionamento de tabelas

No QPM, planos diferentes são diferenciados pelo valor de `plan_hash`. Para entender como `plan_hash` muda, primeiro é necessário conhecer tipos semelhantes de planos.

A combinação de métodos de acesso, nomes de índice sem dígitos e nomes de partição sem dígitos, acumulados no nó de anexação, deve ser constante para que os planos sejam considerados iguais. As partições específicas acessadas nos planos não são significativas. No exemplo a seguir, uma tabela `tbl_a` é criada com quatro partições.

```
postgres=>create table tbl_a(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table tbl_a1 partition of tbl_a for values from (0) to (1000);
CREATE TABLE
postgres=>create table tbl_a2 partition of tbl_a for values from (1001) to (2000);
CREATE TABLE
postgres=>create table tbl_a3 partition of tbl_a for values from (2001) to (3000);
CREATE TABLE
postgres=>create table tbl_a4 partition of tbl_a for values from (3001) to (4000);
CREATE TABLE
postgres=>create index t_i on tbl_a using btree (i);
CREATE INDEX
postgres=>create index t_j on tbl_a using btree (j);
CREATE INDEX
postgres=>create index t_k on tbl_a using btree (k);
CREATE INDEX
```

Os planos a seguir são considerados iguais porque um único método de varredura está sendo usado para analisar `tbl_a`, independentemente do número de partições que a consulta procura.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 999 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------------
Seq Scan on tbl_a1 tbl_a
    Filter: ((i >= 990) AND (i <= 999) AND (j < 9910) AND (k > 50))
SQL Hash: 1553185667, Plan Hash: -694232056
(3 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;
```

```
                        QUERY PLAN
------------------------------------------------------------------
Append
    ->  Seq Scan on tbl_a1 tbl_a_1
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    ->  Seq Scan on tbl_a2 tbl_a_2
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    SQL Hash: 1553185667, Plan Hash: -694232056
    (6 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(8 rows)
```

Os três planos a seguir também são considerados iguais porque, no nível pai, os métodos de acesso, os nomes de índice sem dígitos e os nomes de partição sem dígitos são `SeqScan tbl_a`, `IndexScan (i_idx) tbl_a`.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_i_idx on tbl_a2 tbl_a_2
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
```

```
(7 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(11 rows)
```

Independentemente da diferença de ordem e número de ocorrências nas partições filho, os métodos de acesso, os nomes de índice sem dígitos e os nomes de partição sem dígitos são constantes no nível pai de cada um dos planos acima.

No entanto, os planos seriam considerados diferentes se alguma das condições a seguir fosse atendida:

- Algum método de acesso adicional é usado no plano.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 1134525070
(11 rows)
```

- Um dos métodos de acesso do plano não é mais usado.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(6 rows)
```

- O índice associado a um método de indexação é alterado.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;
```

```
                              QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_j_idx on tbl_a2 tbl_a_2
         Index Cond: (j < 9910)
         Filter: ((i >= 990) AND (i <= 1100) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993343726
(7 rows)
```

Aplicar um plano de particionamento de tabelas

Os planos aprovados para tabelas particionadas são aplicados com correspondência posicional. Os planos não são específicos para as partições e podem ser aplicados em partições diferentes dos planos referenciados na consulta original. Os planos também podem ser aplicados para consultas que acessam um número diferente de partições em comparação com o esquema original aprovado.

Por exemplo, se o esquema aprovado for para o seguinte plano:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                              QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

Depois, esse plano também pode ser aplicado em consultas SQL que fazem referência a duas, quatro ou mais partições. Os planos que poderiam surgir desses cenários para acessar duas e quatro partições são:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(8 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a4 tbl_a_4
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(12 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
```

```
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(14 rows)
```

Considere outro plano aprovado com métodos de acesso diferentes para cada partição:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 2032136998
(12 rows)
```

Nesse caso, qualquer plano que leia de duas partições não seria aplicado. A menos que todas as combinações (método de acesso, nome de índice) do plano aprovado sejam utilizáveis, o plano não pode ser aplicado. Por exemplo, os planos a seguir têm hashes de plano diferentes e o plano aprovado não pode ser aplicado nesses casos:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;
```

```
                              QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Bitmap Heap Scan on tbl_a1 tbl_a_1
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a1_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
   ->  Bitmap Heap Scan on tbl_a2 tbl_a_2
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a2_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
  Note: This is not an Approved plan.  No usable Approved plan was found.
  SQL Hash: 1553185667, Plan Hash: -568647260
(13 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1900) AND (j < 9910) AND (k > 50))
 Note: This is not an Approved plan.  No usable Approved plan was found.
 SQL Hash: 1553185667, Plan Hash: -496793743
(8 rows)
```

Convenção de nomenclatura

Para que o QPM aplique um plano com tabelas particionadas declarativas, é necessário seguir regras de nomenclatura específicas para tabelas principais, partições de tabelas e índices:

- Nomes de tabelas principais: esses nomes devem ser diferenciados por caracteres alfabéticos ou especiais, e não apenas por dígitos. Por exemplo, tA, tB e tC são nomes aceitáveis para tabelas pais distintas, enquanto t1, t2 e t3 não são.

- Nomes de tabelas de partição individuais: partições da mesma tabela principal devem diferir umas das outras somente por dígitos. Por exemplo, nomes de partição aceitáveis para tA podem ser tA1 e tA2, t1A e t2A, ou até mesmo vários dígitos.

  Quaisquer outras diferenças (letras, caracteres especiais) não garantirão a aplicação do plano.
- Nomes dos índices: na hierarquia das tabelas de partições, todos os índices deverão ter nomes exclusivos. Isso significa que as partes não numéricas dos nomes devem ser diferentes. Por exemplo, se você tiver uma tabela particionada denominada `tA` com um índice chamado `tA_col1_idx1`, não poderá ter outro índice denominado `tA_col1_idx2`. No entanto, é possível ter um índice denominado `tA_a_col1_idx2` porque a parte não numérica do nome é exclusiva. Essa regra aplica-se aos índices criados na tabela principal e nas tabelas de partição individuais.

O não cumprimento das convenções de nomenclatura acima pode resultar na falha na aplicação dos planos aprovados. O exemplo a seguir ilustra essa falha de aplicação:

```
postgres=>create table t1(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table t1a partition of t1 for values from (0) to (1000);
CREATE TABLE
postgres=>create table t1b partition of t1 for values from (1001) to (2000);
CREATE TABLE
postgres=>SET apg_plan_mgmt.capture_plan_baselines TO 'manual';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 0;

                           QUERY PLAN
-----------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on t1a t1_1
               Filter: (i > 0)
         ->  Seq Scan on t1b t1_2
               Filter: (i > 0)
 SQL Hash: -1720232281, Plan Hash: -1010664377
(7 rows)
```

```
postgres=>SET apg_plan_mgmt.use_plan_baselines TO 'on';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 1000;
```

```
                            QUERY PLAN
------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on t1b t1
         Filter: (i > 1000)
 Note: This is not an Approved plan. No usable Approved plan was found.
 SQL Hash: -1720232281, Plan Hash: 335531806
(5 rows)
```

Embora os dois planos pareçam idênticos, os valores `Plan Hash` são diferentes devido aos nomes das tabelas secundárias. Os nomes das tabelas variam de acordo com caracteres alfa, em vez de apenas dígitos, o que ocasiona uma falha de imposição.

Trabalhar com extensões e invólucros de dados externos

Para estender a funcionalidade ao cluster de banco de dados da edição compatível com o Aurora PostgreSQL, você pode instalar e usar várias extensões do PostgreSQL. Por exemplo, se seu caso de uso exigir muitas entradas de dados em tabelas muito grandes, você poderá instalar a extensão `pg_partman` para particionar os dados e, assim, distribuir a workload.

> **Note**
>
> A partir do Aurora PostgreSQL 14.5, o Aurora PostgreSQL é compatível com extensões de linguagem confiáveis para PostgreSQL. Esse recurso é implementado como a extensão `pg_tle`, que você pode adicionar ao seu Aurora PostgreSQL. Ao usar essa extensão, os desenvolvedores podem criar suas próprias extensões do PostgreSQL em um ambiente seguro que simplifica os requisitos de instalação e configuração, bem como grande parte dos testes preliminares de novas extensões. Para obter mais informações, consulte Trabalhar com Trusted Language Extensions para PostgreSQL.

Em alguns casos, em vez de instalar uma extensão, você pode adicionar um módulo específico à lista de `shared_preload_libraries` no grupo de parâmetros do cluster de banco de dados personalizado de seu cluster de banco de dados do Aurora PostgreSQL. Normalmente, o grupo de parâmetros padrão do cluster de banco de dados carrega somente as `pg_stat_statements`, mas vários outros módulos estão disponíveis para serem adicionados à lista. Por exemplo, você pode incluir a capacidade de agendamento adicionando o módulo `pg_cron`, conforme detalhado em Agendar manutenção com a extensão pg_cron do PostgreSQL. Como outro exemplo, você pode registrar planos de execução de consultas carregando o módulo `auto_explain`. Para saber mais, consulte Logging execution plans of queries (Registrar em log planos de execução de consultas) no Centro de Conhecimentos da AWS.

Uma extensão que fornece acesso a dados externos é conhecida mais especificamente como um invólucro de dados externos (FDW). Por exemplo, a extensão `oracle_fdw` permite que o cluster de banco de dados do Aurora PostgreSQL funcione com bancos de dados Oracle.

Também é possível especificar com precisão quais extensões podem ser instaladas na instância de banco de dados do Aurora PostgreSQL, listando-as no parâmetro `rds.allowed_extensions`. Para obter mais informações, consulte Restringir a instalação de extensões do PostgreSQL.

A seguir, você encontra informações sobre como configurar e usar algumas das extensões, módulos e FDWs disponíveis para o Aurora PostgreSQL. Por uma questão de simplicidade, todas elas são chamadas de “extensões”. Para encontrar listas das extensões que você pode usar com as versões do Aurora PostgreSQL atualmente disponíveis, consulte Extension versions for Amazon Aurora PostgreSQL (Versões de extensão para o Amazon Aurora PostgreSQL) nas Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

- Gerenciar objetos grandes com o módulo lo
- Gerenciar dados espaciais com a extensão PostGIS
- Gerenciar partições do PostgreSQL com a extensão pg_partman
- Agendar manutenção com a extensão pg_cron do PostgreSQL
- Usar pgAudit para registrar a atividade do banco de dados
- Usar pglogical para sincronizar dados entre instâncias
- Trabalhar com um banco de dados Oracle usando a extensão oracle_fdw
- Trabalhar com bancos de dados do SQL Server usando a extensão tds_fdw

Usar o suporte de extensão delegado do Amazon Aurora para PostgreSQL

Usando o suporte de extensão delegado do Amazon Aurora para PostgreSQL, é possível delegar o gerenciamento de extensões a um usuário que não precisa ser um `rds_superuser`. Com esse suporte de extensão delegado, um novo perfil chamado `rds_extension` é criado, e você deve atribuí-lo a um usuário para gerenciar outras extensões. Esse perfil pode criar, atualizar e eliminar extensões.

É possível especificar quais extensões podem ser instaladas na instância de banco de dados do Aurora PostgreSQL, listando-as no parâmetro `rds.allowed_extensions`. Para obter mais informações, consulte Usar extensões PostgreSQL com o Amazon RDS para PostgreSQL.

É possível restringir a lista de extensões disponíveis que podem ser gerenciadas pelo usuário com o perfil `rds_extension` usando o parâmetro `rds.allowed_delegated_extensions`.

O suporte de extensão delegado está disponível nas seguintes versões:

- Todas as versões posteriores
- 15.5 e versões 15 posteriores
- 14.10 e versões 14 posteriores

- 13.13 e versões 13 posteriores
- 12.17 e versões 12 posteriores

Tópicos

- Ativar o suporte de extensão delegado a um usuário
- Configuração usada no suporte de extensão delegado do Aurora para PostgreSQL
- Desativar o suporte para a extensão delegada
- Benefícios de usar o suporte de extensão delegado do Amazon Aurora
- Limitação do suporte de extensão delegado do Aurora para PostgreSQL
- Permissões necessárias para determinadas extensões
- Considerações sobre segurança
- Descartar cascata de extensão desabilitado
- Exemplos de extensões que podem ser adicionadas usando o suporte de extensão delegado

Ativar o suporte de extensão delegado a um usuário

É necessário executar o seguinte para habilitar o suporte de extensão delegado a um usuário:

1. Conceder o perfil **`rds_extension`** a um usuário: conecte-se ao banco de dados `rds_superuser` e execute o seguinte comando:

   ```
   Postgres => grant rds_extension to user_name;
   ```

2. Definir a lista de extensões disponíveis para usuários delegados gerenciarem: o `rds.allowed_delegated_extensions` permite que você especifique um subconjunto das extensões disponíveis usando o parâmetro de cluster de banco de dados `rds.allowed_extensions`. É possível fazer isso em um dos seguintes níveis:

   - No cluster ou no grupo de parâmetros da instância, por meio do Console de gerenciamento da AWS ou da API. Para obter mais informações, consulte Grupos de parâmetros para Amazon Aurora.
   - Use o seguinte comando em nível de banco de dados:

     ```
     alter database database_name set rds.allowed_delegated_extensions =
      'extension_name_1,
     ```

```
                    extension_name_2,...extension_name_n';
```

- Use o seguinte comando em nível de usuário:

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                    extension_name_2,...extension_name_n';
```

> **Note**
>
> Não é necessário reiniciar o banco de dados depois de alterar o parâmetro dinâmico `rds.allowed_delegated_extensions`.

3. Permitir que o usuário delegado acesse os objetos criados durante o processo de criação da extensão: determinadas extensões criam objetos que exigem que permissões adicionais sejam concedidas para que o usuário com o perfil `rds_extension` possa acessá-los. O `rds_superuser` deve conceder ao usuário delegado acesso a esses objetos. Uma das opções é usar um gatilho de eventos para conceder permissão automaticamente ao usuário delegado.

   Exemplo de gatilho de eventos

   Se quiser permitir que um usuário delegado com `rds_extension` use extensões que exijam permissões de configuração nos objetos criados pela criação da extensão, você poderá personalizar o exemplo abaixo de um gatilho de eventos e adicionar somente as extensões para as quais você deseja que os usuários delegados tenham acesso à funcionalidade completa. Esse gatilho de eventos pode ser criado no modelo 1 (o modelo padrão), portanto, todo banco de dados criado a partir do modelo 1 terá esse gatilho de eventos. Quando um usuário delegado instalar a extensão, esse gatilho concederá automaticamente a propriedade dos objetos criados pela extensão.

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;
```

```
  r RECORD;

BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_catalog.pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';

        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );

        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
 schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
 schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_gaz TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_lex TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_rules TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE pg_catalog.format('ALTER TEXT SEARCH DICTIONARY %I.intdict
 OWNER TO %I;', schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
```

```
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE
 ON TABLE %I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname,
 databaseowner);
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL
 TABLES IN SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE, SELECT ON ALL SEQUENCES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA
 topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE ON SCHEMA topology TO %I WITH
 GRANT OPTION;', databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE
 create_ext();
```

Configuração usada no suporte de extensão delegado do Aurora para PostgreSQL

Nome da configuração	Descrição	Valor padrão	Observações	Quem pode modificar ou conceder permissão
`rds.allowed_delegated_extensions`	Esse parâmetro limita as extensões que um perfil rds_extension pode gerenciar em um banco de dados. Ele deve ser um subconjunto de rds.allowed_extensions.	string vazia	• Por padrão, esse parâmetro é uma string vazia, o que significa que nenhuma extensão foi delegada aos usuários com `rds_extension` .	rds_superuser

Nome da configuração	Descrição	Valor padrão	Observações	Quem pode modificar ou conceder permissão
			• Qualquer extensão compatível poderá ser adicionada se o usuário tiver permissões para isso. Para fazer isso, defina o parâmetro `rds.allowed_delegated_extensions` como uma string de nomes de extensão separados por vírgulas. Ao adicionar uma lista de extensões a esse parâmetro, você deve identificar explicitamente as extensões que o usuário com o perfil `rds_extension` pode instalar. • Quando definido como `*`, significa	

Nome da configuração	Descrição	Valor padrão	Observações	Quem pode modificar ou conceder permissão
			que todas as extensões listadas em `rds_allowed_extensions` são delegadas aos usuários com o perfil `rds_extension` . Para saber mais sobre como configurar esse parâmetro, consulte Ativar o suporte de extensão delegado a um usuário.	

Nome da configuração	Descrição	Valor padrão	Observações	Quem pode modificar ou conceder permissão
`rds.allowed_extensions`	Esse parâmetro permite que o cliente limite as extensões que podem ser instaladas na instância de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Restringir a instalação de extensões do PostgreSQL.	"*"	Por padrão, esse parâmetro é definido como “*”, o que significa que todas as extensões aceitas no RDS para PostgreSQL e no Aurora PostgreSQL podem ser criadas por usuários com os privilégios necessários. Vazio significa que nenhuma extensão pode ser instalada na instância de banco de dados do Aurora PostgreSQL.	administrador

Nome da configuração	Descrição	Valor padrão	Observações	Quem pode modificar ou conceder permissão
`rds-delegated_extension_allow_drascac`	Esse parâmetro controla a capacidade do usuário com `rds_extension` de eliminar a extensão usando uma opção em cascata.	desligar	Por padrão, `rds-delegated_extension_allow_drop_cascade` é definido como `off`. Isso significa que os usuários com `rds_extension` não têm permissão para descartar uma extensão usando a opção em cascata. Para conceder essa capacidade, o parâmetro `rds.delegated_extension_allow_drop_cascade` deve ser definido como on.	rds_superuser

Desativar o suporte para a extensão delegada

Desativar parcialmente

Os usuários delegados não podem criar extensões, mas ainda podem atualizar as existentes.

- Redefina `rds.allowed_delegated_extensions` como o valor padrão no grupo de parâmetros de cluster de banco de dados.

- Use o seguinte comando em nível de banco de dados:

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- Use o seguinte comando em nível de usuário:

```
alter user user_name reset rds.allowed_delegated_extensions;
```

Desativar totalmente

Revogar o perfil `rds_extension` de um usuário vai restaurar as permissões padrão do usuário. O usuário não pode mais criar, atualizar nem descartar extensões.

```
postgres => revoke rds_extension from user_name;
```

Benefícios de usar o suporte de extensão delegado do Amazon Aurora

Usando o suporte de extensão delegado do Amazon Aurora para PostgreSQL, é possível delegar, com segurança, o gerenciamento de extensões a usuários que não têm o perfil `rds_superuser`. Esse recurso oferece os seguintes benefícios:

- É possível delegar facilmente o gerenciamento de extensões aos usuários de sua escolha.
- Isso não exige o perfil `rds_superuser`.
- Oferece a capacidade de oferecer compatibilidade com um conjunto diferente de extensões para bancos de dados diferentes no mesmo cluster de banco de dados.

Limitação do suporte de extensão delegado do Aurora para PostgreSQL

- Objetos criados durante o processo de criação da extensão podem exigir privilégios adicionais para que a extensão funcione corretamente.

Permissões necessárias para determinadas extensões

Para criar, usar ou atualizar as extensões a seguir, o usuário delegado deve ter os privilégios necessários nestas funções, tabelas e esquemas.

Exter que preci de propr de ou perm s	Função	Tabelas	Schema	Dicionário de pesquisa de texto	Comment
addre tanda er_da s		us_gaz, us_lex, us_lex, l.us_rules			
amch	bt_index_check, bt_index_ parent_check				
dict_i				intdict	
pg_p n		custom_ti me_partitions, part_config, part_config_sub			
pg_s tatem					
Post(	st_tileenvelope	spatial_ref_sys			
postg aster					
postg opolo		topologia, camada	topologia		o usuário delegado deve ser o proprietá

Exter que preci de propr de ou perm s	Função	Tabelas	Schema	Dicionário de pesquisa de texto	Comment
					rio do banco de dados
log_f	create_fo reign_tab le_for_log_file				
rds_t	role_pass word_encr yption_type				
postg iger_ oder		geocode_s ettings_default, geocode_s ettings	tiger		
pg_fr acem	pg_freespace				
pg_vi lity	pg_visibility				

Considerações sobre segurança

Lembre-se de que um usuário com o perfil `rds_extension` poderá gerenciar extensões em todos os bancos de dados nos quais tiver o privilégio de conexão. Se a intenção for fazer com que um usuário delegado gerencie a extensão em um único banco de dados, uma prática recomendada é

revogar todos os privilégios públicos em cada banco de dados e, depois, conceder explicitamente o privilégio de conexão desse banco de dados específico ao usuário delegado.

Existem várias extensões que podem permitir que um usuário acesse informações de vários bancos de dados. Garanta que os usuários que receberem `rds_extension` tenham recursos de vários bancos de dados antes de adicionar essas extensões a `rds.allowed_delegated_extensions`. Por exemplo, `postgres_fdw` e `dblink` oferecem a capacidade de consultar vários bancos de dados na mesma instância ou em instâncias remotas. `log_fdw` lê os arquivos de log do mecanismo postgres, relacionados a todos os bancos de dados na instância, possivelmente contendo consultas lentas ou mensagens de erro de vários bancos de dados. `pg_cron` permite a execução de trabalhos agendados em segundo plano na instância de banco de dados e pode configurar trabalhos para execução em um banco de dados diferente.

Descartar cascata de extensão desabilitado

A capacidade de descartar a extensão com a opção em cascata por um usuário com o perfil `rds_extension` é controlada pelo parâmetro `rds.delegated_extension_allow_drop_cascade`. Por padrão, `rds-delegated_extension_allow_drop_cascade` é definido como `off`. Isso significa que os usuários com o perfil `rds_extension` não têm permissão para descartar uma extensão usando a opção em cascata, conforme mostrado na consulta a seguir.

```
DROP EXTENSION CASCADE;
```

Pois isso descartará automaticamente os objetos que dependem da extensão e, por sua vez, todos os objetos que dependem desses objetos. A tentativa de usar a opção em cascata gerará um erro.

Para conceder essa capacidade, o parâmetro `rds.delegated_extension_allow_drop_cascade` deve ser definido como on.

Alterar o parâmetro dinâmico `rds.delegated_extension_allow_drop_cascade` não requer a reinicialização do banco de dados. É possível fazer isso em um dos seguintes níveis:

- No cluster ou no grupo de parâmetros da instância, por meio do Console de gerenciamento da AWS ou da API.
- Usar o seguinte comando em nível de banco de dados:

```
alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
```

- Usar o seguinte comando em nível de usuário:

```
alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
```

Exemplos de extensões que podem ser adicionadas usando o suporte de extensão delegado

- `rds_tools`

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- `amcheck`

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- `pg_freespacemap`

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- `pg_visibility`

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- `postgres_fdw`

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

Gerenciar objetos grandes com o módulo lo

O módulo lo (extensão) é para usuários de banco de dados e desenvolvedores que trabalham com bancos de dados PostgreSQL por meio de drivers JDBC ou ODBC. Tanto no caso do JDBC quanto no caso do ODBC, é esperado que o banco de dados processe a exclusão de objetos grandes quando as referências a eles mudam. No entanto, o PostgreSQL não funciona dessa maneira. No PostgreSQL, não se espera que um objeto seja excluído quando sua referência é alterada. O resultado é que os objetos permanecem no disco, sem referência. A extensão lo inclui uma função usada para acionar alterações de referência para excluir objetos, se necessário.

> **Tip**
>
> Para determinar se seu banco de dados pode se beneficiar da extensão lo, use o utilitário `vacuumlo` para verificar se há objetos grandes órfãos. Para obter contagens de objetos grandes órfãos sem realizar nenhuma ação, execute o utilitário com a opção `-n` (no-op). Para saber como, consulte vacuumlo utility a seguir.

O módulo lo está disponível para Aurora PostgreSQL 13.7, 12.11, 11.16, 10.21 e versões secundárias superiores.

Para instalar o módulo (extensão), você precisa de privilégios `rds_superuser`. Instalar a extensão lo adiciona o seguinte ao seu banco de dados:

- `lo`: é um tipo de dados de objeto grande (lo) que você pode usar para objetos grandes binários (BLOBs) e outros objetos grandes. O tipo de dados `lo` é um domínio do tipo de dados `oid`. Em outras palavras, é um identificador de objeto com restrições opcionais. Para saber mais, consulte Identificadores de objeto na documentação do PostgreSQL. Em termos simples, você pode usar o tipo de dados `lo` para distinguir suas colunas de banco de dados que contêm referências de objetos grandes de outros identificadores de objeto (OIDs).
- `lo_manage`: é uma função que você pode usar em gatilhos em colunas de tabela que contêm referências a objetos grandes. Sempre que você excluir ou modificar um valor que faça referência a um objeto grande, o gatilho desvincula o objeto (`lo_unlink`) de sua referência. Use o gatilho em uma coluna somente se a coluna for a única referência de banco de dados ao objeto grande.

Para obter mais informações sobre o módulo de objetos grandes, consulte lo na documentação do PostgreSQL.

Instalar a extensão lo

Antes de instalar a extensão lo, verifique se você tem privilégios `rds_superuser`.

Como instalar a extensão lo

1. Use o `psql` para conectar-se à instância de banco de dados primária do cluster de banco de dados Aurora PostgreSQL.

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Insira sua senha quando for solicitado. O cliente `psql` conecta-se e exibe o banco de dados de conexão administrativa padrão, `postgres=>`, como o prompt.

2. Instale a extensão da forma a seguir.

   ```
   postgres=> CREATE EXTENSION lo;
   CREATE EXTENSION
   ```

Agora é possível usar o tipo de dados `lo` para definir colunas em suas tabelas. Por exemplo, você pode criar uma tabela (`images`) que contém dados de imagem raster. Você pode usar o tipo de dados `lo` para uma coluna `raster`, conforme mostrado no exemplo a seguir, que cria uma tabela.

```
postgres=> CREATE TABLE images (image_name text, raster lo);
```

Usar a função de gatilho lo_manage para excluir objetos

É possível usar a função `lo_manage` em um gatilho em uma `lo` ou outras colunas de objetos grandes para limpar (e evitar objetos órfãos) quando a `lo` é atualizada ou excluída.

Como configurar gatilhos em colunas que fazem referência a objetos grandes

- Execute um destes procedimentos:
  - Crie um gatilho BEFORE UPDATE OR DELETE em cada coluna para conter referências exclusivas a objetos grandes, usando o nome da coluna como argumento.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OR DELETE ON images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```
  - Aplique um gatilho somente quando a coluna estiver sendo atualizada.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OF images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```

A função de gatilho `lo_manage` funciona apenas no contexto de inserção ou exclusão de dados de colunas, dependendo de como você define o gatilho. Isso não tem efeito quando você executa uma operação DROP ou TRUNCATEem um banco de dados. Isso significa que é necessário excluir colunas de objeto de qualquer tabela antes de soltá-las, para evitar a criação de objetos órfãos.

Por exemplo, suponha que você queira descartar o banco de dados que contém a tabela `images`. Exclua a coluna da maneira a seguir.

```
postgres=> DELETE FROM images COLUMN raster
```

Supondo que a função `lo_manage` seja definida nessa coluna para lidar com exclusões, agora você pode descartar a tabela com segurança.

Remoção de objetos grandes órfãos usando o **`vacuumlo`**

O utilitário `vacuumlo` identifica e remove objetos grandes órfãos dos bancos de dados. Esse utilitário está disponível desde o PostgreSQL 9.1.24. Se os usuários do banco de dados trabalham

rotineiramente com objetos grandes, recomendamos executar o `vacuumlo` ocasionalmente para limpar objetos grandes órfãos.

Antes de instalar a extensão lo, você pode usar o `vacuumlo` para avaliar se o cluster de banco de dados do Aurora PostgreSQL pode se beneficiar. Para isso, execute `vacuumlo` com a opção `-n` (no-op) para mostrar o que seria removido, conforme mostrado no seguinte:

```
$ vacuumlo -v -n -h your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com -
p 5433 -U postgres docs-lab-spatial-db
Password:*****
Connected to database "docs-lab-spatial-db"
Test run: no large objects will be removed!
Would remove 0 large objects from database "docs-lab-spatial-db".
```

Como mostra a saída, objetos grandes órfãos não são um problema para esse banco de dados específico.

Para obter mais informações sobre esse utilitário, consulte `vacuumlo` na documentação do PostgreSQL.

Como funciona o **`vacuumlo`**

O comando `vacuumlo` remove objetos grandes órfãos (LOs) do seu banco de dados PostgreSQL sem afetar ou entrar em conflito com suas tabelas de usuário.

O comando funciona assim:

1. `vacuumlo` começa criando uma tabela temporária contendo todos os IDs de objeto (OIDs) dos objetos grandes em seu banco de dados.
2. Depois, o `vacuumlo` examina todas as colunas do banco de dados que usam os tipos de dados `oid` ou `lo`. Se o `vacuumlo` encontrar um OID correspondente nessas colunas, ele removerá o OID da tabela temporária. O `vacuumlo` verifica somente colunas com os nomes específicos `oid` ou `lo`, não domínios com base nesses tipos.
3. As entradas restantes na tabela temporária representam LOs órfãos, que o `vacuumlo` remove com segurança.

Aprimorar a performance do **`vacuumlo`**

Você pode melhorar o desempenho do `vacuumlo` aumentando o tamanho do lote usando a opção `-l`. Isso permite que o `vacuumlo` processe mais LOs de uma só vez.

Se o sistema tiver memória suficiente e você puder acomodar a tabela temporária completamente na memória, aumentar a configuração `temp_buffers` no nível do banco de dados poderá melhorar o desempenho. Isso permite que a tabela resida inteiramente na memória, o que pode melhorar o desempenho geral.

A consulta a seguir estima o tamanho da tabela temporária:

```
SELECT
    pg_size_pretty(SUM(pg_column_size(oid))) estimated_lo_temp_table_size
FROM
    pg_largeobject_metadata;
```

Considerações para objetos grandes

A seguir, você encontrará algumas considerações importantes a serem observadas ao trabalhar com objetos grandes:

- `Vacuumlo` é a única solução, pois atualmente não há outro método para remover LOs órfãos.
- Ferramentas como pglogical, replicação lógica nativa e AWS DMS que usam tecnologias de replicação não oferecem suporte à replicação de objetos grandes.
- Ao projetar seu esquema de banco de dados, evite usar objetos grandes sempre que possível e considere usar tipos de dados alternativos, como `bytea`.
- Execute o `vacuumlo` regularmente, pelo menos semanalmente, para evitar problemas com LOs órfãos.
- Use um gatilho com a função `lo_manage` em tabelas que armazenam objetos grandes para ajudar a evitar a criação de LOs órfãos.

Gerenciar dados espaciais com a extensão PostGIS

PostGIS é uma extensão do PostgreSQL para armazenar e gerenciar informações espaciais. Para saber mais sobre a extensão PostGIS, consulte Postgis.net.

Desde a versão 10.5, o PostgreSQL é compatível com a biblioteca libprotobuf 1.3.0 usada pelo PostGIS para trabalhar com dados de blocos vetoriais do Mapbox.

A configuração da extensão PostGIS exige privilégios `rds_superuser`. Recomendamos criar um usuário (perfil) para gerenciar a extensão PostGIS e os dados espaciais. A extensão PostGIS

e seus componentes relacionados adicionam milhares de funções ao PostgreSQL. Considere criar a extensão PostGIS em seu próprio esquema se isso fizer sentido para o seu caso de uso. O exemplo a seguir mostra como instalar a extensão em seu próprio banco de dados, mas isso não é necessário.

Tópicos

- Etapa 1: Criar um usuário (função) para gerenciar a extensão PostGIS
- Etapa 2: Carregar as extensões PostGIS
- Etapa 3: Transferir a propriedade dos esquemas de extensão
- Etapa 4: Transferir a propriedade dos objetos PostGIS
- Etapa 5: Testar as extensões
- Etapa 6: Atualize a extensão PostGIS
- Versões de extensão PostGIS
- Upgrade do PostGIS 2 para o PostGIS 3

Etapa 1: Criar um usuário (função) para gerenciar a extensão PostGIS

Primeiro, conecte-se a uma instância de banco de dados do RDS para PostgreSQL como um usuário que tem privilégios `rds_superuser`. Se você manteve o nome padrão ao configurar a instância, se conectará como `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

Crie um perfil separado (usuário) para administrar a extensão PostGIS.

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

Conceda a esse perfil privilégios `rds_superuser` para permitir que ele instale a extensão.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

Crie um banco de dados a ser usado para seus artefatos PostGIS. Esta etapa é opcional. Como alternativa, você pode criar um esquema em seu banco de dados de usuário para as extensões PostGIS, mas isso também não é necessário.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

Conceda a `gis_admin` todos os privilégios no banco de dados `lab_gis`.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

Saia da sessão e reconecte-se a uma instância de banco de dados do RDS para PostgreSQL como `gis_admin`.

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

Continue a configurar a extensão conforme detalhado nas próximas etapas.

Etapa 2: Carregar as extensões PostGIS

A extensão PostGIS inclui várias extensões relacionadas que funcionam juntas para fornecer funcionalidade geoespacial. Dependendo do seu caso de uso, talvez você não precise de todas as extensões criadas nesta etapa.

Use instruções `CREATE EXTENSION` para carregar as extensões PostGIS.

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
```

```
CREATE EXTENSION
```

É possível verificar os resultados executando a consulta SQL mostrada no exemplo a seguir, que lista as extensões e seus proprietários.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Etapa 3: Transferir a propriedade dos esquemas de extensão

Use as instruções ALTER SCHEMA para transferir a propriedade dos esquemas à função `gis_admin`.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

Se você quiser confirmar a alteração de propriedade, realize a consulta SQL a seguir. Ou é possível usar o metacomando \dn na linha de comando do psql.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;

      List of schemas
```

```
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

Etapa 4: Transferir a propriedade dos objetos PostGIS

> **Note**
>
> Não altere a propriedade das funções do PostGIS. A operação adequada e as futuras atualizações do PostGIS exigem que essas funções retenham a propriedade original. Para ter mais informações sobre permissões do PostGIS, consulte Segurança do PostgreSQL.

Use a seguinte função para transferir a propriedade das tabelas do PostGIS ao perfil `gis_admin`. Execute a seguinte instrução no prompt psql para criar a função.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

Depois, execute a consulta a seguir para executar a função `exec` que, por sua vez, executa as instruções e altera as permissões.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Etapa 5: Testar as extensões

Para evitar a necessidade de especificar o nome do esquema, adicione o esquema `tiger` ao seu caminho de pesquisa usando o seguinte comando.

```
SET search_path=public,tiger;
SET
```

Teste o esquema `tiger` usando a seguinte instrução SELECT.

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Para saber mais sobre essa extensão, consulte Geocodificador Tiger na documentação do PostGIS.

Teste o acesso ao esquema `topology` usando a seguinte instrução `SELECT`. Isso chama a função `createtopology` para registrar um novo objeto de topologia (my_new_topo) com o identificador de referência espacial especificado (26986) e a tolerância padrão (0,5). Para saber mais, consulte CreateTopology na documentação do PostGIS.

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Etapa 6: Atualize a extensão PostGIS

Cada nova versão do PostgreSQL oferece suporte a uma ou mais versões da extensão PostGIS compatíveis com essa versão. A atualização do mecanismo PostgreSQL para uma nova versão não atualiza automaticamente a extensão PostGIS. Antes de atualizar o mecanismo PostgreSQL, faça upgrade do PostGIS para a versão mais recente disponível para a versão atual do PostgreSQL. Para obter detalhes, consulte Versões de extensão PostGIS.

Após a atualização do mecanismo PostgreSQL, faça upgrade da extensão PostGIS novamente, desta vez para a versão compatível com a versão recém-atualizada do mecanismo PostgreSQL. Para obter mais informações sobre como fazer upgrade do mecanismo PostgreSQL, consulte Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal.

Você pode verificar as atualizações de versão da extensão PostGIS disponíveis no seu cluster de banco de dados do Aurora PostgreSQL a qualquer momento. Para fazer isso, execute o comando a seguir. Esta função está disponível com PostGIS 2.5.0 e versões posteriores.

```
SELECT postGIS_extensions_upgrade();
```

Se a sua aplicação não oferecer suporte à versão mais recente do PostGIS, você poderá instalar uma versão mais antiga do PostGIS que esteja disponível na sua versão principal conforme o exposto a seguir.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

Se quiser fazer upgrade para uma versão específica do PostGIS usando uma versão mais antiga, também poderá usar o comando a seguir.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

Dependendo de sua versão atual antes do upgrade, talvez você precise usar essa função novamente. O resultado da primeira execução da função determina a necessidade de uma função de atualização adicional. Por exemplo, isso acontece em caso de upgrade do PostGIS 2 para o PostGIS 3. Para obter mais informações, consulte Upgrade do PostGIS 2 para o PostGIS 3.

Se você atualizou essa extensão para se preparar para uma atualização da versão principal do mecanismo PostgreSQL, poderá continuar com outras tarefas preliminares. Para obter mais informações, consulte Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal.

Versões de extensão PostGIS

Recomendamos que você instale as versões de todas as extensões, como PostGIS, conforme listado em "Extension versions for Aurora PostgreSQL-Compatible Edition" (Versões de extensões para a edição compatível com Aurora PostgreSQL) nas Notas de lançamento do Aurora PostgreSQL. Você pode conferir quais versões estão disponíveis na sua versão usando o comando a seguir.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

Informações sobre versões estão disponíveis nas seções a seguir das Notas de lançamento do Amazon RDS para PostgreSQL:

- Versões de extensões para o Aurora PostgreSQL 14
- Versões de extensões para a edição do Aurora compatível com PostgreSQL 13
- Versões de extensões para a edição do Aurora compatível com PostgreSQL 12
- Versões de extensões para a edição do Aurora compatível com PostgreSQL 11
- Versões de extensões para a edição do Aurora compatível com PostgreSQL 10
- Versões de extensões para a edição do Aurora compatível com PostgreSQL 9.6

Upgrade do PostGIS 2 para o PostGIS 3

A partir da versão 3.0, a funcionalidade de rasterização do PostGIS é uma extensão separada, `postgis_raster`. Essa extensão tem seu próprio caminho de instalação e upgrade. Isso remove dezenas de funções, tipos de dados e outros artefatos necessários para o processamento de imagens rasterizadas da extensão `postgis` principal. Isso significa que, se o seu caso de uso não exigir processamento de rasterização, você não precisará instalar a extensão `postgis_raster`.

No exemplo de upgrade a seguir, o primeiro comando de upgrade extrai a funcionalidade de rasterização na extensão `postgis_raster`. Um segundo comando de upgrade é necessário para atualizar `postgis_raster` para a nova versão.

Como fazer upgrade do PostGIS 2 para o PostGIS 3

1. Identifique a versão padrão do PostGIS que está disponível para a versão do PostgreSQL em seucluster de banco de dados do Aurora PostgreSQL. Para fazer isso, execute a consulta a seguir.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. Identifique as versões do PostGIS instaladas em cada banco de dados na instância de leitor do seu cluster de banco de dados do Aurora PostgreSQL. Em outras palavras, consulte cada banco de dados do usuário da seguinte forma.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                            Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

Essa divergência entre a versão padrão (PostGIS 3.1.4) e a versão instalada (PostGIS 2.3.7) significa que você precisa atualizar a extensão PostGIS.

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. Execute a consulta a seguir para verificar se a funcionalidade de rasterização já está em seu próprio pacote.

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
```

```
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

O resultado mostra que ainda há uma diferença entre as versões. As funções do PostGIS são da versão 3 (postgis-3), enquanto as funções de rasterização (rtpostgis) são da versão 2 (rtpostgis-2.3). Para concluir a atualização, execute o comando de upgrade novamente, da seguinte forma.

```
postgres=> SELECT postgis_extensions_upgrade();
```

Você pode ignorar as mensagens de aviso. Execute a consulta a seguir novamente para verificar se a atualização foi concluída. A atualização é concluída quando o PostGIS e todas as extensões relacionadas deixam de estar sinalizadas como necessitando de atualização.

```
SELECT postgis_full_version();
```

4. Use a consulta a seguir para ver o processo de atualização concluído e as extensões empacotadas separadamente, e verifique se as versões correspondem.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                            Description
----------------+---------+--------
+------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
```

```
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

A saída mostra que a extensão PostGIS 2 foi atualizada para PostGIS 3, e tanto `postgis` quanto a extensão `postgis_raster` agora separada estão na versão 3.1.5.

Depois que essa atualização for concluída, se você não planejar usar a funcionalidade de rasterização, poderá descartar a extensão da seguinte forma.

```
DROP EXTENSION postgis_raster;
```

Gerenciar partições do PostgreSQL com a extensão pg_partman

O particionamento de tabelas PostgreSQL fornece um framework para processamento de alta performance de entrada e relatórios de dados. Use o particionamento para bancos de dados que exigem entrada muito rápida de grandes quantidades de dados. O particionamento também fornece consultas mais rápidas de tabelas grandes. O particionamento ajuda a manter os dados sem afetar a instância do banco de dados, pois requer menos recursos de E/S.

Ao usar o particionamento, você pode dividir dados em blocos de tamanho personalizado para processamento. Por exemplo, você pode particionar dados de séries temporais para intervalos como por hora, diário, semanal, mensal, trimestral, anual, personalizado ou qualquer combinação destes. Para um exemplo de dados de séries temporais, se você particionar a tabela por hora, cada partição conterá uma hora de dados. Se você particionar a tabela de séries temporais por dia, as partições manterão dados de um dia e assim por diante. A chave de partição controla o tamanho de uma partição.

Quando você usa um comando SQL `INSERT` ou `UPDATE` em uma tabela particionada, o mecanismo de banco de dados roteia os dados para a partição apropriada. As partições de tabela PostgreSQL que armazenam os dados são tabelas filhas da tabela principal.

Durante as leituras de consulta de banco de dados, o otimizador PostgreSQL examina a cláusula `WHERE` da consulta e, se possível, direciona a verificação do banco de dados apenas para as partições relevantes.

A partir da versão 10, o PostgreSQL usa o particionamento declarativo para implementar o particionamento de tabela. Isso também é conhecido como particionamento nativo do PostgreSQL. Antes do PostgreSQL versão 10, você usou gatilhos para implementar partições.

O particionamento de tabelas PostgreSQL fornece os seguintes recursos:

- Criação de novas partições a qualquer momento.
- Intervalos variáveis de partição.
- Partições destacáveis e reanexáveis usando instruções DDL (Data Definition Language, linguagem de definição de dados).

  Por exemplo, partições destacáveis são úteis para remover dados históricos da partição principal, mas manter dados históricos para análise.
- Novas partições herdam as propriedades da tabela do banco de dados pai, incluindo:
  - Índices
  - Chaves primárias, que devem incluir a coluna de chave de partição
  - Chaves externas
  - Restrições de verificação
  - Referências
- Criação de índices para a tabela completa ou cada partição específica.

Você não pode alterar o esquema de uma partição individual. No entanto, você pode fazer uma alteração na tabela pai (adicionando uma nova coluna, por exemplo) que se propaga para as partições.

Tópicos

- Visão geral da extensão pg_partman do PostgreSQL
- Ativar a extensão pg_partman
- Configurar partições usando a função create_parent
- Configurar a manutenção da partição usando a função run_maintenance_proc

Visão geral da extensão pg_partman do PostgreSQL

Você pode usar a extensão `pg_partman` do PostgreSQL para automatizar a criação e a manutenção de partições de tabelas. Para obter mais informações gerais, consulte PG Partition Manager na documentação `pg_partman`.

> **Note**
>
> A extensão `pg_partman` é compatível com o Aurora PostgreSQL versões 12.6 e posteriores.

Em vez de ter que criar manualmente cada partição, você ajusta o `pg_partman` com as seguintes configurações:

- Tabela a ser particionada
- Tipo de partição
- Chave de partição
- Granularidade de partição
- Opções de pré-criação e gerenciamento de partições

Depois de criar uma tabela particionada do PostgreSQL, registre-a com `pg_partman` chamando a função `create_parent`. Fazer isso cria as partições necessárias com base nos parâmetros que você passa para a função.

A extensão `pg_partman` também fornece a função `run_maintenance_proc`, que você pode chamar de maneira programada para gerenciar as partições automaticamente. Para garantir que as partições adequadas sejam criadas conforme necessário, agende essa função para ser executada periodicamente (por hora, por exemplo). Você também pode garantir que as partições sejam descartadas automaticamente.

Ativar a extensão pg_partman

Se você tiver vários bancos de dados dentro da mesma instância de banco de dados PostgreSQL para a qual deseja gerenciar partições, habilite a extensão `pg_partman` separadamente para cada banco de dados. Para habilitar a extensão `pg_partman` para um banco de dados específico, crie o esquema de manutenção de partição e crie a extensão `pg_partman` da maneira a seguir.

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> Para criar a extensão `pg_partman`, certifique-se de que você tenha privilégios `rds_superuser`.

Se você receber um erro como o seguinte, conceda os privilégios de `rds_superuser` à conta ou use sua conta de superusuário.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

Para conceder privilégios de `rds_superuser`, conecte-se à sua conta de superusuário e execute o seguinte comando.

```
GRANT rds_superuser TO user-or-role;
```

Para os exemplos que mostram usando a extensão pg_partman, usamos a seguinte tabela de banco de dados de amostra e partição. Esse banco de dados usa uma tabela particionada com base em um carimbo de data/hora. Um esquema `data_mart` contém uma tabela chamada `events` com uma coluna chamada `created_at`. As seguintes configurações estão incluídas na tabela `events`:

- Chaves primárias `event_id` e `created_at`, que devem ter a coluna usada para orientar a partição.
- Uma restrição de verificação `ck_valid_operation` para impor valores para uma coluna `operation` da tabela.
- Duas chaves estrangeiras, onde uma (`fk_orga_membership`) aponta para a tabela externa `organization` e a outra (`fk_parent_event_id`) é uma chave estrangeira autorreferenciada.
- Dois índices, onde um (`idx_org_id`) é para a chave estrangeira e o outro (`idx_event_type`) é para o tipo de evento.

As declarações DDL a seguir criam esses objetos, que serão incluídos automaticamente em cada partição.

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
```

```
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

Configurar partições usando a função create_parent

Depois de habilitar a extensão `pg_partman`, use a função `create_parent` para configurar partições dentro do esquema de manutenção de partição. Aqui é usado o exemplo de tabela `events` criado em Ativar a extensão pg_partman. Chame a função `create_parent` da seguinte forma:

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

Os parâmetros são os seguintes:

- `p_parent_table` – a tabela particionada pai. Essa tabela já deve existir e estar totalmente qualificada, incluindo o esquema.

- `p_control` – a coluna na qual o particionamento deve ser baseado. O tipo de dados deve ser um inteiro ou baseado em tempo.
- `p_type`: o tipo é `'range'` ou `'list'`.
- `p_interval` – o intervalo ou a faixa de inteiros para cada partição. Os valores de exemplo incluem `1 day`, `1 hour` e assim por diante.
- `p_premake` – o número de partições para criar antecipadamente a fim de dar suporte a novas inserções.

Para obter uma descrição completa da função `create_parent`, consulte Funções de criação na documentação do `pg_partman`..

Configurar a manutenção da partição usando a função run_maintenance_proc

Você pode executar operações de manutenção de partição para criar automaticamente novas partições, desanexar partições ou remover partições antigas. A manutenção da partição depende da função `run_maintenance_proc` da extesão `pg_partman` e da extensão `pg_cron`, que inicia um programador interno. O agendador `pg_cron` executa automaticamente instruções SQL, funções e procedimentos definidos em seus bancos de dados.

A seguir, será usado o exemplo de tabela `events` criado em Ativar a extensão pg_partman para definir que as operações de manutenção de partição serão executadas automaticamente. Como pré-requisito, adicione `pg_cron` ao parâmetro `shared_preload_libraries` no grupo de parâmetros da instância de banco de dados.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

A seguir, você pode encontrar uma explicação detalhada do exemplo anterior:

1. Modifique o grupo de parâmetros associado à sua instância de banco de dados e adicione `pg_cron` ao valor do parâmetro `shared_preload_libraries`. Essa alteração exige a reinicialização da instância de banco de dados para que tenha efeito. Para obter mais

informações, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

2. Execute o comando `CREATE EXTENSION pg_cron;` usando uma conta que tenha as permissões de `rds_superuser`. Isso habilita a extensão `pg_cron`. Para obter mais informações, consulte Agendar manutenção com a extensão pg_cron do PostgreSQL.
3. Execute o comando `UPDATE partman.part_config` para ajustar as configurações `pg_partman` para a tabela `data_mart.events`.
4. Execute o comando `SET . . .` para configurar a tabela `data_mart.events`, com estas cláusulas:
   a. `infinite_time_partitions = true,` – configura a tabela para poder criar novas partições automaticamente, sem qualquer limite.
   b. `retention = '3 months',` – configura a tabela para ter uma retenção máxima de três meses.
   c. `retention_keep_table=true` – Configura a tabela para que, quando o período de retenção for devido, a tabela não seja excluída automaticamente. Em vez disso, as partições que são mais antigas do que o período de retenção são apenas separadas da tabela pai.
5. Execute o comando `SELECT cron.schedule . . .` para fazer uma chamada da função `pg_cron`. Esta chamada define com que frequência o programador executa o procedimento de manutenção `pg_partman`, `partman.run_maintenance_proc`. Para este exemplo, o procedimento é executado a cada hora.

Para obter uma descrição completa da função `run_maintenance_proc`, consulte Funções de manutenção na documentação do `pg_partman`.

Agendar manutenção com a extensão pg_cron do PostgreSQL

Você pode utilizar a extensão `pg_cron` do PostgreSQL para programar comandos de manutenção dentro de um banco de dados do PostgreSQL. Para obter mais informações sobre a extensão, consulte O que é pg_cron? na documentação do pg_cron.

A extensão `pg_cron` é compatível com o mecanismo do Aurora PostgreSQL versões 12.6 e posteriores

Para saber mais sobre como usar `pg_cron`, consulte Programar trabalhos com pg_cron em bancos de dados do RDS para PostgreSQL ou compatíveis com o Aurora PostgreSQL.

> **Note**
>
> A versão da extensão `pg_cron` é exibida como uma versão de dois dígitos, por exemplo, 1.6, na visualização pg_available_extensions. Embora você possa ver versões de três dígitos, por exemplo, 1.6.4 ou 1.6.5, listadas em alguns contextos, você deve especificar a versão de dois dígitos ao realizar uma atualização de extensão.

Tópicos

- Configurar a extensão pg_cron
- Conceder permissões de banco de dados para usar pg_cron
- Agendar trabalhos de pg_cron
- Referência para a extensão pg_cron

Configurar a extensão pg_cron

Configure a extensão `pg_cron` da seguinte forma:

1. Modifique o grupo de parâmetros personalizado associado à sua instância de banco de dados do PostgreSQL adicionando `pg_cron` ao valor do parâmetro `shared_preload_libraries`.

   Reinicie a instância de banco de dados do PostgreSQL para que as alterações no grupo de parâmetros entrem em vigor. Para saber mais sobre como trabalhar com grupos de parâmetros, consulte Amazon Aurora PostgreSQL parameters.
2. Após a reinicialização da instância de banco de dados do PostgreSQL, execute o comando a seguir usando uma conta que tenha permissões `rds_superuser`. Por exemplo, se você usou as configurações padrão ao criar o cluster de banco de dados do Aurora PostgreSQL, conecte-se como o usuário `postgres` e crie a extensão.

   ```
   CREATE EXTENSION pg_cron;
   ```

   O agendador do `pg_cron` é definido no banco de dados PostgreSQL padrão chamado `postgres`. Os objetos `pg_cron` são criados neste banco de dados `postgres` e todas as ações de agendamento são executadas neste banco de dados.
3. Você pode usar as configurações padrão ou programar trabalhos para serem executados em outros bancos de dados dentro de sua instância de banco de dados PostgreSQL. Para programar

trabalhos a serem executados em outros bancos de dados em sua instância de banco de dados PostgreSQL, consulte o exemplo em Agendar um trabalho cron para um banco de dados diferente do banco de dados padrão.

Conceder permissões de banco de dados para usar pg_cron

A instalação da extensão `pg_cron` requer privilégios de `rds_superuser`. No entanto, as permissões para usar `pg_cron` podem ser concedidas (por um membro do grupo/perfil de `rds_superuser`) para outros usuários do banco de dados, para que eles possam programar seus próprios trabalhos. Recomendamos que você conceda permissões para o esquema `cron` somente conforme necessário se ele melhorar as operações do ambiente de produção.

Para conceder permissão a um usuário do banco de dados no esquema `cron`, execute o seguinte comando:

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

Isso concede a *db-user* permissão para acessar o esquema `cron` para programar trabalhos cron para os objetos que o usuário tem permissão para acessar. Se o usuário do banco de dados não tiver permissões, o trabalho falhará após a publicação da mensagem de erro no arquivo `postgresql.log`, conforme mostrado a seguir:

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

Em outras palavras, certifique-se de que os usuários do banco de dados que tenham permissões no esquema `cron` também tenham permissões nos objetos (tabelas, esquemas e assim por diante) que planejam programar.

Os detalhes do trabalho cron e seu sucesso ou falha também são capturados na tabela `cron.job_run_details`. Para obter mais informações, consulte Tabelas para agendar trabalhos e capturar status.

Agendar trabalhos de pg_cron

As seções a seguir mostram como você pode agendar várias tarefas de gerenciamento usando trabalhos `pg_cron`.

> **Note**
>
> Ao criar trabalhos pg_cron, verifique se a configuração max_worker_processes a configuração é maior do que o número de cron.max_running_jobs. Um trabalho pg_cron falhará se ficar sem processos de operador em segundo plano. O número padrão de trabalhos pg_cron é 5. Para obter mais informações, consulte Parâmetros para gerenciar a extensão pg_cron.

Tópicos

- Vacuum de tabelas
- Limpar a tabela de histórico de pg_cron
- Registrar em log erros somente no arquivo postgresql.log
- Agendar um trabalho cron para um banco de dados diferente do banco de dados padrão

Vacuum de tabelas

O autovacuum lida com manutenção de vacuum para a maioria dos casos. No entanto, você pode agendar o vacuum de uma tabela específica quando quiser.

Veja a seguir um exemplo de uso da função cron.schedule para configurar um trabalho a ser usado VACUUM FREEZE em uma tabela específica todos os dias às 22:00 (GMT).

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

Após o exemplo anterior ser executado, você pode verificar o histórico na tabela cron.job_run_details da seguinte forma.

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                        |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+--------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
```

```
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

A seguir está uma consulta à tabela `cron.job_run_details` para ver os trabalhos que falharam.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                        | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+--------------------------------+--------
+-----------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

Para obter mais informações, consulte Tabelas para agendar trabalhos e capturar status.

Limpar a tabela de histórico de pg_cron

A tabela `cron.job_run_details` contém um histórico de trabalhos cron que podem se tornar muito grandes ao longo do tempo. Recomendamos que você agende um trabalho que limpe essa tabela. Por exemplo, manter uma semana de registros pode ser suficiente para fins de solução de problemas.

O exemplo a seguir usa a função cron.schedule para agendar um trabalho que é executado todos os dias à meia-noite para limpar a tabela `cron.job_run_details`. O trabalho mantém apenas os últimos sete dias. Use sua `rds_superuser` para agendar o trabalho da seguinte forma.

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

Para obter mais informações, consulte Tabelas para agendar trabalhos e capturar status.

Registrar em log erros somente no arquivo postgresql.log

Para impedir a gravação na tabela `cron.job_run_details`, modifique o grupo de parâmetros associado à instância de banco de dados do PostgreSQL e defina o parâmetro `cron.log_run`

como desativado. A extensão pg_cron não gravará mais na tabela e vai capturar erros somente no arquivo postgresql.log. Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Use o comando a seguir para verificar o valor do parâmetro cron.log_run.

```
postgres=> SHOW cron.log_run;
```

Para obter mais informações, consulte Parâmetros para gerenciar a extensão pg_cron.

Agendar um trabalho cron para um banco de dados diferente do banco de dados padrão

Os metadados para pg_cron são todos mantidos no banco de dados padrão PostgreSQL chamado postgres. Como os operadores em segundo plano são usados para executar os trabalhos cron de manutenção, você pode agendar um trabalho em qualquer um dos seus bancos de dados dentro da instância de banco de dados do PostgreSQL.

> **Note**
>
> Somente usuários com o perfil rds_superuser ou privilégios rds_superuser podem listar todos os trabalhos cron no banco de dados. Outros usuários podem visualizar somente seus próprios trabalhos na tabela cron.job.

1. No banco de dados cron, agende o trabalho como você normalmente faria usando a cron.schedule.

   ```
   postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
    freeze test_table');
   ```

2. Como um usuário com a função rds_superuser, atualize a coluna do banco de dados para o trabalho que você acabou de criar para que ele seja executado em outro banco de dados dentro de sua instância de banco de dados do PostgreSQL.

   ```
   postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
   ```

3. Verifique consultando a tabela cron.job.

   ```
   postgres=> SELECT * FROM cron.job;
   ```

```
jobid | schedule    | command                          | nodename  | nodeport |
 database | username  | active | jobname
------+-------------+----------------------------------+-----------+----------
+----------+----------+--------+-----------------------
106   | 29 03 * * * | vacuum freeze test_table       | localhost | 8192     |
 database1| adminuser | t      | database1 manual vacuum
  1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
 postgres | adminuser | t      | manual vacuum
(2 rows)
```

> **Note**
>
> Em algumas situações, você pode adicionar um cron job que você pretende executar em um banco de dados diferente. Nesses casos, o job pode tentar executar no banco de dados padrão (`postgres`) antes de atualizar a coluna correta do banco de dados. Se o nome de usuário tiver permissões, o trabalho será executado com êxito no banco de dados padrão.

Referência para a extensão pg_cron

Você pode usar os seguintes parâmetros, funções e tabelas com a extensão `pg_cron`. Para obter mais informações, consulte O que é pg_cron? na documentação do pg_cron.

Tópicos

- Parâmetros para gerenciar a extensão pg_cron
- Referência da função: cron.schedule
- Referência da função: cron.schedule
- Tabelas para agendar trabalhos e capturar status

Parâmetros para gerenciar a extensão pg_cron

Veja a seguir uma lista de parâmetros que controlam o comportamento da extensão `pg_cron`.

Parâmetro	Descrição
cron.database_name	O banco de dados em que os metadados de `pg_cron` são mantidos.

Parâmetro	Descrição
`cron.host`	O nome do host para se conectar ao PostgresSQL. Não é possível modificar esse valor.
`cron.log_run`	Registre todos os trabalhos executados na tabela `job_run_details` . Os valores são on ou `off`. Para obter mais informações, consulte Tabelas para agendar trabalhos e capturar status.
`cron.log_statement`	Registre todas as instruções cron antes de executá-las. Os valores são on ou `off`.
`cron.max_running_jobs`	O número máximo de trabalhos que podem ser executados simultaneamente.
`cron.use_background_workers`	Use trabalhadores em segundo plano em vez de sessões de cliente. Não é possível modificar esse valor.

Use o seguinte comando SQL para exibir esses parâmetros e seus valores.

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

Referência da função: cron.schedule

Essa função agenda um trabalho cron. Inicialmente, o trabalho é agendado no banco de dados `postgres` padrão. A função retorna um valor `bigint` que representa o identificador de trabalho. Para agendar trabalhos a serem executados em outros bancos de dados em sua instância de banco de dados PostgreSQL, consulte o exemplo em Agendar um trabalho cron para um banco de dados diferente do banco de dados padrão.

A função tem dois formatos de sintaxe.

Sintaxe

```
cron.schedule (job_name,
```

```
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

Parâmetros

Parâmetro	Descrição
`job_name`	O nome do trabalho cron.
`schedule`	Texto indicando a programação do trabalho cron. O formato é o formato cron padrão.
`command`	Texto do comando a ser executado.

Exemplos

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

Referência da função: cron.schedule

Esta função exclui um trabalho cron. Você pode especificar `job_name` ou `job_id`. Uma política garante que você seja o proprietário para remover a programação do trabalho. A função retorna um booleano indicando êxito ou falha.

A função tem os seguintes formatos de sintaxe.

Sintaxe

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Parâmetros

Parâmetro	Descrição
`job_id`	Um identificador de trabalho que foi retornado da função `cron.schedule` quando o trabalho cron foi programado.
`job_name`	O nome de um trabalho cron que foi agendado com a função `cron.schedule` .

Exemplos

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

Tabelas para agendar trabalhos e capturar status

As tabelas a seguir são usadas para agendar os trabalhos cron e registrar como os trabalhos foram concluídos.

Tabela	Descrição
`cron.job`	Contém os metadados sobre cada trabalho agendado. A maioria das interações com esta tabela deve ser feita por meio das funções `cron.schedule` e `cron.unschedule`. **Important** Não recomendamos conceder privilégios de atualização ou inserção diretamente a essa tabela. Isso permitiria que o usuário atualizasse a coluna `username` para ser executada como `rds-superuser`.
`cron.job_run_details`	Contém informações históricas sobre trabalhos agendados passados que foram executados. Isso é útil para investigar o status, as mensagens de retorno e as horas de início e término do trabalho executado. **Note** Para evitar que esta tabela cresça indefinidamente, purgue-a regularmente. Para ver um exemplo, consulte Limpar a tabela de histórico de pg_cron.

Usar pgAudit para registrar a atividade do banco de dados

Instituições financeiras, agências governamentais e muitos setores precisam manter registros de auditoria para atender aos requisitos regulatórios. Ao usar a extensão do PostgreSQL Audit (pgAudit) com seu cluster de banco de dados do Aurora PostgreSQL, você pode capturar os registros detalhados que normalmente são necessários aos auditores ou para atender aos requisitos regulatórios. Por exemplo, você pode configurar a extensão pgAudit para monitorar alterações feitas em tabelas e bancos de dados específicos, registrar o usuário que fez a alteração e muitos outros detalhes.

A extensão pgAudit se baseia na funcionalidade da infraestrutura de registro em log nativa do PostgreSQL, estendendo as mensagens de log com mais detalhes. Em outras palavras, é usada a

mesma abordagem para visualizar o log de auditoria e quaisquer mensagens de log. Para obter mais informações sobre o registro em log do PostgreSQL, consulte Arquivos de log do banco de dados Aurora PostgreSQL.

A extensão pgAudit retira dados confidenciais, como senhas de texto não criptografado, dos logs. Se seu cluster de banco de dados do Aurora PostgreSQL estiver configurado para registrar declarações de linguagem de manipulação de dados (DML) conforme detalhado em Ativar o registro em log de consultas para o cluster de banco de dados do Aurora PostgreSQL, você poderá evitar o problema de senha de texto não criptografado usando a extensão do PostgreSQL Audit.

Você pode configurar a auditoria em suas instâncias de banco de dados com um alto grau de especificidade. É possível auditar todos os bancos de dados e todos os usuários. Ou você pode optar por auditar somente determinados bancos de dados, usuários e outros objetos. Também é possível excluir explicitamente da auditoria determinados usuários e bancos de dados. Para obter mais informações, consulte Excluir usuários ou bancos de dados do registro em log de auditoria.

Dada a quantidade de detalhes que podem ser capturados, recomendamos que, se você usar pgAudit, monitore seu consumo de armazenamento.

A extensão pgAudit é compatível com todas as versões disponíveis do Aurora PostgreSQL. Para obter uma lista de versões de pgAudit compatíveis com a versão do Aurora PostgreSQL, consulte Extension versions for Amazon Aurora PostgreSQL (Versões de extensão para o Amazon Aurora PostgreSQL) em Release Notes for Aurora PostgreSQL (Notas de versão do Aurora PostgreSQL).

Tópicos

- Configurar a extensão pgAudit
- Auditar objetos de banco de dados
- Excluir usuários ou bancos de dados do registro em log de auditoria
- Referência para a extensão pgAudit

Configurar a extensão pgAudit

Para configurar a extensão pgAudit em , seu cluster de banco de dados do Aurora PostgreSQL, primeiro adicione pgAudit às bibliotecas compartilhadas no grupo de parâmetros de cluster de banco de dados personalizado para seu cluster de banco de dados do Aurora PostgreSQL. Para obter informações sobre como criar um grupo de parâmetros de cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora. Depois, instale a extensão pgAudit. Por fim,

especifique os bancos de dados e os objetos que deseja auditar. Os procedimentos nesta seção mostram o procedimento. É possível usar o Console de gerenciamento da AWS ou a AWS CLI.

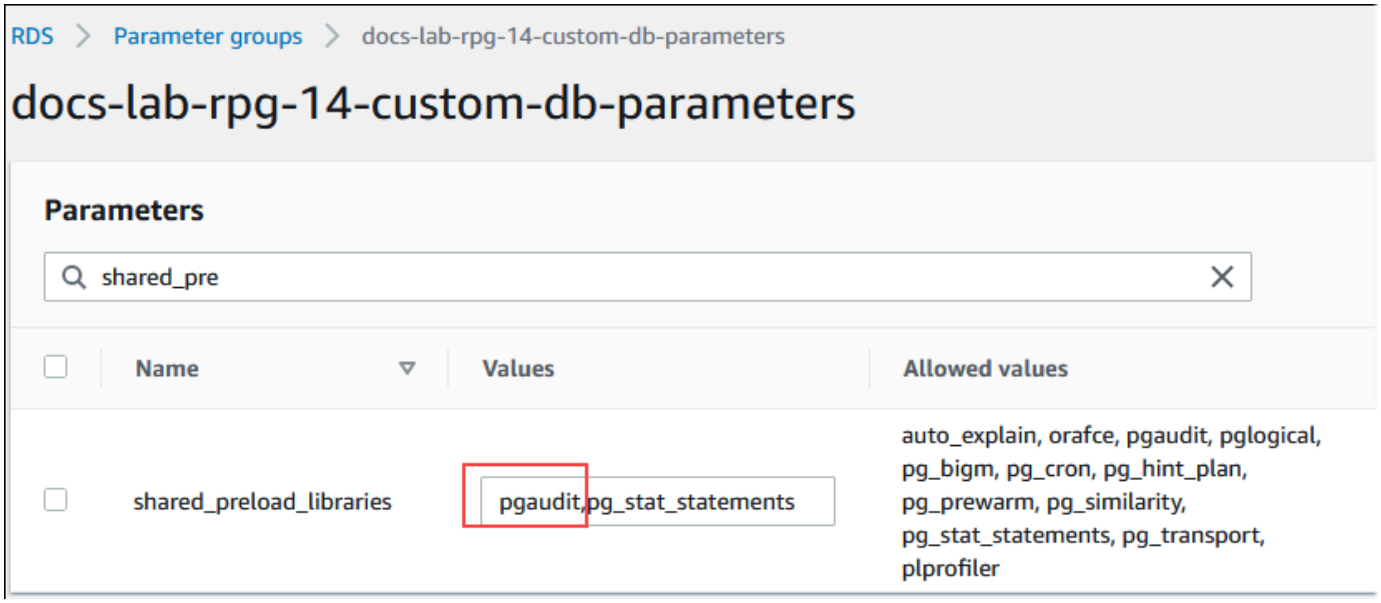
Você deve ter permissões como a função `rds_superuser` para realizar todas essas tarefas.

As etapas a seguir pressupõem que seu cluster de banco de dados do Aurora PostgreSQL esteja associado a um grupo de parâmetros de cluster de banco de dados.

Console

Como configurar a extensão pgAudit

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione sua instância do gravador do cluster de banco de dados do Aurora PostgreSQL.
3. Abra a guia Configuration (Configuração) para sua instância do gravador de cluster de banco de dados do Aurora PostgreSQL. Entre os detalhes da instância, encontre o link Parameter group (Grupo de parâmetros).
4. Clique no link para abrir os parâmetros personalizados associados ao seu cluster de banco de dados do Aurora PostgreSQL.
5. No campo Parameters (Parâmetros), digite `shared_pre` para encontrar o parâmetro `shared_preload_libraries`.
6. Selecione Edit parameters (Editar parâmetros) para acessar os valores das propriedades.
7. Adicione `pgaudit` à lista no campo Values (Valores). Use uma vírgula para separar itens na lista de valores.



8. Reinicie a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para que a alteração no parâmetro `shared_preload_libraries` tenha efeito.
9. Quando a instância estiver disponível, verifique se a pgAudit foi inicializada. Use `psql` para se conectar à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL e depois execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. Com a pgAudit inicializada, agora você pode criar a extensão. Você precisa criar a extensão depois de inicializar a biblioteca porque a extensão `pgaudit` instala acionadores de eventos para auditar declarações de linguagem de definição de dados (DDL).

```
CREATE EXTENSION pgaudit;
```

11. Feche a sessão `psql`.

```
labdb=> \q
```

12. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

13. Encontre o parâmetro `pgaudit.log` na lista e defina como o valor apropriado para o caso de uso. Por exemplo, definir o parâmetro `pgaudit.log` como `write` conforme mostrado na imagem a seguir captura inserções, atualizações, exclusões e alguns outros tipos de alterações no log.

Você também pode selecionar um dos valores a seguir para o parâmetro `pgaudit.log`.

- none: esse é o valor padrão. Nenhuma alteração no banco de dados é registrada.
- all: registra tudo (read, write, function, role, ddl, misc).
- ddl: registra todas as instruções de linguagem de definição de dados (DDL) não incluídas na classe `ROLE`.
- function: registra chamadas de função e blocos de `DO`.
- misc: registra comandos diversos, como `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.
- read: registra `SELECT` e `COPY` quando a fonte é uma relação (como uma tabela) ou uma consulta.
- role: registra declarações relacionadas a funções e privilégios, como `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando o destino é uma relação (tabela).

14. Escolha Salvar alterações.
15. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
16. Selecione a instância do gravador do cluster de banco de dados do Aurora PostgreSQL na lista Bancos de dados.

AWS CLI

Como configurar a pgAudit

Para configurar a pgAudit usando a AWS CLI, chame a operação modify-db-parameter-group para modificar os parâmetros do log de auditoria em seu grupo de parâmetros personalizado, conforme mostrado no procedimento a seguir.

1. Use o comando AWS CLI a seguir para adicionar `pgaudit` ao parâmetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Use o comando AWS CLI a seguir para reinicializar a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para que a biblioteca da pgaudit seja inicializada.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando a instância estiver disponível, verifique se a `pgaudit` foi inicializada. Use `psql` para se conectar à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL e, depois, execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

   Com a pgAudit inicializada, agora você pode criar a extensão.

```
CREATE EXTENSION pgaudit;
```

4. Feche a sessão `psql` para que você possa usar a AWS CLI.

```
labdb=> \q
```

5. Use o comando AWS CLI a seguir para especificar as classes de declaração que devem ser registradas pelo registro em log de auditoria da sessão. O exemplo define o parâmetro `pgaudit.log` como `write`, que captura inserções, atualizações e exclusões no log.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

Você também pode selecionar um dos valores a seguir para o parâmetro `pgaudit.log`.

- none: esse é o valor padrão. Nenhuma alteração no banco de dados é registrada.
- all: registra tudo (read, write, function, role, ddl, misc).
- ddl: registra todas as instruções de linguagem de definição de dados (DDL) não incluídas na classe `ROLE`.
- function: registra chamadas de função e blocos de `DO`.
- misc: registra comandos diversos, como `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.
- read: registra `SELECT` e `COPY` quando a fonte é uma relação (como uma tabela) ou uma consulta.
- role: registra declarações relacionadas a funções e privilégios, como `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando o destino é uma relação (tabela).

Reinicie a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL usando o comando AWS CLI a seguir.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Auditar objetos de banco de dados

Com a pgAudit configurada em seu cluster de banco de dados do Aurora PostgreSQL e configurada para seus requisitos, informações mais detalhadas são capturadas no log do PostgreSQL. Por exemplo, enquanto a configuração de registro em padrão do PostgreSQL identifica a data e a hora em que uma alteração foi feita em uma tabela do banco de dados, com a extensão pgAudit, a entrada do log pode incluir o esquema, o usuário que fez a alteração e outros detalhes, dependendo de como os parâmetros da extensão estão configurados. Você pode configurar a auditoria para monitorar as alterações das maneiras a seguir.

- Para cada sessão, por usuário. Para o nível da sessão, você pode capturar o texto do comando totalmente qualificado.
- Para cada objeto, por usuário e por banco de dados.

O recurso de auditoria de objetos é ativado quando você cria a função `rds_pgaudit` no sistema e depois a adiciona ao parâmetro `pgaudit.role` no grupo de parâmetros personalizado. Por padrão, o parâmetro `pgaudit.role` não está definido e o único valor permitido é `rds_pgaudit`. As etapas a seguir pressupõem que a `pgaudit` tenha sido inicializada e que você tenha criado a extensão `pgaudit` seguindo o procedimento em Configurar a extensão pgAudit.

```
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: statement: SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: AUDIT: SESSION,2,1,READ,SELECT,TABLE,public.support,"SELECT
feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;",<none>
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: QUERY STATISTICS
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:DETAIL: ! system usage stats:
! 0.009494 s user, 0.007442 s system, 0.141985 s elapsed
! [0.022327 s user, 0.007442 s system total]
```

Conforme mostrado neste exemplo, a linha “LOG: AUDIT: SESSION” fornece informações sobre a tabela e o respectivo esquema, entre outros detalhes.

Como configurar a auditoria de objetos

1. Use `psql` para se conectar à instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

   ```
   psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
   username=postgrespostgres --password --dbname=labdb
   ```

2. Crie uma função de banco de dados chamada `rds_pgaudit` usando o comando a seguir.

```
labdb=> CREATE ROLE rds_pgaudit;
CREATE ROLE
labdb=>
```

3. Feche a sessão `psql`.

```
labdb=> \q
```

Nas próximas etapas, use a AWS CLI para modificar os parâmetros de log de auditoria no grupo de parâmetros personalizado.

4. Use o comando AWS CLI a seguir para definir o parâmetro `pgaudit.role` como `rds_pgaudit`. Por padrão, esse parâmetro está vazio, e `rds_pgaudit` é o único valor permitido.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
   --region aws-region
```

5. Use o comando AWS CLI a seguir para reinicializar a instância do gravador do cluster de banco de dados do Aurora PostgreSQL para que as alterações nos parâmetros tenham efeito.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

6. Execute o comando a seguir para confirmar que `pgaudit.role` está definido como `rds_pgaudit`.

```
SHOW pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

Para testar o registro em log da extensão pgAudit, execute vários comandos de exemplo semelhantes ao que você deseja auditar. Por exemplo, você pode executar os seguintes comandos.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

Os logs do banco de dados devem conter uma entrada semelhante à seguinte.

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

Para obter informações sobre como visualizar os logs, consulte Monitorar arquivos de log do Amazon Aurora.

Para saber mais sobre a extensão pgAudit, consulte pgAudit no GitHub.

Excluir usuários ou bancos de dados do registro em log de auditoria

Conforme discutido em Arquivos de log do banco de dados Aurora PostgreSQL, os logs do PostgreSQL consomem espaço de armazenamento. O uso da extensão pgAudit aumenta o volume de dados reunidos nos logs em vários graus, dependendo das alterações monitoradas. Talvez você não precise auditar todos os usuários nem bancos de dados no cluster de banco de dados do Aurora PostgreSQL.

Para minimizar os impactos no armazenamento e evitar a captura desnecessária de registros de auditoria, você pode excluir usuários e bancos de dados da auditoria. Você também pode alterar o registro em log em determinada sessão. Os exemplos a seguir mostram o procedimento.

> **Note**
>
> As configurações de parâmetros no nível da sessão têm precedência sobre as configurações no grupo de parâmetros de cluster de banco de dados personalizado para a instância do gravador do cluster de banco de dados do Aurora PostgreSQL. Se você não quiser que os usuários do banco de dados ignorem suas configurações de registro em log de auditoria, não se esqueça de alterar as permissões.

Suponha que seu cluster banco de dados do Aurora RDS PostgreSQL esteja configurado) para auditar o mesmo nível de atividade para todos os usuários e bancos de dados. Depois, decida que não quer auditar o usuário `myuser`. Você pode desativar a auditoria para `myuser` com o comando SQL a seguir.

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

Depois, você pode usar a consulta a seguir para conferir a coluna `user_specific_settings` para `pgaudit.log` a fim de confirmar se o parâmetro está definido como `NONE`.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

Você deve ver a saída da forma a seguir.

```
 user_name | user_specific_settings
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

Você pode desativar o registro em log de determinado usuário no meio da sessão com o banco de dados com o comando a seguir.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

Use a consulta a seguir para conferir a coluna de configurações de pgaudit.log para uma combinação específica de usuário e banco de dados.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
```

```
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

Você verá uma saída semelhante à seguinte.

```
  user_name | database_name |     settings
-----------+---------------+-----------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

Depois de desativar a auditoria de `myuser`, você decide que não deseja monitorar as alterações em `mydatabase`. Você pode desativar a auditoria para esse banco de dados específico usando o comando a seguir.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

Depois, use a consulta a seguir para conferir a coluna database_specific_settings a fim de confirmar se pgaudit.log está definido como NONE.

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
a.datname = 'mydatabase';
```

Você deve ver a saída da forma a seguir.

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

Para restaurar as configurações padrão para myuser, use o seguinte comando:

```
ALTER USER myuser RESET pgaudit.log;
```

Para restaurar as configurações padrão para um banco de dados, use o comando a seguir.

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

Para restaurar as configurações padrão de usuário e banco de dados, use o comando a seguir.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

Você também pode capturar eventos específicos no log definindo `pgaudit.log` como um dos outros valores permitidos para o parâmetro `pgaudit.log`. Para ter mais informações, consulte Lista de configurações permitidas para o parâmetro `pgaudit.log`.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

Referência para a extensão pgAudit

Você pode especificar o nível de detalhes que deseja para o log de auditoria alterando um ou mais dos parâmetros listados nesta seção.

Controlar o comportamento da pgAudit

Você pode controlar o registro em log de auditoria alterando um ou mais dos parâmetros listados na tabela a seguir.

Parameter	Descrição
`pgaudit.log`	Especifica as classes de declaração que serão registradas pelo registro em log de auditoria de sessão. Os valores permitido s incluem ddl, function, misc, read, role, write, none, all. Para obter mais informações, consulte Lista de configurações permitidas para o parâmetro `pgaudit.log` .

Parameter	Descrição
`pgaudit.log_catalog`	Quando ativado (definido como 1), adiciona declarações à trilha de auditoria se todas as relações em uma declaração estiverem em pg_catalog.
`pgaudit.log_level`	Especifica o nível de log que será usado para entradas de log. Valores permitidos: debug5, debug4, debug3, debug2, debug1, info, notice, warning, log
`pgaudit.log_parameter`	Quando ativado (definido como 1), os parâmetros passados com a declaração são capturados no log de auditoria.
`pgaudit.log_relation`	Quando ativado (definido como 1), o log de auditoria da sessão cria uma entrada de log separada para cada relação (TABLE, VIEW etc.) referenciada em uma declaração SELECT ou DML.
`pgaudit.log_statement_once`	Especifica se o registro incluirá o texto e os parâmetros da instrução com a primeira entrada de log para uma combinação de instrução/subinstrução ou com cada entrada.
`pgaudit.role`	Especifica a função primária a ser usada para o registro em log de auditoria de objetos. A única entrada permitida é `rds_pgaudit` .

Lista de configurações permitidas para o parâmetro **`pgaudit.log`**

Valor	Descrição
nenhuma	Esse é o padrão. Nenhuma alteração no banco de dados é registrada.
todas	Registra tudo (read, write, function, role, ddl, misc).
ddl	Registra todas as declarações de linguagem de definição de dados (DDL) não incluídas na classe `ROLE`.
função	Registra chamadas de função e blocos de `DO`.

Valor	Descrição
misc	Registra comandos diversos, como `DISCARD`, `FETCH`, `CHECKPOINT` , `VACUUM` e `SET`.
leitura	Registra `SELECT` e `COPY` quando a fonte é uma relação (como uma tabela) ou uma consulta.
perfil	Registra declarações relacionadas a funções e privilégios, como `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
write	Registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando o destino é uma relação (tabela).

Para registrar vários tipos de eventos com auditoria de sessões, use uma lista separada por vírgulas. Para registrar todos os tipos de eventos, defina `pgaudit.log` para `ALL`. Reinicie a instância de banco de dados para aplicar as alterações.

Com a auditoria de objetos, você pode refinar o registro em log de auditoria para trabalhar com relações específicas. Por exemplo, você pode especificar que deseja o registro em log de auditoria para operações `READ` em uma ou mais tabelas.

Usar pglogical para sincronizar dados entre instâncias

Todas as versões do Aurora PostgreSQL atualmente disponíveis são compatíveis com a extensão `pglogical`. A extensão pglogical é anterior ao recurso de replicação lógica funcionalmente similar que foi introduzido pelo PostgreSQL na versão 10. Para obter mais informações, consulte Visão geral da replicação lógica do PostgreSQL com o Aurora.

A extensão `pglogical` é compatível com a replicação lógica entre dois ou mais clusters de banco de dados Aurora PostgreSQL. Ela também é compatível com a replicação entre diferentes versões do PostgreSQL e entre bancos de dados executados em instâncias de banco de dados RDS para PostgreSQL e clusters de banco de dados Aurora PostgreSQL. A extensão `pglogical` usa um modelo de publicação e assinatura para replicar alterações em tabelas e outros objetos, como sequências, de um editor para um assinante. Ela depende de um slot de replicação para garantir que as alterações sejam sincronizadas de um nó do editor para um nó assinante, definido da seguinte forma.

- O nó do editor é o cluster de banco de dados Aurora PostgreSQL, que é a fonte de dados a serem replicados para outros nós. O nó do editor define as tabelas a serem replicadas em um conjunto de publicações.
- O nó do assinante é o cluster de banco de dados RDS para PostgreSQL que recebe atualizações WAL do editor. O assinante cria uma assinatura para se conectar ao editor e obter os dados WAL decodificados. Quando o assinante cria a assinatura, o slot de replicação é criado no nó do editor.

Depois, você pode encontrar informações sobre a configuração da extensão `pglogical`.

Tópicos

- Requisitos e limitações da extensão pglogical
- Configurar a extensão pglogical
- Configurar a replicação lógica para o cluster de banco de dados Aurora PostgreSQL
- Restabelecer a replicação lógica após uma atualização principal
- Gerenciar slots de replicação lógica para o Aurora PostgreSQL
- Referência de parâmetros da extensão pglogical

Requisitos e limitações da extensão pglogical

Todas as versões atualmente disponíveis do Aurora PostgreSQL são compatíveis com a extensão `pglogical`.

Tanto o nó do editor quanto o do assinante devem estar configurados para replicação lógica.

As tabelas que você deseja replicar de um publicador para um assinante devem ter os mesmos nomes e o mesmo esquema. Essas tabelas também devem conter as mesmas colunas, e as colunas devem usar os mesmos tipos de dados. As tabelas de editores e assinantes devem ter as mesmas chaves primárias. Recomendamos que você use somente a CHAVE PRIMÁRIA como restrição exclusiva.

As tabelas no nó do assinante podem ter mais restrições permissivas do que as do nó do editor para restrições CHECK e NOT NULL.

A extensão `pglogical` fornece recursos como replicação bidirecional que não são compatíveis com o recurso de replicação lógica incorporado ao PostgreSQL (versão 10 e superior). Para obter mais informações, consulte PostgreSQL bi-directional replication using pglogical (Replicação bidirecional do PostgreSQL usando pglogical).

Configurar a extensão pglogical

Para configurar a extensão `pglogical` em seu cluster de banco de dados Aurora PostgreSQL, adicione `pglogical` às bibliotecas compartilhadas no grupo de parâmetros de cluster de banco de dados personalizado para seu cluster de banco de dados Aurora PostgreSQL. Você também precisa definir o valor do parâmetro `rds.logical_replication` como `1`, para ativar a decodificação lógica. Finalmente, você cria a extensão no banco de dados. Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para essas tarefas.

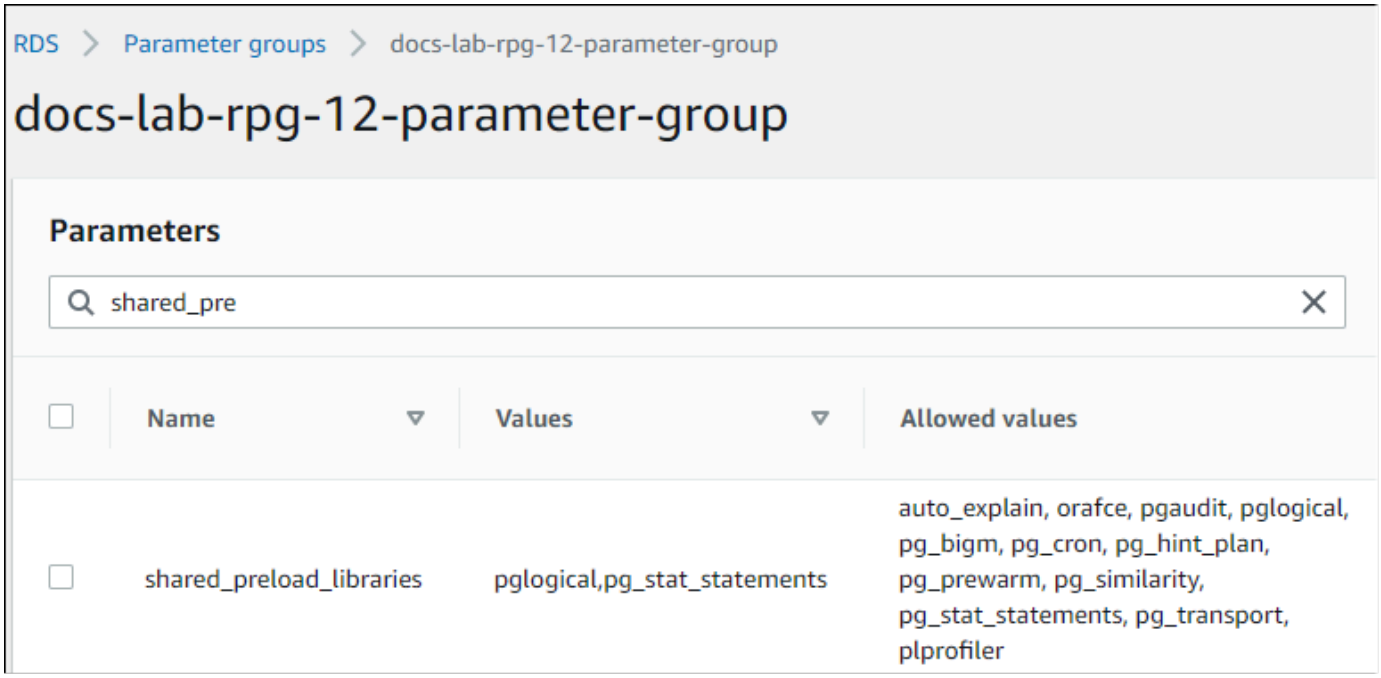
Você deve ter permissões como a função `rds_superuser` para realizar essas tarefas.

As etapas a seguir pressupõem que o cluster de banco de dados do Aurora PostgreSQL está associado a um grupo de parâmetros de cluster de banco de dados. Para obter informações sobre como criar um grupo de parâmetros de cluster de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.

Console

Como configurar a extensão pglogical

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione sua instância do gravador do cluster de banco de dados do Aurora PostgreSQL.
3. Abra a guia Configuration (Configuração) para sua instância do gravador de cluster de banco de dados do Aurora PostgreSQL. Entre os detalhes da instância, encontre o link Parameter group (Grupo de parâmetros).
4. Clique no link para abrir os parâmetros personalizados associados ao seu cluster de banco de dados do Aurora PostgreSQL.
5. No campo Parâmetros, digite `shared_pre` para encontrar o parâmetro `shared_preload_libraries`.
6. Selecione Edit parameters (Editar parâmetros) para acessar os valores das propriedades.
7. Adicione `pglogical` à lista no campo Values (Valores). Use uma vírgula para separar itens na lista de valores.



8. Encontre o parâmetro `rds.logical_replication` e defina-o como 1 para ativar a replicação lógica.
9. Reinicialize a instância do gravador do cluster de banco de dados do Aurora PostgreSQL para que suas alterações tenham efeito.
10. Quando a instância estiver disponível, você poderá usar `psql` (ou pgAdmin) para se conectar à instância do gravador do cluster de banco de dados Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. Para verificar se pglogical foi inicializada, execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. Verifique a configuração que permite a decodificação lógica da forma a seguir.

```
SHOW wal_level;
wal_level
-----------
 logical
```

```
(1 row)
```

13. Crie a extensão da forma a seguir.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Escolha Salvar alterações.
15. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
16. Selecione a instância do gravador do cluster de banco de dados do Aurora PostgreSQL na lista de bancos de dados para selecioná-la e depois selecione Reboot (Reinicializar) no menu Actions (Ações).

AWS CLI

Como configurar a extensão pglogical

Para configurar a pglogical usando a AWS CLI, chame a operação modify-db-parameter-group para modificar determinados parâmetros em seu grupo de parâmetros personalizado, conforme mostrado no procedimento a seguir.

1. Use o comando AWS CLI a seguir para adicionar `pglogical` ao parâmetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Use o comando AWS CLI a seguir para definir `rds.logical_replication` como `1` a fim de ativar o recurso de decodificação lógica para a instância de gravador do cluster de banco de dados do Aurora PostgreSQL.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
```

```
    --region aws-region
```

3. Use o comando AWS CLI a seguir para reinicializar a instância do gravador de seu cluster de banco de dados Aurora PostgreSQL para que a biblioteca da pglogical seja inicializada.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

4. Quando a instância estiver disponível, use `psql` para se conectar à instância do gravador do cluster de banco de dados Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. Crie a extensão da forma a seguir.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. Reinicie a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL usando o comando AWS CLI a seguir.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Configurar a replicação lógica para o cluster de banco de dados Aurora PostgreSQL

O procedimento a seguir mostra como iniciar a replicação lógica entre dois clusters de banco de dados Aurora PostgreSQL. As etapas pressupõem que tanto a fonte (editor) quanto o destino (assinante) tenham a extensão `pglogical` configurada conforme detalhado em Configurar a extensão pglogical.

> **Note**
>
> O `node_name` de um nó assinante não pode começar com `rds`.

Como criar o nó do editor e definir as tabelas a serem replicadas

Estas etapas pressupõem que o cluster de banco de dados do Aurora PostgreSQL tenha uma instância de gravador com um banco de dados com uma ou mais tabelas que você deseja replicar para outro nó. Você precisa recriar a estrutura da tabela do editor no assinante, então, primeiro, obtenha a estrutura da tabela, se necessário. Você pode fazer isso usando o metacomando `psql \d` *`tablename`* e criando a mesma tabela na instância do assinante. O procedimento a seguir cria uma tabela de exemplo no editor (fonte) para fins de demonstração.

1. Use `psql` para se conectar à instância que tem a tabela que você deseja usar como fonte para assinantes.

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   Se você não tiver uma tabela, crie uma tabela de exemplo da forma a seguir.

   a. Crie uma tabela de exemplo usando a declaração SQL a seguir.

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. Preencha a tabela com dados gerados usando a instrução SQL a seguir.

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

   c. Verifique se os dados existem na tabela usando a declaração SQL a seguir.

      ```
      SELECT count(*) FROM docs_lab_table;
      ```

2. Identifique esse cluster de banco de dados do Aurora PostgreSQL como o nó do editor da forma a seguir.

   ```
   SELECT pglogical.create_node(
       node_name := 'docs_lab_provider',
       dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
    dbname=labdb');
    create_node
   -------------
      3410995529
   ```

```
(1 row)
```

3. Adicione a tabela que você deseja replicar ao conjunto de replicação padrão. Para obter mais informações sobre conjuntos de replicação, consulte Replication sets (Conjuntos de replicação) na documentação da pglogical.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  ---------------------------
  t
  (1 row)
```

A configuração do nó do editor está concluída. Agora você pode configurar o nó de assinante para receber as atualizações do editor.

Como configurar o nó de assinante e criar uma assinatura para receber atualizações

Estas etapas pressupõem que o cluster de banco de dados do Aurora PostgreSQL tenha sido configurado com a extensão `pglogical`. Para obter mais informações, consulte Configurar a extensão pglogical.

1. Use `psql` para se conectar à instância em que você deseja receber atualizações do editor.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. No cluster de banco de dados do Aurora PostgreSQL do assinante, crie a mesma tabela que existe no editor. Neste exemplo, a tabela é `docs_lab_table`. Você pode criar a tabela da seguinte maneira.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. Verifique se essa tabela está vazia.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. Identifique esse cluster de banco de dados Aurora PostgreSQL como o nó do assinante da forma a seguir.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. Crie a assinatura.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

Ao concluir essa etapa, os dados da tabela no editor são criados na tabela no assinante. Se você quiser verificar se isso ocorreu, verifique a consulta SQL a seguir.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

Desse ponto em diante, as alterações feitas na tabela do editor são replicadas na tabela do assinante.

Restabelecer a replicação lógica após uma atualização principal

Antes de realizar uma atualização de versão principal de um cluster de banco de dados do Aurora PostgreSQL que está sendo configurado como um nó de editor para replicação lógica, você deve eliminar todos os slots de replicação, mesmo aqueles que não estão ativos. Recomendamos que você desvie temporariamente as transações do banco de dados do nó do editor, elimine os slots de replicação, atualize o cluster de banco de dados do Aurora PostgreSQL, e, depois, restabeleça e reinicie a replicação.

Os slots de replicação são hospedados somente no nó do editor. O nó de assinante do Aurora PostgreSQL em um cenário de replicação lógica não tem slots a serem descartados. O processo de atualização da versão principal do Aurora PostgreSQL é compatível com a atualização do assinante para uma nova versão principal do PostgreSQL, independente do nó do editor. No entanto, o processo de atualização interrompe o processo de replicação e interfere na sincronização dos dados WAL entre o nó do editor e o nó do assinante. Você precisa restabelecer a replicação lógica entre o editor e o assinante depois de atualizar o editor, o assinante ou ambos. O procedimento a seguir mostra como determinar se a replicação foi interrompida e como resolver o problema.

Determinar se a replicação lógica foi interrompida

Você pode determinar se o processo de replicação foi interrompido consultando o nó do editor ou o nó do assinante da forma a seguir.

Como conferir o nó do editor

- Use `psql` para se conectar ao nó do editor e, depois, consultar a função `pg_replication_slots`. Observe o valor na coluna ativa. Normalmente, isso retornará `t` (true) mostrando que a replicação está ativa. Se a consulta retornar `f` (false), é uma indicação de que a replicação para o assinante foi interrompida.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

Como conferir o nó do assinante

No nó do assinante, você pode conferir o status da replicação de três maneiras diferentes.

- Examine os logs do PostgreSQL no nó do assinante para encontrar mensagens de falha. O log identifica falhas com mensagens que incluem o código de saída 1, conforme mostrado a seguir.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- Consulte a função `pg_replication_origin`. Conecte-se ao banco de dados no nó do assinante usando `psql` e consulte a função `pg_replication_origin` da forma a seguir.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  O conjunto de resultados vazio significa que a replicação foi interrompida. Normalmente, você deve ver a saída da forma a seguir.

```
   roident |                       roname
  ---------+-----------------------------------------------------
         1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
  (1 row)
```

- Consulte a função `pglogical.show_subscription_status` conforme exibido no exemplo a seguir.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
---====----------------+--------+-----------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

  Essa saída mostra que a replicação foi interrompida. Seu status é down. Normalmente, a saída mostra o status como `replicating`.

Se seu processo de replicação lógica tiver sido interrompido, você poderá restabelecer a replicação seguindo estas etapas.

Como restabelecer a replicação lógica entre os nós do editor e do assinante

Para restabelecer a replicação, primeiro você desconecta o assinante do nó do editor e depois restabelece a assinatura, conforme descrito nestas etapas.

1. Conecte-se ao nó do assinante usando `psql` da forma a seguir.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Desative a assinatura usando a função `pglogical.alter_subscription_disable`.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. Obtenha o identificador do nó do editor consultando a `pg_replication_origin` da forma a seguir.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+--------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. Use a resposta da etapa anterior com o comando `pg_replication_origin_create` para atribuir o identificador que pode ser usado pela assinatura quando restabelecida.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. Ative a assinatura passando seu nome com um status de `true`, conforme exibido no exemplo a seguir.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
```

```
 t
(1 row)
```

Consulte o status do nó. Seu status deve ser `replicating` conforme mostrado neste exemplo.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-------------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

Confira o status do slot de replicação do assinante no nó do editor. A coluna `active` do slot deve retornar `t` (true), indicando que a replicação foi restabelecida.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                   slot_name              |      plugin      | slot_type | active
------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

Gerenciar slots de replicação lógica para o Aurora PostgreSQL

Antes de realizar uma atualização de versão principal de uma instância de gravador do cluster de banco de dados do Aurora PostgreSQL que está atuando como um nó de editor em um cenário de replicação lógica, você deve eliminar os slots de replicação na instância. O processo de pré-conferência da atualização da versão principal notifica você de que a atualização não pode continuar até que os slots sejam eliminados.

Para identificar os slots de replicação que foram criados usando a extensão `pglogical`, faça login em cada banco de dados e obtenha o nome dos nós. Ao consultar o nó do assinante, você obtém os nós do editor e do assinante na saída, conforme mostrado neste exemplo.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+-------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
```

```
(2 rows)
```

Você pode obter os detalhes sobre a assinatura com a consulta a seguir.

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

Agora você pode cancelar a assinatura da forma a seguir.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

Depois de cancelar a assinatura, você pode excluir o nó.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
-----------
 t
(1 row)
```

Você pode verificar se o nó não existe mais da forma a seguir.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

Referência de parâmetros da extensão pglogical

Na tabela, você pode encontrar parâmetros associados à extensão `pglogical`. Parâmetros como `pglogical.conflict_log_level` e `pglogical.conflict_resolution` são usados para lidar com conflitos de atualização. Podem surgir conflitos quando alterações são feitas localmente nas mesmas tabelas que estão inscritas para receber alterações do editor. Os conflitos também

podem ocorrer durante vários cenários, como replicação bidirecional ou quando vários assinantes estão se replicando do mesmo editor. Para obter mais informações, consulte PostgreSQL bi-directional replication using pglogical (Replicação bidirecional do PostgreSQL usando pglogical).

Parameter	Descrição
pglogical.batch_inserts	Inserções em lote, se possível. Não definido por padrão. Mude para “1” para ativar, “0” para desativar.
pglogical.conflict_log_level	Define o nível de log a ser usado para registrar em log conflitos resolvidos. Os valores de string compatíveis são debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, panic.
pglogical.conflict_resolution	Define o método a ser usado para resolver conflitos quando eles podem ser resolvidos. Os valores de string compatíveis são error, apply_remote, keep_local, last_update_wins, first_update_wins.
pglogical.extra_connection_options	Opções de conexão para adicionar a todas as conexões de nó de pares.
pglogical.synchronous_commit	Valor de confirmação síncrona específica do pglogical
pglogical.use_spi	Use a SPI (interface de programação de servidores) em vez da API de baixo nível para aplicar alterações. Defina como “1” para ativar, “0” para desativar. Para obter mais informações sobre a SPI, consulte Server Programming Interface (Interface de programação de servidores) na documentação do PostgreSQL.

Trabalhar com os invólucros de dados externos compatíveis do Amazon Aurora PostgreSQL

Um Foreign Data Wraper (FDW – Empacotador de dados externos) é um tipo específico de extensão que fornece acesso a dados externos. Por exemplo, a extensão `oracle_fdw` permite que a instância de banco de dados do Aurora PostgreSQL funcione com bancos de dados Oracle.

A seguir, você encontrará informações sobre vários invólucros de dados externos do PostgreSQL compatíveis.

Tópicos

- Usar a extensão log_fdw para acessar o log de banco de dados usando SQL
- Usar a extensão postgres_fdw para acessar dados externos
- Trabalhar com bancos de dados MySQL usando a extensão mysql_fdw
- Trabalhar com um banco de dados Oracle usando a extensão oracle_fdw
- Trabalhar com bancos de dados do SQL Server usando a extensão tds_fdw

Usar a extensão log_fdw para acessar o log de banco de dados usando SQL

O cluster do Aurora PostgreSQL DB é compatível com a extensão `log_fdw`, que permite acessar o log do mecanismo de banco de dados usando uma interface SQL. A extensão `log_fdw` apresenta duas novas funções que facilitam a criação de tabelas externas para logs de banco de dados:

- `list_postgres_log_files` – lista os arquivos no diretório do log do banco de dados e o tamanho do arquivo em bytes.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – cria uma tabela externa para o arquivo especificado no banco de dados atual.

Todas as funções criadas por `log_fdw` são de propriedade do `rds_superuser`. Os membros da função `rds_superuser` podem conceder acesso a essas funções para outros usuários do banco de dados.

Por padrão, os arquivos de log são gerados pelo Amazon Aurora no formato (erro padrão) `stderr`, conforme especificado no parâmetro `log_destination`. Existem apenas duas opções para esse parâmetro, `stderr` e `csvlog` (valores separados por vírgula, CSV). Se você adicionar a opção `csvlog` ao parâmetro, o Amazon Aurora gerará os logs `stderr` e `csvlog`. Isso pode afetar a capacidade de armazenamento em seu cluster de banco de dados. Portanto, você precisa estar ciente dos outros parâmetros que afetam o processamento de logs. Para obter mais informações, consulte Definir o destino dos logs (`stderr, csvlog`).

Um benefício de gerar logs do `csvlog` é que a extensão `log_fdw` permite criar tabelas externas com dados divididos ordenadamente em várias colunas. Para fazer isso, sua instância precisa

estar associada a um grupo de parâmetros de banco de dados personalizado para que você possa alterar a configuração para `log_destination`. Para obter mais informações sobre como fazer isso, consulte Grupos de parâmetros para Amazon Aurora.

O exemplo a seguir pressupõe que o parâmetro `log_destination` inclua `cvslog`.

Para usar a extensão log_fdw

1. Instale a extensão `log_fdw`.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Crie o servidor de log como um wrapper externo de dados.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Selecione todos os arquivos de log na lista.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

   A seguir você encontra um exemplo de resposta.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. Crie uma tabela com uma única coluna "log_entry" para o arquivo selecionado.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

   A resposta não fornece nenhum detalhe além de que a tabela agora existe.

```
---------------------------------
```

```
(1 row)
```

5. Selecione um exemplo de arquivo de log. O código a seguir recupera o horário do log e a descrição da mensagem de erro.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

A seguir você encontra um exemplo de resposta.

```
            log_time             |                                 message
---------------------------------
+--------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

Usar a extensão postgres_fdw para acessar dados externos

Você pode acessar dados em uma tabela em um servidor de banco de dados remoto com a extensão postgres_fdw. Se você configurar uma conexão remota usando a instância de banco de dados do PostgreSQL, o acesso também estará disponível para a réplica de leitura.

Para usar postgres_fdw para acessar um servidor de banco de dados remoto

1. Instale a extensão postgres_fdw.

```
CREATE EXTENSION postgres_fdw;
```

2. Crie um servidor de dados externo usando CREATE SERVER.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Crie um mapeamento de usuário para identificar a função a ser usada no servidor remoto.

> **Important**
>
> Para ocultar a senha a fim de que ela não apareça nos logs, defina `log_statement=none` no nível da sessão. A configuração no nível do parâmetro não oculta a senha.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Crie uma tabela que mapeia para a tabela no servidor remoto.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Trabalhar com bancos de dados MySQL usando a extensão mysql_fdw

Para acessar um banco de dados compatível com MySQL pelo cluster de banco de dados do Aurora PostgreSQL, você pode instalar e usar a extensão `mysql_fdw`. Esse invólucro de dados externos permite que você trabalhe com o RDS for MySQL, o Aurora MySQL, o MariaDB e outros bancos de dados compatíveis com MySQL. A conexão do cluster de banco de dados do RDS para PostgreSQL ao banco de dados MySQL é criptografada com base no melhor esforço, dependendo das configurações do cliente e do servidor. No entanto, você pode impor a criptografia, se quiser. Para obter mais informações, consulte Usar criptografia em trânsito com a extensão.

A extensão `mysql_fdw` é compatível com o Amazon Aurora PostgreSQL versões 15.4, 14.9, 13.12, 12.16, e posteriores. Ela é compatível com seleções, inserções, atualizações e exclusões de um banco de dados do RDS for PostgreSQL para tabelas em uma instância de banco de dados compatível com MySQL.

Tópicos

- Configurar um banco de dados Aurora PostgreSQL para usar a extensão mysql_fdw

- Exemplo: trabalhar com um banco de dados do Aurora MySQL pelo Aurora PostgreSQL
- Usar criptografia em trânsito com a extensão

Configurar um banco de dados Aurora PostgreSQL para usar a extensão mysql_fdw

Para configurar a extensão `mysql_fdw` no cluster de banco de dados do Aurora PostgreSQL, é necessário carregar a extensão no cluster de banco de dados e, em seguida, criar o ponto de conexão com a instância de banco de dados do MySQL. Para essa tarefa, você precisa ter os seguintes detalhes sobre a instância de banco de dados do MySQL:

- Nome de host ou endpoint. Para um cluster de banco de dados do Aurora MySQL, é possível encontrar o endpoint usando o console. Escolha a guia “Connectivity & security” (Conectividade e segurança) e procure na seção “Endpoint and port” (Endpoint e porta).
- Número da porta. O número da porta padrão do MySQL é 3306.
- O nome do banco de dados. O identificador do banco de dados.

Você também precisa fornecer acesso no grupo de segurança ou na lista de controle de acesso (ACL) para a porta 3306 do MySQL. Tanto o cluster de banco de dados do Aurora PostgreSQL como o cluster de banco de dados do Aurora . Se o acesso não estiver configurado corretamente, ao tentar se conectar à tabela compatível com o MySQL, será exibida uma mensagem de erro semelhante à seguinte:

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

No procedimento a seguir, você (como a conta `rds_superuser`) cria o servidor externo. Depois, você concede acesso ao servidor externo a usuários específicos. Em seguida, esses usuários criam seus próprios mapeamentos para as contas de usuário apropriadas do MySQL para trabalhar com a instância de banco de dados do MySQL.

Para usar mysql_fdw a fim de acessar um servidor de banco de dados MySQL

1. Conecte-se à instância de banco de dados do PostgreSQL usando uma conta que tenha a função `rds_superuser`. Se tiver aceitado os padrões ao criar o cluster de banco de dados do Aurora PostgreSQL, o nome de usuário será `postgres` e você poderá se conectar usando a ferramenta da linha de comando `psql` da seguinte forma:

```
psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Instale a extensão `mysql_fdw` da seguinte forma:

```
postgres=> CREATE EXTENSION mysql_fdw;
CREATE EXTENSION
```

Depois que a extensão é instalada no cluster de banco de dados do Aurora PostgreSQL, é possível configurar o servidor externo que fornece a conexão com um banco de dados MySQL.

Para criar o servidor externo

Execute essas tarefas no cluster de banco de dados do Aurora PostgreSQL. As etapas presumem que você esteja conectado como usuário com privilégios `rds_superuser`, como `postgres`.

1. Crie um servidor externo no cluster de banco de dados do Aurora PostgreSQL:

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-
name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Conceda aos usuários apropriados acesso ao servidor externo. Eles devem ser usuários não administradores, ou seja, usuários sem a função `rds_superuser`.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

Os usuários do PostgreSQL criam e gerenciam suas próprias conexões com o banco de dados do MySQL por meio do servidor externo.

Exemplo: trabalhar com um banco de dados do Aurora MySQL pelo Aurora PostgreSQL

Suponha que você tenha uma tabela simples em uma instância de banco de dados do Aurora PostgreSQL. Os usuários do Aurora PostgreSQL querem consultar os itens `SELECT`, `INSERT`, `UPDATE` e `DELETE` nessa tabela. Suponha que a extensão `mysql_fdw` foi criada na instância de banco de dados do RDS for PostgreSQL, conforme detalhado no procedimento anterior. Depois de

se conectar à instância de banco de dados do RDS for PostgreSQL como um usuário com privilégios `rds_superuser`, é possível prosseguir com as etapas abaixo.

1. Crie um servidor externo no cluster de banco de dados do Aurora PostgreSQL:

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Conceda uso a um usuário que não tenha permissões `rds_superuser`; por exemplo, `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. Conecte-se como *`user1`* e, em seguida, crie um mapeamento para o usuário do MySQL:

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. Crie uma tabela externa vinculada a uma tabela do MySQL:

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. Execute uma consulta simples na tabela externa:

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. Você pode adicionar, alterar e remover dados da tabela do MySQL. Por exemplo:

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

   Execute a consulta `SELECT` novamente para ver os resultados:

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

Usar criptografia em trânsito com a extensão

A conexão com o MySQL pelo Aurora PostgreSQL usa criptografia em trânsito (TLS/SSL) por padrão. No entanto, a conexão volta para não criptografada quando a configuração do cliente e do servidor é diferente. Você pode impor a criptografia para todas as conexões de saída especificando a opção `REQUIRE SSL` nas contas de usuário do RDS for MySQL. Essa mesma abordagem também funciona para contas de usuário do MariaDB e do Aurora MySQL.

Para contas de usuário do MySQL configuradas como `REQUIRE SSL`, a tentativa de conexão falhará se não for possível estabelecer uma conexão segura.

Para aplicar criptografia a contas de usuário do banco de dados do MySQL existentes, você pode usar o comando `ALTER USER`. A sintaxe varia, dependendo da versão do MySQL, conforme mostrado na tabela a seguir. Para obter mais informações, consulte ALTER USER no Manual de referência do MySQL.

MySQL 5.7, MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

Para obter mais informações sobre a extensão `mysql_fdw`, consulte a documentação do mysql_fdw.

Trabalhar com um banco de dados Oracle usando a extensão oracle_fdw

Para acessar um banco de dados Oracle pelo cluster de banco de dados do Aurora PostgreSQL , você pode instalar e usar a extensão `oracle_fdw`. Essa extensão é um invólucro de dados externos para bancos de dados Oracle. Para saber mais sobre a extensão, consulte a documentação do oracle_fdw.

A extensão `oracle_fdw` é compatível com o Aurora PostgreSQL 12.7 (Amazon Aurora versão 4.2) e versões posteriores.

Tópicos

- Ativação da extensão oracle_fdw
- Exemplo: usar um servidor externo vinculado a um banco de dados Amazon RDS for Oracle
- Como trabalhar com criptografia em trânsito
- Noções básicas da visualização e das permissões de pg_user_mappings

Ativação da extensão oracle_fdw

Para usar a extensão oracle_fdw, realize o procedimento a seguir.

Para ativar a extensão oracle_fdw

- Execute o comando a seguir usando uma conta que tenha as permissões de `rds_superuser`.

```
CREATE EXTENSION oracle_fdw;
```

Exemplo: usar um servidor externo vinculado a um banco de dados Amazon RDS for Oracle

O exemplo a seguir mostra o uso de um servidor externo vinculado a um banco de dados Amazon RDS for Oracle.

Para criar um servidor externo vinculado a um banco de dados do RDS for Oracle

1. Na instância de banco de dados do RDS for Oracle, observe:
   - Endpoint
   - Port
   - Database name
2. Crie um servidor externo.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. Conceda uso a um usuário que não tenha privilégios `rds_superuser`, por exemplo `user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. Conecte-se como `user1` e crie um mapeamento para um usuário Oracle.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Crie uma tabela estrangeira vinculada a uma tabela Oracle.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. Consulte a tabela externa.

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

Se a consulta relatar o seguinte erro, verifique seu grupo de segurança e a lista de controle de acesso (ACL) para verificar se as duas instâncias podem se comunicar.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

Como trabalhar com criptografia em trânsito

A criptografia PostgreSQL-to-Oracle em trânsito é baseada em uma combinação de parâmetros de configuração de cliente e servidor. Para obter um exemplo usando o Oracle 21c, consulte About the Values for Negotiating Encryption and Integrity (Sobre os valores para negociar criptografia e integridade) na documentação do Oracle. Como o cliente usado para oracle_fdw no Amazon RDS está configurado com `ACCEPTED`, a criptografia depende da configuração do servidor de banco de dados do Oracle e utiliza a Oracle Security Library (libnnz) para isso.

Se o banco de dados estiver no RDS for Oracle, consulte Criptografia de rede nativa do Oracle para configurar a criptografia.

Noções básicas da visualização e das permissões de pg_user_mappings

O catálogo PostgreSQL `pg_user_mapping` armazena o mapeamento de um usuário do Aurora PostgreSQL para o usuário em um servidor de dados externo (remoto). O acesso ao catálogo é restrito, mas você usa a visualização `pg_user_mappings` para ver os mapeamentos. Veja a seguir um exemplo que mostra como as permissões se aplicam a um banco de dados Oracle de exemplo, mas essas informações se aplicam de forma mais geral a qualquer wrapper de dados externo.

Na saída a seguir, você pode encontrar funções e permissões mapeadas para três usuários de exemplo diferentes. Usuários eo `rdssu1` e `rdssu2` são membros da função `rds_superuser`, e `user1` não é. O exemplo usa o metacomando `\du` do `psql` para listar as funções existentes.

```
test=>  \du
                                                              List of roles
    Role name    |                          Attributes                          |
                  Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                              |
 {rds_superuser}
 rdssu2          |                                                              |
 {rds_superuser}
 user1           |                                                              | {}
```

Todos os usuários, incluindo aqueles com privilégios `rds_superuser`, têm permissão para visualizar seus próprios mapeamentos de usuário (`umoptions`) na tabela `pg_user_mappings`. Como mostrado no exemplo a seguir, quando `rdssu1` tenta obter todos os mapeamentos do usuário, é gerado um erro, mesmo com privilégios `rdssu1rds_superuser`:

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

Veja a seguir alguns exemplos:

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
```

```
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

Devido às diferenças na implementação de `information_schema._pg_user_mappings` e `pg_catalog.pg_user_mappings`, um `rds_superuser` criado manualmente requer outras permissões para visualizar senhas no `pg_catalog.pg_user_mappings`.

Nenhuma permissão adicional é necessária para um `rds_superuser` visualizar senhas no `information_schema._pg_user_mappings`.

Usuários que não tenham a função `rds_superuser` poderão visualizar senhas em `pg_user_mappings` somente nas seguintes condições:

- O usuário atual é o usuário que está sendo mapeado e é proprietário do servidor ou mantém o privilégio `USAGE` sobre ele.
- O usuário atual é o proprietário do servidor, e o mapeamento é para `PUBLIC`.

Trabalhar com bancos de dados do SQL Server usando a extensão tds_fdw

Você pode usar a extensão `tds_fdw` do PostgreSQL para acessar bancos de dados compatíveis com o protocolo de fluxo de dados tabular (TDS), como bancos de dados do Sybase e do Microsoft

SQL Server. Esse invólucro de dados externos permite que você se conecte pelo cluster de banco de dados do Aurora PostgreSQL a bancos de dados que usam o protocolo TDS, incluindo o Amazon RDS for Microsoft SQL Server. Para obter mais informações, consulte a documentação do tds-fdw/tds_fdw no GitHub.

A extensão `tds_fdw` é compatível com o Amazon Aurora PostgreSQL versão 13.6 e posterior.

Configurar o banco de dados do RDS for PostgreSQL para usar a extensão tds_fdw

Nos procedimentos a seguir, você pode encontrar um exemplo de configuração e uso de `tds_fdw` com um cluster de banco de dados do Aurora PostgreSQL. Antes de se conectar a um banco de dados do SQL Server usando `tds_fdw`, é necessário obter os seguintes detalhes da instância:

- Nome de host ou endpoint. Para uma instância de banco de dados do RDS for SQL Server, você pode encontrar o endpoint usando o console. Escolha a guia “Connectivity & security” (Conectividade e segurança) e procure na seção “Endpoint and port” (Endpoint e porta).
- Número da porta. O número da porta padrão para o Microsoft SQL Server é 1433.
- O nome do banco de dados. O identificador do banco de dados.

Você também precisa fornecer acesso no grupo de segurança ou na lista de controle de acesso (ACL) para a porta 1433 do SQL Server. Tanto o cluster de banco de dados do Aurora PostgreSQL como a instância de banco de dados do RDS for SQL Server precisam de acesso à porta 1433. Se o acesso não estiver configurado corretamente, ao tentar consultar o Microsoft SQL Server, será exibida a seguinte mensagem de erro:

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

Para usar tds_fdw a fim de se conectar a um banco de dados do SQL Server

1. Conecte-se à instância primária do cluster de banco de dados do Aurora PostgreSQL usando uma conta que tenha a função `rds_superuser`:

   ```
   psql --host=your-cluster-name-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=test --password
   ```

2. Instale a extensão `tds_fdw`:

```
test=> CREATE EXTENSION tds_fdw;
CREATE EXTENSION
```

Depois que a extensão for instalada no cluster de banco de dados do Aurora PostgreSQL , configure o servidor externo.

Para criar o servidor externo

Execute essas tarefas no cluster de banco de dados do Aurora PostgreSQL usando uma conta com privilégios `rds_superuser`.

1. Crie um servidor externo no cluster de banco de dados do Aurora PostgreSQL:

   ```
   test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
    (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
    'tds_fdw_testing');
   CREATE SERVER
   ```

   Para acessar dados não ASCII do lado do SQLServer, crie um link de servidor com a opção character_set no cluster de banco de dados do Aurora PostgreSQL:

   ```
   test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
    'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
    character_set 'UTF-8');
   CREATE SERVER
   ```

2. Conceda uso a um usuário que não tenha permissões da função `rds_superuser`; por exemplo, `user1`:

   ```
   test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
   ```

3. Conecte-se como user1 e, em seguida, crie um mapeamento para um usuário do SQL Server:

   ```
   test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
    'sqlserveruser', password 'password');
   CREATE USER MAPPING
   ```

4. Crie uma tabela externa vinculada a uma tabela do SQL Server:

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. Consulte a tabela externa:

```
test=> SELECT * FROM mytab;
 a
---
 1
(1 row)
```

Usar criptografia em trânsito para a conexão

A conexão do Aurora PostgreSQL com o SQL Server usa criptografia em trânsito (TLS/SSL), dependendo da configuração do banco de dados do SQL Server. Se o SQL Server não estiver configurado para criptografia, o cliente do RDS for PostgreSQL que faz a solicitação ao banco de dados do SQL Server retornará ao estado de não criptografado.

Você pode aplicar a criptografia para a conexão com instâncias de banco de dados do RDS for SQL Server definindo o parâmetro `rds.force_ssl`. Para saber como, consulte Forçar conexões com a instância de banco de dados para usar SSL. Para obter mais informações sobre a configuração de SSL/TLS para o RDS for SQL Server, consulte Usar SSL com uma instância de banco de dados do Microsoft SQL Server.

Trabalhar com Trusted Language Extensions para PostgreSQL

O Trusted Language Extensions para PostgreSQL é um kit de desenvolvimento de código aberto para criar extensões do PostgreSQL. Ele permite que você crie extensões do PostgreSQL de alta performance e as execute com segurança em seu cluster de banco de dados do Aurora PostgreSQL. Ao usar o Trusted Language Extensions (TLE) para PostgreSQL, você pode criar extensões do PostgreSQL que sigam a abordagem documentada para estender a funcionalidade do PostgreSQL. Para obter mais informações, consulte Packaging Related Objects into an Extension (Compactar objetos relacionados em uma extensão) na documentação do PostgreSQL.

Um dos principais benefícios do TLE é que você pode usá-lo em ambientes que não fornecem acesso ao sistema de arquivos subjacente à instância do PostgreSQL. Anteriormente, a instalação de uma nova extensão exigia acesso ao sistema de arquivos. O TLE remove essa restrição. Ele fornece um ambiente de desenvolvimento para criar extensões para qualquer banco de dados do PostgreSQL, como as executadas em seus clusters de banco de dados do Aurora PostgreSQL.

O TLE foi projetado para impedir o acesso a recursos inseguros para as extensões criadas com o uso do TLE. Seu ambiente de execução limita o impacto de qualquer defeito de extensão em uma única conexão de banco de dados. O TLE também oferece aos administradores de banco de dados um controle refinado sobre quem pode instalar extensões e fornece um modelo de permissões para executá-las.

O TLE é compatível com o Aurora PostgreSQL versão 14.5 e versões posteriores.

O ambiente de desenvolvimento e de execução do Trusted Language Extensions são empacotados como a extensão `pg_tle` do PostgreSQL, versão 1.0.1. É compatível com a criação de extensões em JavaScript, Perl, Tcl, PL/pgSQL e SQL. Você instala a extensão `pg_tle` em seu cluster de banco de dados do Aurora PostgreSQL da mesma forma que instala outras extensões do PostgreSQL. Depois de configurar `pg_tle`, os desenvolvedores podem usá-lo para criar extensões do PostgreSQL, conhecidas como extensões TLE.

Nos tópicos a seguir, você encontrará informações sobre como configurar Trusted Language Extensions e como começar a criar suas próprias extensões TLE.

Tópicos

- Terminologia
- Requisitos para usar Trusted Language Extensions para PostgreSQL

- Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL
- Visão geral do Trusted Language Extensions para PostgreSQL
- Criar extensões TLE para Aurora PostgreSQL
- Descartar suas extensões TLE de um banco de dados
- Desinstalar o Trusted Language Extensions para PostgreSQL
- Usar ganchos do PostgreSQL com suas extensões TLE
- Referência de funções para Trusted Language Extensions para PostgreSQL
- Referência de ganchos para Trusted Language Extensions para PostgreSQL

Terminologia

Para ajudar você a entender melhor o Trusted Language Extensions, consulte o glossário a seguir para conhecer os termos usados neste tópico.

Trusted Language Extensions para PostgreSQL

: Trusted Language Extensions para PostgreSQL é o nome oficial do kit de desenvolvimento de código aberto que vem com a extensão `pg_tle`. Ele está disponível para uso em qualquer sistema PostgreSQL. Para obter mais informações, consulte aws/pg_tle no GitHub.

Trusted Language Extensions

: Trusted Language Extensions é o nome abreviado de Trusted Language Extensions para PostgreSQL. Esse nome curto e sua abreviatura (TLE) também são utilizados nesta documentação.

linguagem confiável

: Uma linguagem confiável é uma linguagem de programação ou script que tem atributos de segurança específicos. Por exemplo, as linguagens confiáveis geralmente restringem o acesso ao sistema de arquivos e limitam o uso de propriedades de rede especificadas. O kit de desenvolvimento TLE foi projetado para oferecer suporte a linguagens confiáveis. O PostgreSQL é compatível com várias linguagens diferentes que são usadas para criar extensões confiáveis ou não confiáveis. Por exemplo, consulte Trusted and Untrusted PL/Perl (PL/Perl confiável e não confiável) na documentação do PostgreSQL. Quando você cria uma extensão com o uso de Trusted Language Extensions, a extensão usa inerentemente mecanismos de linguagem confiáveis.

Extensão TLE

Uma extensão TLE é uma extensão do PostgreSQL criada com o uso do kit de desenvolvimento Trusted Language Extensions (TLE).

Requisitos para usar Trusted Language Extensions para PostgreSQL

A seguir são apresentados os requisitos para configurar e usar o kit de desenvolvimento TLE.

- Versões do Aurora PostgreSQL: o Trusted Language Extensions só é compatível com o Aurora PostgreSQL versão 14.5 e versões posteriores.
  - Se você precisar atualizar seu cluster de banco de dados do Aurora PostgreSQL, consulte Como atualizar os clusters de banco de dados de Amazon Aurora MySQL.
  - Se você ainda não tem um cluster de banco de dados do Aurora, que execute o PostgreSQL, você pode criar um . Para ter mais informações, consulte Criar um cluster de banco de dados do Aurora PostgreSQL e se conectar a ele.
- Requer privilégios **`rds_superuser`**: para instalar e configurar a extensão `pg_tle`, sua função de usuário do banco de dados deve ter as permissões da função `rds_superuser`. Por padrão, essa função é concedida ao usuário `postgres` que cria o cluster de banco de dados do Aurora PostgreSQL.
- Requer um grupo de parâmetros de banco de dados personalizado: seu cluster de banco de dados do Aurora PostgreSQL deve ser configurado com um grupo de parâmetros de banco de dados personalizado. Use o grupo de parâmetros de banco de dados personalizado para a instância de gravador de seu cluster de banco de dados do Aurora PostgreSQL.
  - Se seu cluster de banco de dados do Aurora PostgreSQL não estiver configurado com um grupo de parâmetros de banco de dados personalizado, você deverá criar um e associá-lo à instância de gravador de seu cluster de banco de dados do Aurora PostgreSQL. Para obter um breve resumo das etapas, consulte Criar e aplicar um grupo de parâmetros de banco de dados personalizado.
  - Se seu cluster de banco de dados do Aurora PostgreSQL já estiver configurado usando um grupo de parâmetros de banco de dados, você poderá configurar o Trusted Language Extensions. Para obter detalhes, consulte Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

Criar e aplicar um grupo de parâmetros de banco de dados personalizado

Use as etapas a seguir para criar um grupo de parâmetros de banco de dados personalizado e configurar seu cluster de banco de dados do Aurora PostgreSQL para usá-lo.

Console

Como criar um grupo de parâmetros de banco de dados personalizado e usá-lo com o cluster de banco de dados do Aurora PostgreSQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Selecione Parameter groups (Grupos de parâmetros) no menu do Amazon RDS
3. Escolha Criar grupo de parâmetros.
4. Na página Parameter group details (Detalhes do grupo de parâmetros), insira as informações a seguir.
   - Em Parameter group family (Família de grupos de parâmetros), selecione aurora-postgres14.
   - Em Type (Tipo), selecione DB Parameter Group (Grupo de parâmetros de banco de dados).
   - Em Group name (Nome do grupo), forneça ao seu grupo de parâmetros um nome significativo no contexto de suas operações.
   - Em Description (Descrição), insira uma descrição útil para que outras pessoas de sua equipe possam encontrá-lo facilmente.
5. Escolha Criar. Seu grupo de parâmetros de banco de dados personalizado é criado em sua Região da AWS. Agora você pode modificar seu cluster de banco de dados do Aurora PostgreSQL para usá-lo seguindo as próximas etapas.
6. Selecione Databases (Bancos de dados) no menu do Amazon RDS.
7. Selecione o cluster de banco de dados do Aurora PostgreSQL que você deseja usar com o TLE dentre as opções listadas e, depois, selecione Modify (Modificar).
8. Na página Modify DB cluster settings (Modificar configurações do cluster de banco de dados), encontre Database options (Opções de banco de dados) e use o seletor para selecionar seu grupo de parâmetros de banco de dados personalizado.
9. Selecione Continue (Continuar) para salvar a alteração.
10. Selecione Apply immediately (Aplicar imediatamente) para que você possa continuar configurando o cluster de banco de dados do Aurora PostgreSQL para usar o TLE.

Para continuar configurando seu sistema para Trusted Language Extensions, consulte Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

Para obter mais informações sobre como trabalhar com grupos de parâmetros de banco de dados e cluster de banco de dados, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

AWS CLI

Você pode evitar especificar o argumento `--region` ao usar comandos da CLI configurando sua AWS CLI com sua Região da AWS padrão. Para obter mais informações, consulte Conceitos básicos de configuração da no Guia do usuário do AWS Command Line Interface.

Como criar um grupo de parâmetros de banco de dados personalizado e usá-lo com o cluster de banco de dados do Aurora PostgreSQL

1. Use o comando create-db-parameter-group da AWS CLI para criar um grupo de parâmetros de banco de dados personalizado com base em aurora-postgresql14 para sua Região da AWS. Observe que nesta etapa você cria um grupo de parâmetros de banco de dados para aplicar à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family aurora-postgresql14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Para Windows:

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family aurora-postgresql14 ^
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Seu grupo de parâmetros de banco de dados personalizado está disponível em sua Região da AWS, para que você possa modificar a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para usá-lo.

2. Use o comando modify-db-instance da AWS CLI para aplicar seu grupo de parâmetros de banco de dados personalizado à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL. Esse comando reinicia imediatamente a instância ativa.

   Para Linux, macOS ou Unix:

   ```
   aws rds modify-db-instance \
     --region aws-region \
     --db-instance-identifier your-writer-instance-name \
     --db-parameter-group-name custom-params-for-pg-tle \
     --apply-immediately
   ```

   Para Windows:

   ```
   aws rds modify-db-instance ^
     --region aws-region ^
     --db-instance-identifier your-writer-instance-name ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --apply-immediately
   ```

Para continuar configurando seu sistema para Trusted Language Extensions, consulte Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

Para ter mais informações, consulte Grupos de parâmetros de banco de dados para instâncias de banco de dados do Amazon Aurora .

Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL

As etapas a seguir pressupõem que seu cluster de banco de dados do Aurora PostgreSQL esteja associado a um grupo de parâmetros de cluster de banco de dados. Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para essas etapas.

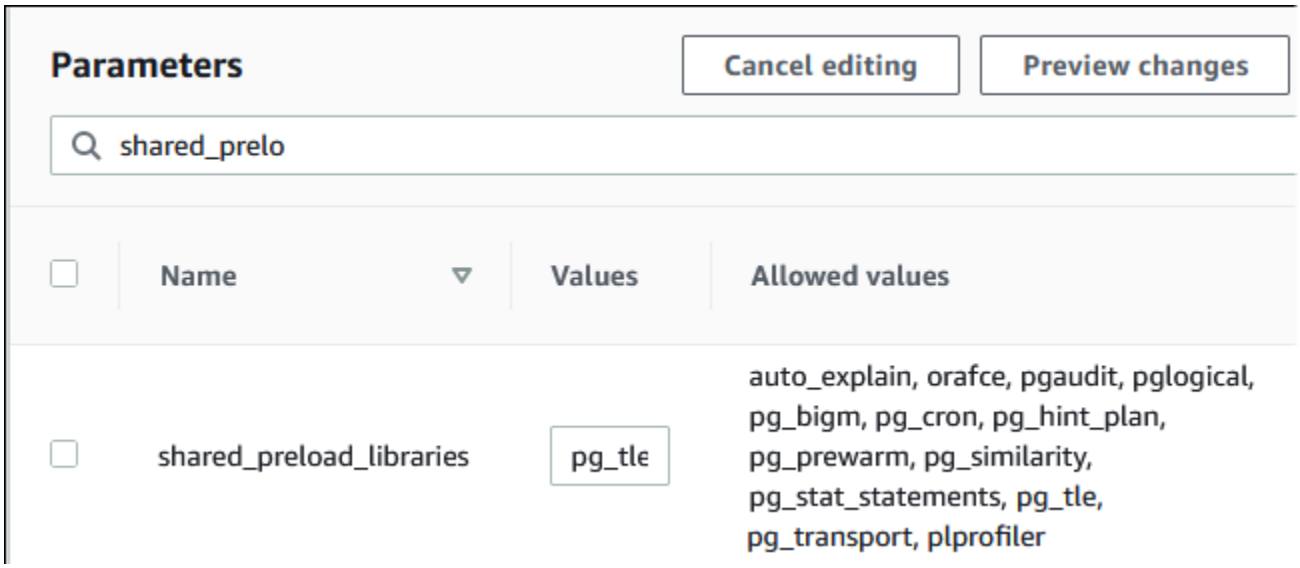
Ao configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL , você o instala em um banco de dados específico para uso pelos usuários do banco de dados que têm permissões nesse banco de dados.

Console

Como configurar o Trusted Language Extensions

Execute as etapas a seguir usando uma conta que seja membro do grupo `rds_superuser` (função).

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione sua instância do gravador do cluster de banco de dados do Aurora PostgreSQL.
3. Abra a guia Configuration (Configuração) para sua instância do gravador de cluster de banco de dados do Aurora PostgreSQL. Entre os detalhes da instância, encontre o link Parameter group (Grupo de parâmetros).
4. Clique no link para abrir os parâmetros personalizados associados ao seu cluster de banco de dados do Aurora PostgreSQL.
5. No campo Parameters (Parâmetros), digite `shared_pre` para encontrar o parâmetro `shared_preload_libraries`.
6. Selecione Edit parameters (Editar parâmetros) para acessar os valores das propriedades.
7. Adicione `pg_tle` à lista no campo Values (Valores). Use uma vírgula para separar itens na lista de valores.



8. Reinicie a instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL para que a alteração no parâmetro `shared_preload_libraries` tenha efeito.

9. Quando a instância estiver disponível, verifique se `pg_tle` foi inicializado. Use `psql` para se conectar à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL e depois execute o comando a seguir.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. Com a extensão `pg_tle` inicializada, agora você pode criar a extensão.

```
CREATE EXTENSION pg_tle;
```

Para verificar se a extensão está instalada, você pode usar o metacomando `psql` a seguir.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. Conceda a função `pgtle_admin` ao nome de usuário principal que você criou para seu cluster de banco de dados do Aurora PostgreSQL ao configurá-la. Se você aceitou o padrão, é `postgres`.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

É possível verificar se a concessão ocorreu usando o metacomando `psql` conforme mostrado no exemplo a seguir. Somente as funções `pgtle_admin` e `postgres` são mostradas na saída. Para ter mais informações, consulte Noções básicas de perfis e permissões do PostgreSQL.

```
labdb=> \du
                          List of roles
    Role name    |           Attributes            |              Member of
-----------------+---------------------------------
+----------------------------------
pgtle_admin     | Cannot login                    | {}
```

```
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. Feche a sessão `psql` usando o metacomando `\q`.

```
\q
```

Para começar a criar extensões TLE, consulte Exemplo: Criar uma extensão de linguagem confiável usando SQL.

AWS CLI

Você pode evitar especificar o argumento `--region` ao usar comandos da CLI configurando sua AWS CLI com sua Região da AWS padrão. Para obter mais informações, consulte Conceitos básicos de configuração da no Guia do usuário do AWS Command Line Interface.

Como configurar o Trusted Language Extensions

1. Utilize o comando modify-db-parameter-group AWS CLI para adicionar `pg_tle` ao parâmetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilize o comando reboot-db-instance da AWS CLI para reinicializar a instância de seu cluster de banco de dados do Aurora PostgreSQL e inicialize a biblioteca `pg_tle`.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando a instância estiver disponível, verifique se a `pg_tle` foi inicializada. Use `psql` para se conectar à instância do gravador de seu cluster de banco de dados do Aurora PostgreSQL e, depois, execute o comando a seguir.

```
SHOW shared_preload_libraries;
```

```
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

Com `pg_tle` inicializado, agora você pode criar a extensão.

```
CREATE EXTENSION pg_tle;
```

4. Conceda a função `pgtle_admin` ao nome de usuário principal que você criou para seu cluster de banco de dados do Aurora PostgreSQL ao configurá-la. Se você aceitou o padrão, é `postgres`.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

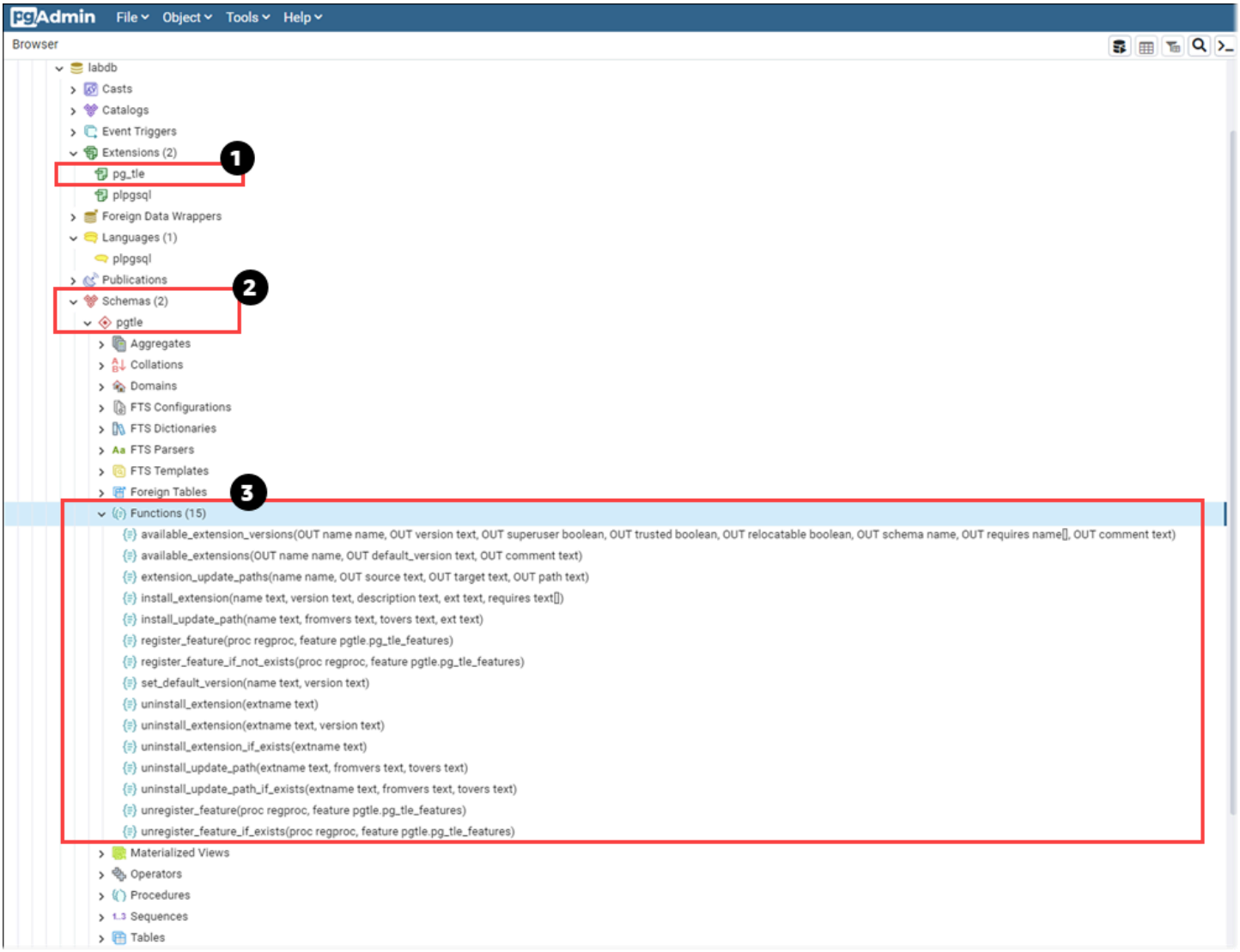
5. Feche a sessão `psql` da forma a seguir.

```
labdb=> \q
```

Para começar a criar extensões TLE, consulte Exemplo: Criar uma extensão de linguagem confiável usando SQL.

Visão geral do Trusted Language Extensions para PostgreSQL

O Trusted Language Extensions para PostgreSQL é uma extensão do PostgreSQL que você instala em seu cluster de banco de dados do Aurora PostgreSQL da mesma forma que configura outras extensões do PostgreSQL. Na imagem a seguir de um exemplo de banco de dados na ferramenta cliente pgAdmin, você pode ver alguns dos componentes que compõem a extensão `pg_tle`.



É possível ver os detalhes a seguir.

1. O kit de desenvolvimento Trusted Language Extensions para PostgreSQL é embalado como a extensão `pg_tle`. Assim, o `pg_tle` é adicionado às extensões disponíveis para o banco de dados no qual está instalado.
2. O TLE tem seu próprio esquema, `pgtle`. Esse esquema contém funções auxiliares (3) para instalar e gerenciar as extensões criadas.
3. O TLE fornece mais de uma dúzia de funções auxiliares para instalar, registrar e gerenciar suas extensões. Para saber mais sobre essas funções, consulte Referência de funções para Trusted Language Extensions para PostgreSQL.

São outros componentes da extensão `pg_tle`:

- A função **`pgtle_admin`**: a função `pgtle_admin` é criada quando a extensão `pg_tle` é instalada. Essa função é privilegiada e deve ser tratada como tal. É altamente recomendável seguir o princípio de privilégio mínimo ao conceder a função `pgtle_admin` a usuários de banco de dados. Em outras palavras, conceda a função `pgtle_admin` somente aos usuários do banco de dados que tenham permissão para criar, instalar e gerenciar novas extensões TLE, como `postgres`.
- A tabela **`pgtle.feature_info`**: `pgtle.feature_info` é uma tabela protegida que contém informações sobre seus TLEs, ganchos e as funções e os procedimentos armazenados personalizados utilizados. Se você tiver privilégios `pgtle_admin`, use as funções de Trusted Language Extensions a seguir para adicionar e atualizar essas informações na tabela.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

Criar extensões TLE para Aurora PostgreSQL

Você pode instalar qualquer extensão criada com o TLE em qualquer cluster de banco de dados do Aurora PostgreSQL que tenha a extensão `pg_tle` instalada. A extensão `pg_tle` tem como escopo o banco de dados PostgreSQL no qual ela está instalada. As extensões que você cria usando o TLE têm como escopo o mesmo banco de dados.

Use as várias funções `pgtle` para instalar o código que compõe sua extensão TLE. As funções do Trusted Language Extensions a seguir exigem a função `pgtle_admin`.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension (nome)
- pgtle.uninstall_extension (nome, versão)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists

- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

Exemplo: Criar uma extensão de linguagem confiável usando SQL

O exemplo a seguir mostra como criar uma extensão TLE chamada `pg_distance` que contém algumas funções SQL para calcular distâncias usando fórmulas diferentes. Na lista, você pode encontrar a função para calcular a distância de Manhattan e a função para calcular a distância euclidiana. Para obter mais informações sobre a diferença entre essas fórmulas, consulte Geometria taxicab Geometria euclidiana na Wikipedia.

Você poderá usar esse exemplo em seu próprio cluster de banco de dados do Aurora PostgreSQL se tiver a extensão `pg_tle` configurada conforme detalhado em Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

> **Note**
>
> Você precisa ter os privilégios da função `pgtle_admin` para seguir esse procedimento.

Como criar o exemplo de extensão TLE

As etapas a seguir usam um exemplo de banco de dados chamado `labdb`. Esse banco de dados é de propriedade do usuário primário `postgres`. A função `postgres` também tem as permissões da função `pgtle_admin`.

1. Use o `psql` para se conectar à instância de gravador do cluster de banco de dados do Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Crie uma extensão TLE denominada `pg_distance` copiando o código a seguir e colando-o no console da sessão `psql`.

   ```
   SELECT pgtle.install_extension
   (
    'pg_distance',
    '0.1',
   ```

```
  'Distance functions for two points',
$_pg_tle_$
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

Você verá a saída da forma a seguir.

```
install_extension
---------------
 t
(1 row)
```

Os artefatos que compõem a extensão `pg_distance` agora estão instalados em seu banco de dados. Esses artefatos incluem o arquivo de controle e o código da extensão, que são itens que precisam estar presentes para que a extensão possa ser criada usando o comando `CREATE EXTENSION`. Em outras palavras, você ainda precisa criar a extensão para disponibilizar suas funções aos usuários do banco de dados.

3. Para criar a extensão, use o comando `CREATE EXTENSION` como você faz com qualquer outra extensão. Assim como acontece com outras extensões, o usuário do banco de dados precisa ter as permissões `CREATE` no banco de dados.

```
CREATE EXTENSION pg_distance;
```

4. Para testar a extensão TLE `pg_distance`, você pode usá-la para calcular a distância de Manhattan entre quatro pontos.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

Para calcular a distância euclidiana entre o mesmo conjunto de pontos, você pode usar o seguinte.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

A extensão `pg_distance` carrega as funções no banco de dados e as disponibiliza para qualquer usuário com permissões no banco de dados.

Modificar a extensão TLE

Para melhorar a performance da consulta para as funções empacotadas nessa extensão TLE, adicione os dois atributos do PostgreSQL a seguir às suas especificações.

- `IMMUTABLE`: o atributo `IMMUTABLE` garante que o otimizador de consultas possa usar otimizações para melhorar os tempos de resposta da consulta. Para obter mais informações, consulte Function Volatility Categories (Categorias de volatilidade de funções) na documentação do PostgreSQL.
- `PARALLEL SAFE`: o atributo `PARALLEL SAFE` é outro atributo que permite que o PostgreSQL execute a função no modo paralelo. Para obter mais informações, consulte CREATE FUNCTION na documentação do PostgreSQL.

No exemplo a seguir, você pode ver como a função `pgtle.install_update_path` é usada para adicionar esses atributos a cada função para criar uma versão `0.2` da extensão TLE `pg_distance`. Para ter mais informações sobre essa função, consulte pgtle.install_update_path. Você precisa ter a função `pgtle_admin` para realizar essa tarefa.

Como atualizar uma extensão TLE existente e especificar a versão padrão

1. Conecte-se à instância de gravador de seu cluster de banco de dados do Aurora PostgreSQL usando `psql` ou outra ferramenta de cliente, como o pgAdmin

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
```

```
--port=5432 --username=postgres --password --dbname=labdb
```

2. Modifique a extensão TLE existente copiando o código a seguir e colando-o no console da sessão `psql`.

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

Você verá uma resposta semelhante ao seguinte.

```
install_update_path
--------------------
 t
(1 row)
```

Você pode tornar essa versão da extensão a versão padrão, para que os usuários do banco de dados não precisem especificar uma versão ao criar ou atualizar a extensão em seu banco de dados.

3. Para especificar que a versão modificada (versão 0.2) de sua extensão TLE é a versão padrão, use a função `pgtle.set_default_version` conforme mostrado no exemplo a seguir.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

Para ter mais informações sobre essa função, consulte pgtle.set_default_version.

4. Com o código implementado, você pode atualizar a extensão TLE instalada da maneira usual, utilizando o comando `ALTER EXTENSION ... UPDATE`, conforme mostrado aqui:

```
ALTER EXTENSION pg_distance UPDATE;
```

Descartar suas extensões TLE de um banco de dados

Você pode descartar suas extensões TLE usando o comando `DROP EXTENSION` da mesma forma que faz com outras extensões do PostgreSQL. Descartar a extensão não remove os arquivos de instalação que a compõem, o que permite aos usuários recriar a extensão. Para remover a extensão e seus arquivos de instalação, execute o processo de duas etapas a seguir.

Como descartar a extensão TLE e remover seus arquivos de instalação

1. Use `psql` ou outra ferramenta de cliente para se conectar à instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=dbname
```

2. Descarte a extensão como você faria com qualquer extensão do PostgreSQL.

```
DROP EXTENSION your-TLE-extension
```

Por exemplo, se você criar a extensão `pg_distance` conforme detalhado em Exemplo: Criar uma extensão de linguagem confiável usando SQL, poderá descartar a extensão da forma a seguir.

```
DROP EXTENSION pg_distance;
```

Você vê uma saída confirmando que a extensão foi descartada da forma a seguir.

```
DROP EXTENSION
```

Neste ponto, a extensão não estará mais ativa no banco de dados. No entanto, seus arquivos de instalação e arquivo de controle ainda estão disponíveis no banco de dados, portanto, os usuários do banco de dados poderão criar a extensão novamente, se desejarem.

- Se quiser deixar os arquivos de extensão intactos para que os usuários do banco de dados possam criar sua extensão TLE, você pode parar neste ponto.
- Se quiser remover todos os arquivos que compõem a extensão, siga para a próxima etapa.

3. Para remover todos os arquivos de instalação da sua extensão, use a função `pgtle.uninstall_extension`. Essa função remove todos os arquivos de código e controle de sua extensão.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

Por exemplo, para remover todos os arquivos de instalação `pg_distance`, use o comando a seguir.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Desinstalar o Trusted Language Extensions para PostgreSQL

Se você não quiser mais criar suas próprias extensões TLE usando o TLE, poderá descartar a extensão `pg_tle` e remover todos os artefatos. Essa ação inclui descartar todas as extensões TLE no banco de dados e o esquema `pgtle`.

Como descartar a extensão **pg_tle** e seu esquema de um banco de dados

1. Use `psql` ou outra ferramenta de cliente para se conectar à instância do gravador do cluster de banco de dados do Aurora PostgreSQL.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password --dbname=dbname
   ```

2. Descarte a extensão `pg_tle` do banco de dados. Se o banco de dados tiver suas próprias extensões TLE ainda em execução no banco de dados, você também precisará descartar essas extensões. Para isso, você pode usar a palavra-chave `CASCADE`, conforme mostrado a seguir.

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   Se a extensão `pg_tle` ainda não estiver ativa no banco de dados, você não precisará usar a palavra-chave `CASCADE`.

3. Descarte o esquema `pgtle`. Essa ação remove todas as funções de gerenciamento do banco de dados.

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   O comando retornará o seguinte quando o processo for concluído.

   ```
   DROP SCHEMA
   ```

   A extensão `pg_tle`, seu esquema e funções e todos os artefatos são removidos. Para criar extensões usando o TLE, execute o processo de configuração novamente. Para ter mais informações, consulte Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

Usar ganchos do PostgreSQL com suas extensões TLE

Um gancho é um mecanismo de retorno de chamada disponível no PostgreSQL que possibilita aos desenvolvedores chamar funções personalizadas ou outras rotinas durante operações regulares de banco de dados. O kit de desenvolvimento TLE é compatível com ganchos do PostgreSQL para que você possa integrar funções personalizadas com o comportamento do PostgreSQL no ambiente de execução. Por exemplo, você pode usar um gancho para associar o processo de autenticação ao

seu próprio código personalizado ou modificar o processo de planejamento e execução de consultas de acordo com suas necessidades específicas.

Suas extensões TLE podem usar ganchos. Se um gancho tiver escopo global, ele se aplicará a todos os bancos de dados. Portanto, se sua extensão TLE usar um gancho global, você precisará criar sua extensão TLE em todos os bancos de dados que seus usuários puderem acessar.

Ao usar a extensão `pg_tle` para criar seu próprio Trusted Language Extensions, você pode usar os ganchos disponíveis de uma API SQL para criar as funções de sua extensão. Você deve registrar todos os ganchos com `pg_tle`. Para alguns ganchos, talvez você também precise definir vários parâmetros de configuração. Por exemplo, o gancho de verificação `passcode` pode ser configurado como ativado, desativado ou obrigatório. Para obter mais informações sobre os requisitos específicos dos ganchos `pg_tle` disponíveis, consulte Referência de ganchos para Trusted Language Extensions para PostgreSQL.

Exemplo: Criar uma extensão que use um gancho do PostgreSQL

O exemplo abordado nesta seção usa um gancho do PostgreSQL para conferir a senha fornecida durante operações específicas de SQL e impede que os usuários do banco de dados definam suas senhas como qualquer uma das contidas na tabela `password_check.bad_passwords`. A tabela contém as dez opções de senhas mais usadas, mas facilmente identificáveis.

Para configurar esse exemplo em seu cluster de banco de dados do Aurora PostgreSQL, você já deve ter instalado o Trusted Language Extensions. Para obter detalhes, consulte Configurar o Trusted Language Extensions em seu cluster de banco de dados do Aurora PostgreSQL.

Como configurar o exemplo de gancho de verificação de senha

1. Use o `psql` para se conectar à instância de gravador do cluster de banco de dados do Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Copie o código do Lista de códigos do gancho de verificação de senha e cole-o no banco de dados.

   ```
   SELECT pgtle.install_extension (
     'my_password_check_rules',
     '1.0',
     'Do not let users use the 10 most commonly used passwords',
   ```

```
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
```

```
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Quando a extensão for carregada em seu banco de dados, você verá uma saída como a seguinte.

```
 install_extension
-------------------
 t
(1 row)
```

3. Enquanto ainda estiver conectado ao banco de dados, agora você poderá criar a extensão.

```
CREATE EXTENSION my_password_check_rules;
```

4. Você pode confirmar que a extensão foi criada no banco de dados usando o metacomando `psql` a seguir.

```
\dx
                       List of installed extensions
          Name           | Version |   Schema   |
 Description
------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. Abra outra sessão do terminal para trabalhar com o AWS CLI. Você precisa modificar seu grupo de parâmetros de banco de dados personalizado para ativar o gancho de verificação de senha.

Para isso, use o comando modify-db-parameter-group da CLI, conforme mostrado no exemplo a seguir.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Poderá levar alguns minutos para que a alteração na configuração do grupo de parâmetros tenha efeito. No entanto, esse parâmetro é dinâmico, portanto, você não precisa reiniciar a instância de gravador do cluster de banco de dados do Aurora PostgreSQL para que a configuração tenha efeito.

6. Abra a sessão `psql` e consulte o banco de dados para verificar se o gancho password_check foi ativado.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

O gancho de verificação de senha agora está ativo. É possível testá-lo criando uma função e usando uma das senhas incorretas, conforme mostrado no exemplo a seguir.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

A saída foi formatada para facilitar a leitura.

O exemplo a seguir mostra que o comportamento do metacomando interativo `pgsql` `\password` também é afetado pelo gancho password_check.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

Você poderá descartar essa extensão TLE e desinstalar seus arquivos de origem, se desejar. Para obter mais informações, consulte Descartar suas extensões TLE de um banco de dados.

Lista de códigos do gancho de verificação de senha

O código de exemplo mostrado aqui define a especificação da extensão TLE `my_password_check_rules`. Quando você copia esse código e o cola em seu banco de dados, o código da extensão `my_password_check_rules` é carregado no banco de dados e o gancho `password_check` é registrado para uso pela extensão.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
```

```
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Referência de funções para Trusted Language Extensions para PostgreSQL

Veja a documentação de referência a seguir sobre as funções disponíveis em Trusted Language Extensions para PostgreSQL. Use essas funções para instalar, registrar, atualizar e gerenciar suas extensões TLE, ou seja, as extensões do PostgreSQL que você desenvolve usando o kit de desenvolvimento Trusted Language Extensions.

Funções

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension (nome)
- pgtle.uninstall_extension (nome, versão)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

A função `pgtle.available_extensions` é uma função de retorno de conjunto. Ela retorna todas as extensões TLE disponíveis no banco de dados. Cada linha retornada contém informações sobre uma única extensão TLE.

Protótipo de função

```
pgtle.available_extensions()
```

Função

Nenhum.

Argumentos

Nenhum.

Saída

- `name`: o nome da extensão TLE.
- `default_version`: a versão da extensão TLE a ser usada quando `CREATE EXTENSION` é chamada sem uma versão especificada.
- `description`: uma descrição mais detalhada sobre a extensão TLE.

Exemplo de uso

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

A função `available_extension_versions` é de retorno de conjunto. Ela retorna uma lista de todas as extensões TLE disponíveis e suas versões. Cada linha contém informações sobre uma versão específica de determinada extensão TLE, incluindo se ela requer uma função específica.

Protótipo de função

```
pgtle.available_extension_versions()
```

Função

Nenhum.

Argumentos

Nenhum.

Saída

- `name`: o nome da extensão TLE.
- `version`: a versão da extensão TLE.

- `superuser`: esse valor é sempre `false` para suas extensões TLE. As permissões necessárias para criar a extensão TLE ou atualizá-la são as mesmas para criar outros objetos em determinado banco de dados.
- `trusted`: esse valor é sempre `false` para uma extensão TLE.
- `relocatable`: esse valor é sempre `false` para uma extensão TLE.
- `schema`: especifica o nome do esquema no qual a extensão TLE está instalada.
- `requires`: uma matriz contendo os nomes de outras extensões necessárias para essa extensão TLE.
- `description`: uma descrição detalhada da extensão TLE.

Para obter mais informações sobre valores de saída, consulte Packaging Related Objects into an Extension > Extension Files (Compactação de objetos relacionados em uma extensão > arquivos de extensão) na documentação do PostgreSQL.

Exemplo de uso

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

A função `extension_update_paths` é de retorno de conjunto. Ela retorna uma lista de todos os caminhos de atualização possíveis para uma extensão TLE. Cada linha inclui as atualizações ou downgrades disponíveis para essa extensão TLE.

Protótipo de função

```
pgtle.extension_update_paths(name)
```

Função

Nenhum.

Argumentos

`name`: o nome da extensão TLE da qual obter caminhos de atualização.

Saída

- `source`: a versão de origem de uma atualização.

- `target`: a versão de destino de uma atualização.
- `path`: o caminho de atualização usado para atualizar uma extensão TLE da versão `source` para a `target`, por exemplo, `0.1--0.2`.

Exemplo de uso

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

A função `install_extension` permite que você instale os artefatos que compõem sua extensão TLE no banco de dados, após o qual ela pode ser criada usando o comando `CREATE EXTENSION`.

Protótipo de função

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

Função

Nenhum.

Argumentos

- `name`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `version`: a versão da extensão TLE.
- `description`: uma descrição detalhada da extensão TLE. Essa descrição é exibida no campo `comment` em `pgtle.available_extensions()`.
- `ext`: o conteúdo da extensão TLE. Esse valor contém objetos, como funções.
- `requires`: um parâmetro opcional que especifica dependências para essa extensão TLE. A extensão `pg_tle` é adicionada automaticamente como uma dependência.

Muitos desses argumentos são iguais aos incluídos em um arquivo de controle de extensão para instalar uma extensão do PostgreSQL no sistema de arquivos de uma instância do PostgreSQL. Para obter mais informações, consulte Extension Files (Arquivos de extensão) em Packaging Related Objects into an Extension (Compactação de objetos relacionados em uma extensão) na documentação do PostgreSQL.

Saída

Essa função retorna `OK` em caso de sucesso e `NULL` em caso de erro.

- `OK`: a extensão TLE foi instalada com êxito no banco de dados.
- `NULL`: a extensão TLE não foi instalada com êxito no banco de dados.

Exemplo de uso

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

A função `install_update_path` fornece um caminho de atualização entre duas versões diferentes de uma extensão TLE. Essa função permite que os usuários de sua extensão TLE atualizem sua versão usando a sintaxe `ALTER EXTENSION ... UPDATE`.

Protótipo de função

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

Função

`pgtle_admin`

Argumentos

- `name`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `fromvers`: a versão de origem da extensão TLE para a atualização.
- `tovers`: a versão de destino da extensão TLE para a atualização.

- `ext`: o conteúdo da atualização. Esse valor contém objetos, como funções.

Saída

Nenhum.

Exemplo de uso

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

A função `register_feature` adiciona o recurso interno especificado do PostgreSQL à tabela `pgtle.feature_info`. Os ganchos do PostgreSQL são um exemplo de um recurso interno do PostgreSQL. O kit de desenvolvimento Trusted Language Extensions é compatível com o uso de ganchos do PostgreSQL. Atualmente, essa função é compatível com o recurso a seguir.

- `passcheck`: registra o gancho de verificação de senha com seu procedimento ou função que personaliza o comportamento de verificação de senha do PostgreSQL.

Protótipo de função

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

Função

`pgtle_admin`

Argumentos

- `proc`: o nome de uma função ou um procedimento armazenado a ser usado para o recurso.
- `feature`: o nome do recurso `pg_tle` (como`passcheck`) a ser registrado na função.

Saída

Nenhum.

Exemplo de uso

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

A função `pgtle.register_feature_if_not_exists` adiciona o recurso do PostgreSQL especificado à tabela `pgtle.feature_info` e identifica a extensão TLE ou outro procedimento ou função que usa o recurso. Para obter mais informações sobre ganchos e Trusted Language Extensions, consulte Usar ganchos do PostgreSQL com suas extensões TLE.

Protótipo de função

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

Função

`pgtle_admin`

Argumentos

- `proc`: o nome de uma função ou um procedimento armazenado que contém a lógica (código) a ser usada como um recurso para sua extensão TLE. Por exemplo, o código `pw_hook`.
- `feature`: o nome do recurso do PostgreSQL a ser registrado na função TLE. Atualmente, o único recurso disponível é o gancho `passcheck`. Para ter mais informações, consulte Gancho de verificação de senha (passcheck).

Saída

Retorna `true` após registrar o recurso para a extensão especificada. Retorna `false` se o recurso já estiver registrado.

Exemplo de uso

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

A função `set_default_version` permite que você especifique uma `default_version` para sua extensão TLE. Você pode usar essa função para definir um caminho de atualização e designar a versão como padrão para sua extensão TLE. Quando os usuários do banco de dados especificam sua extensão TLE nos comandos `CREATE EXTENSION` e `ALTER EXTENSION ... UPDATE`, essa versão da extensão TLE é criada no banco de dados para esse usuário.

Essa função retorna `true` em caso de êxito. Se a extensão TLE especificada no argumento `name` não existir, a função retornará um erro. Da mesma forma, se a `version` da extensão TLE não existir, ela retornará um erro.

Protótipo de função

```
pgtle.set_default_version(name text, version text)
```

Função

`pgtle_admin`

Argumentos

- `name`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `version`: a versão da extensão TLE para definir o padrão.

Saída

- `true`: quando a configuração da versão padrão é bem-sucedida, a função retorna `true`.
- `ERROR`: retornará uma mensagem de erro se uma extensão TLE com o nome ou versão especificados não existir.

Exemplo de uso

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension (nome)

A função `uninstall_extension` remove todas as versões de uma extensão TLE de um banco de dados. Essa função impede que futuras chamadas de `CREATE EXTENSION` instalem a extensão TLE. Se a extensão TLE não existir no banco de dados, um erro será gerado.

A função `uninstall_extension` não descartará uma extensão TLE se ela estiver atualmente ativa no banco de dados. Para remover uma extensão TLE que está ativa no momento, você precisa chamar explicitamente `DROP EXTENSION` para removê-la.

Protótipo de função

```
pgtle.uninstall_extension(extname text)
```

Função

`pgtle_admin`

Argumentos

- `extname`: o nome da extensão TLE a ser desinstalada. Esse nome é o mesmo usado com `CREATE EXTENSION` para carregar a extensão TLE para uso em determinado banco de dados.

Saída

Nenhum.

Exemplo de uso

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension (nome, versão)

A função `uninstall_extension(name, version)` remove a versão especificada da extensão TLE do banco de dados. Essa função impede que `CREATE EXTENSION` e `ALTER EXTENSION` instalem ou atualizem uma extensão TLE para a versão especificada. Essa função também remove todos os caminhos de atualização para a versão especificada da extensão TLE. Essa função não desinstalará a extensão TLE se ela estiver atualmente ativa no banco de dados. Você deve chamar

explicitamente `DROP EXTENSION` para remover a extensão TLE. Para desinstalar todas as versões de uma extensão TLE, consultepgtle.uninstall_extension (nome).

Protótipo de função

```
pgtle.uninstall_extension(extname text, version text)
```

Função

`pgtle_admin`

Argumentos

- `extname`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `version`: a versão da extensão TLE a ser desinstalada do banco de dados.

Saída

Nenhum.

Exemplo de uso

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

A função `uninstall_extension_if_exists` remove todas as versões de uma extensão TLE de determinado banco de dados. Se a extensão TLE não existir, a função retornará silenciosamente (nenhuma mensagem de erro será gerada). Se a extensão especificada estiver atualmente ativa em um banco de dados, essa função não a descartará. Você deve chamar explicitamente `DROP EXTENSION` para remover a extensão TLE antes de usar essa função para desinstalar seus artefatos.

Protótipo de função

```
pgtle.uninstall_extension_if_exists(extname text)
```

Função

`pgtle_admin`

Argumentos

- `extname`: o nome da extensão TLE. Esse valor é usado ao chamar `CREATE EXTENSION`.

Saída

A função `uninstall_extension_if_exists` retorna `true` após a desinstalação da extensão especificada. Se a extensão especificada não existir, a função retornará `false`.

- `true`: retorna `true` após a desinstalação da extensão TLE.
- `false`: retorna `false` quando a extensão TLE não existe no banco de dados.

Exemplo de uso

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

A função `uninstall_update_path` remove o caminho de atualização especificado de uma extensão TLE. Isso impede `ALTER EXTENSION ... UPDATE TO` de usar isso como um caminho de atualização.

Se a extensão TLE estiver sendo usada atualmente por uma das versões desse caminho de atualização, ela permanecerá no banco de dados.

Se o caminho de atualização especificado não existir, essa função gerará um erro.

Protótipo de função

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

Função

`pgtle_admin`

Argumentos

- `extname`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `fromvers`: a versão de origem da extensão TLE usada no caminho de atualização.
- `tovers`: a versão de destino da extensão TLE usada no caminho de atualização.

Saída

Nenhum.

Exemplo de uso

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

A função `uninstall_update_path_if_exists` é semelhante a `uninstall_update_path` no sentido de remover o caminho de atualização especificado de uma extensão TLE. No entanto, se o caminho de atualização não existir, essa função não gerará uma mensagem de erro. Em vez disso, a função retornará `false`.

Protótipo de função

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

Função

`pgtle_admin`

Argumentos

- `extname`: o nome da extensão TLE. Esse valor é usado ao chamar`CREATE EXTENSION`.
- `fromvers`: a versão de origem da extensão TLE usada no caminho de atualização.
- `tovers`: a versão de destino da extensão TLE usada no caminho de atualização.

Saída

- `true`: a função atualizou com êxito o caminho da extensão TLE.
- `false`: a função não conseguiu atualizar o caminho da extensão TLE.

Exemplo de uso

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

A função `unregister_feature` fornece uma maneira de remover funções que foram registradas para usar recursos `pg_tle`, como ganchos. Para obter informações sobre como registrar uma recurso, consulte pgtle.register_feature.

Protótipo de função

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

Função

`pgtle_admin`

Argumentos

- `proc`: o nome de uma função armazenada a ser registrada em um recurso `pg_tle`.
- `feature`: o nome do recurso `pg_tle` a ser registrado na função. Por exemplo, `passcheck` é um recurso que pode ser registrado para uso pelas extensões de linguagem confiáveis desenvolvidas por você. Para ter mais informações, consulte Gancho de verificação de senha (passcheck).

Saída

Nenhum.

Exemplo de uso

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

A função `unregister_feature` fornece uma maneira de remover funções que foram registradas para usar recursos `pg_tle`, como ganchos. Para ter mais informações, consulte Usar ganchos do PostgreSQL com suas extensões TLE. Retorna `true` após cancelar o registro do recurso com êxito. Retorna `false` se o recurso não foi registrado.

Para obter informações sobre como registrar recursos `pg_tle` para suas extensões TLE, consulte pgtle.register_feature.

Protótipo de função

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

Função

`pgtle_admin`

Argumentos

- `proc`: o nome da função armazenada que foi registrada para incluir um recurso `pg_tle`.
- `feature`: o nome do recurso `pg_tle` que foi registrado com a extensão de linguagem confiável.

Saída

Retorna `true` ou `false` da maneira a seguir.

- `true`: a função cancelou com êxito o registro do recurso da extensão.
- `false`: a função não conseguiu cancelar o registro do recurso da extensão TLE.

Exemplo de uso

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Referência de ganchos para Trusted Language Extensions para PostgreSQL

O Trusted Language Extensions para PostgreSQL é compatível com ganchos do PostgreSQL. Um gancho é um mecanismo interno de retorno de chamada disponível para que os desenvolvedores estendam a funcionalidade principal do PostgreSQL. Usando ganchos, os desenvolvedores podem implementar suas próprias funções ou procedimentos para uso durante várias operações de banco de dados, modificando assim o comportamento do PostgreSQL de alguma forma. Por exemplo, você pode usar um gancho `passcheck` para personalizar a forma como o PostgreSQL manipula as senhas fornecidas ao criar ou alterar senhas para usuários (funções).

Veja a documentação a seguir para saber mais sobre o hook passcheck disponível para as extensões TLE. Para saber mais sobre os hooks disponíveis, incluindo o hook de autenticação do cliente, consulte Trusted Language Extensions hooks.

Gancho de verificação de senha (passcheck)

O gancho `passcheck` é usado para personalizar o comportamento do PostgreSQL durante o processo de verificação de senha para os comandos SQL e o metacomando `psql` a seguir.

- `CREATE ROLE username ...PASSWORD`: para obter mais informações, consulte CREATE ROLE na documentação do PostgreSQL.
- `ALTER ROLE username...PASSWORD`: para obter mais informações, consulte ALTER ROLE na documentação do PostgreSQL.
- `\password username`: esse metacomando interativo `psql` altera com segurança a senha do usuário especificado usando o hash da senha antes de usar a sintaxe `ALTER ROLE ... PASSWORD` de forma transparente. O metacomando é um invólucro seguro para o comando `ALTER ROLE ... PASSWORD`, portanto, o gancho se aplica ao comportamento do metacomando `psql`.

Para ver um exemplo, consulte Lista de códigos do gancho de verificação de senha.

Sumário

- Protótipo de função
- Argumentos
- Configuração
- Observações de uso

Protótipo de função

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

Argumentos

A função de gancho `passcheck` usa os seguintes argumentos:

- `username`: o nome (como texto) da função (nome de usuário) que está definindo uma senha.
- `password`: o texto simples ou a senha com hash. A senha digitada deve corresponder ao tipo especificado em `password_type`.
- `password_type`: especifique o formato `pgtle.password_type` da senha. Esse formato pode ser uma das opções a seguir.

  - `PASSWORD_TYPE_PLAINTEXT`: uma senha de texto simples.
  - `PASSWORD_TYPE_MD5`: uma senha que foi criptografada usando o algoritmo MD5 (resumo de mensagens 5).
  - `PASSWORD_TYPE_SCRAM_SHA_256`: uma senha que foi criptografada usando o algoritmo SCRAM-SHA-256.
- `valid_until`: especifique a hora em que a senha se torna inválida. Esse argumento é opcional. Se você usar esse argumento, especifique a hora como um valor `timestamptz`.
- `valid_null`: se esse valor booliano estiver definido como `true`, a opção `valid_until` será definida como `NULL`.

Configuração

A função `pgtle.enable_password_check` controla se o gancho passcheck está ativo. O gancho passcheck tem três configurações possíveis.

- `off`: desativa o gancho de verificação de senha `passcheck`. Este é o valor padrão.
- on: ativa o gancho de verificação de senha `passcode` para que as senhas sejam conferidas na tabela.
- `require`: requer que um gancho de verificação de senha seja definido.

Observações de uso

Para ativar ou desativar o gancho `passcheck`, você precisa modificar o grupo de parâmetros de banco de dados personalizado para a instância de gravação do seu cluster de banco de dados Aurora PostgreSQL.

Para Linux, macOS ou Unix:

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-parameter-group ^
```

```
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
"ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Referência do Amazon Aurora PostgreSQL

Nos tópicos a seguir, é possível encontrar informações sobre agrupamentos, funções, parâmetros e eventos de espera no Amazon Aurora PostgreSQL.

Tópicos

- Agrupamentos do Aurora PostgreSQL para EBCDIC e outras migrações de mainframe
- Agrupamentos permitidos no Aurora PostgreSQL.
- Referência de funções do Aurora PostgreSQL
- Amazon Aurora PostgreSQL parameters
- Eventos de espera do Amazon Aurora PostgreSQL

Agrupamentos do Aurora PostgreSQL para EBCDIC e outras migrações de mainframe

A migração de aplicações de mainframe para novas plataformas, como a AWS, preserva de forma ideal o comportamento da aplicação. Preservar o comportamento da aplicação em uma nova plataforma exatamente como era no mainframe requer que os dados migrados sejam agrupados usando as mesmas regras de agrupamento e classificação. Por exemplo, muitas soluções de migração Db2 transferem valores nulos para u0180 (posição Unicode 0180), portanto esses agrupamentos classificam u0180 primeiro. Esse é um exemplo de como os agrupamentos podem variar em relação ao mainframe de origem e por que é necessário escolher um agrupamento que seja melhor mapeado para o agrupamento EBCDIC original.

O Aurora PostgreSQL 14.3 e versões posteriores fornecem muitos agrupamentos ICU e EBCDIC para oferecer suporte a essa migração para a AWS usando o serviço AWS Mainframe Modernization. Para saber mais sobre esse serviço, consulte O que é o AWS Mainframe Modernization?

Na tabela a seguir, você pode encontrar agrupamentos fornecidos pelo Aurora PostgreSQL. Esses agrupamentos seguem as regras do EBCDIC e garantem que as aplicações de mainframe funcionem da mesma forma na AWS como funcionavam no ambiente de mainframe. O nome do agrupamento inclui a página de código relevante (cpnnnn) para que você possa escolher o agrupamento apropriado para sua a origem do seu mainframe. Por exemplo, use `en-US-cp037-x-icu` para obter o comportamento de agrupamento de dados EBCDIC originados de uma aplicação de mainframe que usou a página de código 037.

Agrupamentos EBCDIC	Agrupamentos do AWS Blu Age	Agrupamentos do AWS Micro Focus
da-DK-cp1142-x-icu	da-DK-cp1142b-x-icu	da-DK-cp1142m-x-icu
da-DK-cp277-x-icu	da-DK-cp277b-x-icu	–
de-DE-cp1141-x-icu	de-DE-cp1141b-x-icu	de-DE-cp1141m-x-icu
de-DE-cp273-x-icu	de-DE-cp273b-x-icu	–
en-GB-cp1146-x-icu	en-GB-cp1146b-x-icu	en-GB-cp1146m-x-icu
en-GB-cp285-x-icu	en-GB-cp285b-x-icu	–
en-US-cp037-x-icu	en-US-cp037b-x-icu	–
en-US-cp1140-x-icu	en-US-cp1140b-x-icu	en-US-cp1140m-x-icu
es-ES-cp1145-x-icu	es-ES-cp1145b-x-icu	es-ES-cp1145m-x-icu
es-ES-cp284-x-icu	es-ES-cp284b-x-icu	–
fi-FI-cp1143-x-icu	fi-FI-cp1143b-x-icu	fi-FI-cp1143m-x-icu
fi-FI-cp278-x-icu	fi-FI-cp278b-x-icu	–
fr-FR-cp1147-x-icu	fr-FR-cp1147b-x-icu	fr-FR-cp1147m-x-icu
fr-FR-cp297-x-icu	fr-FR-cp297b-x-icu	–
it-IT-cp1144-x-icu	it-IT-cp1144b-x-icu	it-IT-cp1144m-x-icu
it-IT-cp280-x-icu	it-IT-cp280b-x-icu	–
nl-BE-cp1148-x-icu	nl-BE-cp1148b-x-icu	nl-BE-cp1148m-x-icu
nl-BE-cp500-x-icu	nl-BE-cp500b-x-icu	–

Para saber mais sobre o AWS Blu Age, consulte Tutorial: Tempo de execução gerenciado para AWS Blu Age no Guia do usuário do AWS Mainframe Modernization.

Para obter mais informações sobre como trabalhar com o AWS Micro Focus, consulte Tutorial: Tempo de execução gerenciado para Micro Focus no Guia do usuário do AWS Mainframe Modernization.

Para obter mais informações sobre como gerenciar agrupamentos no PostgreSQL, consulte Collation Support (Compatibilidade com agrupamentos) na documentação do PostgreSQL.

Agrupamentos permitidos no Aurora PostgreSQL.

Agrupamentos são conjuntos de regras que determinam como as strings de caracteres armazenadas no banco de dados são classificadas e comparadas. Eles desempenham um papel fundamental no sistema de computador e são incluídos como parte do sistema operacional. Os agrupamentos mudam com o tempo quando novos caracteres são adicionados aos idiomas ou quando ocorrem alterações nas regras de ordenação.

As bibliotecas de agrupamentos definem regras e algoritmos específicos para um agrupamento. As bibliotecas de agrupamentos mais populares usadas no PostgreSQL são GNU C (glibc) e Componentes Internacionais para Unicode (ICU). Por padrão, o Aurora PostgreSQL usa o agrupamento glibc que inclui ordens de classificação de caracteres Unicode para sequências de caracteres de vários bytes.

Quando você cria um cluster de banco de dados do Aurora PostgreSQL, ele verifica o agrupamento disponível no sistema operacional. Os parâmetros do PostgreSQL `LC_COLLATE` e `LC_CTYPE` do comando `CREATE DATABASE` são usados para especificar um agrupamento, que representa o agrupamento padrão nesse banco de dados. Como alternativa, você também pode usar o parâmetro `LOCALE` em `CREATE DATABASE` para definir esses parâmetros. Isso determina o agrupamento padrão para strings de caracteres no banco de dados e as regras para classificar caracteres como letras, números ou símbolos. Você também pode escolher um agrupamento para usar em uma coluna, um índice ou uma consulta.

O Aurora PostgreSQL depende da biblioteca glibc no sistema operacional para oferecer suporte a agrupamentos. A instância do Aurora PostgreSQL é atualizada periodicamente com as versões mais recentes do sistema operacional. Essas atualizações às vezes incluem uma versão mais recente da biblioteca glibc. Em situações raras, as versões mais recentes da glibc alteram a ordem de classificação ou o agrupamento de alguns caracteres, o que pode fazer com que os dados sejam classificados de forma diferente ou produzam entradas de índice inválidas. Se você descobrir problemas na ordem de classificação para agrupamento durante uma atualização, poderá ser necessário recompilar os índices.

Para reduzir os possíveis impactos das atualizações da glibc, o Aurora PostgreSQL agora inclui uma biblioteca de agrupamentos padrão independente. Essa biblioteca de agrupamentos está disponível no Aurora PostgreSQL 14.6, 13.9, 12.13, 11.18 e versões secundárias mais recentes. É compatível com glibc 2.26-59.amzn2 e oferece estabilidade da ordem de classificação para evitar resultados de consulta incorretos.

Referência de funções do Aurora PostgreSQL

Você pode encontrar uma lista de funções do Aurora PostgreSQL disponíveis para clusters de bancos de dados Aurora que executam o mecanismo de banco de dados Aurora PostgreSQL, edição compatível com PostgreSQL. Essas funções do Aurora PostgreSQL são adicionadas às funções do PostgreSQL padrão. Para obter mais informações sobre funções padrão do PostgreSQL, consulte PostgreSQL – Funções e operadores.

Visão geral

Você pode usar as seguintes funções para instâncias de banco de dados do Amazon RDS que executam o Aurora PostgreSQL:

- aurora_db_instance_identifier
- aurora_ccm_status
- aurora_global_db_instance_status
- aurora_global_db_status
- aurora_list_builtins
- aurora_replica_status
- aurora_stat_activity
- aurora_stat_backend_waits
- aurora_stat_bgwriter
- aurora_stat_database
- aurora_stat_dml_activity
- aurora_stat_get_db_commit_latency
- aurora_stat_logical_wal_cache
- aurora_stat_memctx_usage
- aurora_stat_optimized_reads_cache
- aurora_stat_plans

- aurora_stat_reset_wal_cache
- aurora_stat_statements
- aurora_stat_system_waits
- aurora_stat_wait_event
- aurora_stat_wait_type
- aurora_version
- aurora_volume_logical_start_lsn
- aurora_wait_report

aurora_db_instance_identifier

Informa o nome da instância de banco de dados à qual você está conectado.

Sintaxe

```
aurora_db_instance_identifier()
```

Argumentos

Nenhum

Tipo de retorno

String VARCHAR

Observações de uso

Essa função exibe o nome da instância de banco de dados do cluster da edição do Aurora compatível com PostgreSQL para a conexão do seu cliente ou aplicação de banco de dados.

Essa função está disponível a partir do lançamento do Aurora PostgreSQL versões 13.7, 12.11, 11.16 e 10.21, e para todas as outras versões posteriores.

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_db_instance_identifier`.

```
=> SELECT aurora_db_instance_identifier();
aurora_db_instance_identifier
------------------------------
 test-my-instance-name
```

Você pode unir os resultados dessa função com a função `aurora_replica_status` para obter detalhes sobre a instância de banco de dados para sua conexão. aurora_replica_status sozinho não mostra qual instância de banco de dados você está usando. O exemplo a seguir mostra como.

```
=> SELECT *
    FROM aurora_replica_status() rt,
         aurora_db_instance_identifier() di
    WHERE rt.server_id = di;
-[ RECORD 1 ]----------------------+-----------------------
server_id                         | test-my-instance-name
session_id                        | MASTER_SESSION_ID
durable_lsn                       | 88492069
highest_lsn_rcvd                  |
current_read_lsn                  |
cur_replay_latency_in_usec        |
active_txns                       |
is_current                        | t
last_transport_error              | 0
last_error_timestamp              |
last_update_timestamp             | 2022-06-03 11:18:25+00
feedback_xmin                     |
feedback_epoch                    |
replica_lag_in_msec               |
log_stream_speed_in_kib_per_second | 0
log_buffer_sequence_number        | 0
oldest_read_view_trx_id           |
oldest_read_view_lsn              |
pending_read_ios                  | 819
```

aurora_ccm_status

Exibe o status do gerenciador de cache do cluster.

Sintaxe

```
aurora_ccm_status()
```

Argumentos

Nenhum.

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `buffers_sent_last_minute`: quantos buffers foram enviados ao leitor designado no último minuto.
- `buffers_found_last_minute`: o número de buffers acessados com frequência que foram identificados durante o último minuto.
- `buffers_sent_last_scan`: quantos buffers foram enviados ao leitor designado durante a última verificação completa do cache em buffer.
- `buffers_found_last_scan`: quantos buffers acessados com frequência foram enviados durante a última verificação completa do cache em buffer. Os buffers que já estão em cache no leitor designado não são enviados.
- `buffers_sent_current_scan`: quantos buffers foram enviados durante a verificação atual.
- `buffers_found_current_scan`: o número de buffers acessados com frequência identificados na verificação atual.
- `current_scan_progress`: quantos buffers foram visitados durante a verificação atual até agora.

Observações de uso

É possível usar essa função para verificar e monitorar o recurso de gerenciamento de cache de cluster (CCM). Essa função opera somente quando o CCM está ativo no cluster de banco de dados do Aurora PostgreSQL. Para usar essa função, conecte-se à instância de banco de dados de gravação no cluster de banco de dados do Aurora PostgreSQL.

Ative o CCM para um cluster de banco de dados do Aurora PostgreSQL definindo a opção `apg_ccm_enabled` como 1 no grupo de parâmetros de cluster de banco de dados personalizado. Para saber como, consulte Configuração do gerenciamento de cache do cluster.

O gerenciamento de cache do cluster está ativo em um cluster de banco de dados do Aurora PostgreSQL quando o cluster tem uma instância do leitor do Aurora configurada da seguinte forma:

- A instância do leitor do Aurora usa o mesmo tipo e tamanho de classe que a instância de banco de dados da instância do gravador do cluster.

- A instância do leitor do Aurora está configurada como Nível 0 para o cluster. Se o cluster tiver mais de um leitor, esse será o único leitor Nível 0.

A definição de mais de um leitor como Nível 0 desativa o CCM. Quando o CCM está desativado, a chamada dessa função retornará a seguinte mensagem de erro:

```
ERROR: Cluster Cache Manager is disabled
```

Também é possível usar a extensão pg_buffercache do PostgreSQL para analisar o cache do buffer. Para obter mais informações, consulte pg_buffercache na documentação do PostgreSQL.

Para obter mais informações, consulte Introdução ao gerenciamento de cache do cluster do Aurora PostgreSQL.

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_ccm_status`. Este primeiro exemplo mostra as estatísticas do CCM.

```
=> SELECT * FROM aurora_ccm_status();
 buffers_sent_last_minute | buffers_found_last_minute | buffers_sent_last_scan |
 buffers_found_last_scan | buffers_sent_current_scan | buffers_found_current_scan |
 current_scan_progress
--------------------------+---------------------------+------------------------
+--------------------------+---------------------------+----------------------------
+------------------------
                  2242000 |                   2242003 |               17920442 |
         17923410 |                  14098000 |                   14100964 |
   15877443
```

Para obter detalhes mais completos, use a exibição expandida, conforme mostrado a seguir:

```
\x
Expanded display is on.
SELECT *  FROM aurora_ccm_status();
[ RECORD 1 ]----------------------+---------
buffers_sent_last_minute          | 2242000
buffers_found_last_minute         | 2242003
buffers_sent_last_scan            | 17920442
buffers_found_last_scan           | 17923410
buffers_sent_current_scan         | 14098000
```

```
buffers_found_current_scan         | 14100964
current_scan_progress              | 15877443
```

Este exemplo mostra como verificar a taxa de aquecimento e a porcentagem de aquecimento.

```
=> SELECT buffers_sent_last_minute * 8/60 AS warm_rate_kbps,
100 * (1.0-buffers_sent_last_scan/buffers_found_last_scan) AS warm_percent
FROM aurora_ccm_status ();
 warm_rate_kbps | warm_percent
----------------+--------------
 16523 |        100.0
```

aurora_global_db_instance_status

Exibe o status de todas as instâncias do Aurora, incluindo as réplicas em um cluster de banco de dados global do Aurora.

Sintaxe

```
aurora_global_db_instance_status()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `server_id`: o identificador da instância de banco de dados.
- `session_id`: o identificador exclusivo da sessão. O valor de `MASTER_SESSION_ID` identifica a instância de banco de dados do leitor (primário).
- `aws_region`: a Região da AWS em que essa instância de banco de dados global é executada. Para obter uma lista de regiões, consulte Disponibilidade de regiões.
- `durable_lsn`: o número de sequência de log (LSN) que se tornou durável no armazenamento. Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.

- `highest_lsn_rcvd`: o LSN mais alto recebido pela instância de banco de dados do gravador.
- `feedback_epoch`: a época que a instância de banco de dados usa ao gerar informações de standby a quente. Standby a quente é uma instância de banco de dados que oferece suporte às conexões e às consultas quando o banco de dados primário está no modo de recuperação ou de standby. As informações de standby a quente incluem a época (ponto no tempo) e outros detalhes sobre a instância de banco de dados que está sendo usada como standby a quente. Para obter mais informações, consulte Hot Standby na documentação do PostgreSQL.
- `feedback_xmin`: o ID mínimo (mais antigo) da transação ativa usado pela instância de banco de dados.
- `oldest_read_view_lsn`: o LSN mais antigo usado pela instância de banco de dados para fazer a leitura no armazenamento.
- `visibility_lag_in_msec`: o tempo de atraso dessa instância de banco de dados em relação à instância de banco de dados do gravador, em milissegundos.

Observações de uso

Essa função mostra estatísticas de replicação de um cluster de banco de dados do Aurora. Para cada instância de banco de dados do Aurora PostgreSQL no cluster, a função mostra uma linha de dados que inclui todas as réplicas entre regiões em uma configuração de banco de dados global.

É possível executar essa função em qualquer instância em um cluster de banco de dados do Aurora PostgreSQL ou em um banco de dados global do Aurora PostgreSQL. A função retorna detalhes sobre o atraso de todas as instâncias de réplica.

Para saber mais sobre o monitoramento de atraso usando essa função (`aurora_global_db_instance_status`) ou usando `aurora_global_db_status`, consulte Monitorar banco de dados globais baseados no Aurora PostgreSQL.

Para obter mais informações sobre bancos de dados globais do Aurora, consulte Visão geral do Amazon Aurora Global Database.

Para obter os conceitos básicos sobre bancos de dados globais do Aurora, consulte Conceitos básicos do Amazon Aurora Global Database ou Perguntas frequentes sobre o Amazon Aurora.

Exemplos

Este exemplo mostra as estatísticas de instâncias entre regiões.

```
=> SELECT *
  FROM aurora_global_db_instance_status();
               server_id                |              session_id              |
  aws_region  | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------+--------------------------------------
+-------------+------------+------------------+----------------+---------------
+----------------------+------------------------
 db-119-001-instance-01                  | MASTER_SESSION_ID                    | eu-
west-1    |  2534560273 |           [NULL] |         [NULL] |        [NULL] |
     [NULL] |                [NULL]
 db-119-001-instance-02                  | 4ecff34d-d57c-409c-ba28-278b31d6fc40 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-03                  | 3e8a20fc-be86-43d5-95e5-bdf19d27ad6b | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-04                  | fc1b0023-e8b4-4361-bede-2a7e926cead6 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-instance-05                  | 30319b74-3f08-4e13-9728-e02aa1aa8649 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-global-instance-1            | a331ffbb-d982-49ba-8973-527c96329c60 | eu-
central-1 |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                    996
 db-119-001-global-instance-1            | e0955367-7082-43c4-b4db-70674064a9da | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                     14
 db-119-001-global-instance-1-eu-west-2a  | 1248dc12-d3a4-46f5-a9e2-85850491a897 | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                      0
```

Este exemplo mostra como verificar o atraso de uma réplica global em milissegundos.

```
=> SELECT CASE
          WHEN 'MASTER_SESSION_ID' = session_id THEN 'Primary'
         ELSE 'Secondary'
      END AS global_role,
      aws_region,
      server_id,
      visibility_lag_in_msec
  FROM aurora_global_db_instance_status()
```

```
  ORDER BY 1, 2, 3;
   global_role |  aws_region  |               server_id                |
 visibility_lag_in_msec
-------------+-------------+----------------------------------------
+------------------------
 Primary     | eu-west-1    | db-119-001-instance-01                 |
 [NULL]
 Secondary   | eu-central-1 | db-119-001-global-instance-1           |
     13
 Secondary   | eu-west-1    | db-119-001-instance-02                 |
     10
 Secondary   | eu-west-1    | db-119-001-instance-03                 |
      9
 Secondary   | eu-west-1    | db-119-001-instance-04                 |
      2
 Secondary   | eu-west-1    | db-119-001-instance-05                 |
     18
 Secondary   | eu-west-2    | db-119-001-global-instance-1           |
     14
 Secondary   | eu-west-2    | db-119-001-global-instance-1-eu-west-2a |
     13
```

Este exemplo mostra como verificar o atraso mínimo, máximo e médio por Região da AWS na configuração do banco de dados global.

```
=> SELECT 'Secondary' global_role,
       aws_region,
       min(visibility_lag_in_msec) min_lag_in_msec,
       max(visibility_lag_in_msec) max_lag_in_msec,
       round(avg(visibility_lag_in_msec),0) avg_lag_in_msec
  FROM aurora_global_db_instance_status()
 WHERE aws_region NOT IN (SELECT   aws_region
                            FROM aurora_global_db_instance_status()
                           WHERE session_id='MASTER_SESSION_ID')
                        GROUP BY aws_region
UNION ALL
SELECT  'Primary' global_role,
        aws_region,
        NULL,
        NULL,
        NULL
    FROM aurora_global_db_instance_status()
   WHERE session_id='MASTER_SESSION_ID'
```

```
ORDER BY 1, 5;
 global_role |  aws_region  | min_lag_in_msec | max_lag_in_msec | avg_lag_in_msec
-------------+--------------+-----------------+-----------------+-----------------
 Primary     | eu-west-1    |          [NULL] |          [NULL] |          [NULL]
 Secondary   | eu-central-1 |             133 |             133 |             133
 Secondary   | eu-west-2    |               0 |             495 |             248
```

aurora_global_db_status

Exibe informações sobre diversos aspectos do atraso do banco de dados Aurora global, especificamente o atraso do armazenamento do Aurora subjacente (o chamado atraso de durabilidade) e o atraso entre o objetivo de ponto de recuperação (RPO).

Sintaxe

```
aurora_global_db_status()
```

Argumentos

Nenhum.

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `aws_region`: a Região da AWS em que esse cluster de banco de dados está. Para obter uma lista completa de Regiões da AWS por mecanismo, consulte Regiões e zonas de disponibilidade.
- `highest_lsn_written`: o número de sequência de logs (LSN) mais alto nesse cluster de banco de dados no momento. Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.
- `durability_lag_in_msec`: a diferença nos valores de carimbo de data/hora entre `highest_lsn_written` em um cluster de banco de dados secundário e `highest_lsn_written` no cluster de banco de dados primário. Um valor de -1 identifica o cluster de banco de dados primário do banco de dados Aurora global.
- `rpo_lag_in_msec`: o atraso do objetivo de ponto de recuperação (RPO). O atraso do RPO é o tempo necessário para que a transação COMMIT mais recente do usuário seja armazenada em um cluster de banco de dados secundário após ser armazenada no cluster de banco de dados

primário de um banco de dados Aurora global. Um valor igual a -1 denota o cluster de banco de dados primário (e, portanto, o atraso não é relevante).

Em termos simples, essa métrica calcula o objetivo do ponto de recuperação de cada cluster de banco de dados do Aurora PostgreSQL no banco de dados Aurora global, ou seja, quantos dados poderão ser perdidos se houver uma interrupção. Tal como o atraso, o RPO é medido em tempo.

- `last_lag_calculation_time`: o carimbo de data/hora que especifica quando os valores foram calculados pela última vez para `durability_lag_in_msec` e `rpo_lag_in_msec`. Um valor de tempo como `1970-01-01 00:00:00+00` indica que este é o cluster de banco de dados primário.
- `feedback_epoch`: a época que o cluster de banco de dados secundário usa ao gerar informações de standby a quente. Standby a quente é uma instância de banco de dados que oferece suporte às conexões e às consultas quando o banco de dados primário está no modo de recuperação ou de standby. As informações de standby a quente incluem a época (ponto no tempo) e outros detalhes sobre a instância de banco de dados que está sendo usada como standby a quente. Para obter mais informações, consulte Hot Standby na documentação do PostgreSQL.
- `feedback_xmin`: o ID mínimo (mais antigo) da transação ativa usado por um cluster de banco de dados secundário.

Observações de uso

Todas as versões do Aurora PostgreSQL atualmente disponíveis são compatíveis com essa função. Esta função mostra estatísticas de replicação de um banco de dados do Aurora. A função mostra uma linha para cada cluster de banco de dados em um banco de dados global do Aurora PostgreSQL. É possível executar essa função em qualquer instância em um banco de dados global do Aurora PostgreSQL.

Para avaliar o atraso de replicação de banco de dados global do Aurora, que é o atraso de dados visível, consulte aurora_global_db_instance_status.

Para saber mais sobre o uso de `aurora_global_db_status` e de `aurora_global_db_instance_status` para monitorar o atraso do banco de dados global do Aurora, consulte Monitorar banco de dados globais baseados no Aurora PostgreSQL. Para obter mais informações sobre bancos de dados globais do Aurora, consulte Visão geral do Amazon Aurora Global Database.

Exemplos

Este exemplo mostra como exibir estatísticas de armazenamento entre regiões.

```
=> SELECT CASE
          WHEN '-1' = durability_lag_in_msec THEN 'Primary'
          ELSE 'Secondary'
       END AS global_role,
       *
  FROM aurora_global_db_status();
 global_role | aws_region | highest_lsn_written | durability_lag_in_msec |
 rpo_lag_in_msec | last_lag_calculation_time  | feedback_epoch | feedback_xmin
-------------+------------+---------------------+------------------------
+-----------------+----------------------------+----------------+--------------
 Primary     | eu-west-1  |           131031557 |                     -1 |
 -1 | 1970-01-01 00:00:00+00     |             0 |              0
 Secondary   | eu-west-2  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
 Secondary   | eu-west-3  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
```

aurora_list_builtins

Lista todas as funções integradas do Aurora PostgreSQL disponíveis, juntamente com breves descrições e detalhes da função.

Sintaxe

```
aurora_list_builtins()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_list_builtins`.

```
=> SELECT *
```

```
FROM aurora_list_builtins();

               Name                | Result data type |                    Argument data
 types                     | Type | Volatility |  Parallel  | Security |
             Description
-----------------------------------+------------------
+-------------------------------------------------------------+------+------------
+------------+----------
+-------------------------------------------------------------------------
 aurora_version                    | text             |
                           | func | stable     | safe       | invoker  | Amazon Aurora
 PostgreSQL-Compatible Edition version string
 aurora_stat_wait_type             | SETOF record     | OUT type_id smallint, OUT
 type_name text                 | func | volatile   | restricted | invoker  | Lists all
 supported wait types
 aurora_stat_wait_event            | SETOF record     | OUT type_id smallint, OUT
 event_id integer, OUT event_na.| func | volatile   | restricted | invoker  | Lists all
 supported wait events
                                   |                  |.me text
                          |      |            |            |          |
 aurora_list_builtins              | SETOF record     | OUT "Name" text, OUT "Result
 data type" text, OUT "Argum.| func | stable     | safe       | invoker  | Lists all
 Aurora built-in functions
                                   |                  |.ent data types" text, OUT
 "Type" text, OUT "Volatility" .|      |            |            |          |
                                   |                  |.text, OUT "Parallel" text, OUT
 "Security" text, OUT "Des.|      |            |            |          |
                                   |                  |.cription" text
                          |      |            |            |          |
 .
 .
 .
 aurora_stat_file                  | SETOF record     | OUT filename text, OUT
 allocated_bytes bigint, OUT used_.| func | stable     | safe       | invoker  | Lists
 all files present in Aurora storage
                                   |                  |.bytes bigint
                          |      |            |            |          |
 aurora_stat_get_db_commit_latency | bigint           | oid
                           | func | stable     | restricted | invoker  | Per DB commit
 latency in microsecs
```

aurora_replica_status

Exibe o status de todos os nós de leitor do Aurora PostgreSQL.

Sintaxe

```
aurora_replica_status()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `server_id`: o DB de instância de banco de dados (identificador).
- `session_id`: um identificador exclusivo da sessão atual, retornado para instâncias primárias e do leitor da seguinte forma:
  - Para a instância principal, `session_id` é sempre `` `MASTER_SESSION_ID' ``.
  - Para instâncias de leitor, `session_id` é sempre o `UUID` (identificador universalmente exclusivo) da instância do leitor.
- `durable_lsn`: o número de sequência de log (LSN) que foi salvo no armazenamento.
  - Para o volume principal, o LSN (VDL) durável do volume principal que está atualmente em vigor.
  - Para quaisquer volumes secundários, o VDL do principal até o secundário que foi aplicado com êxito.

> **Note**
>
> Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. Os LSNs são ordenados de forma que um LSN maior represente uma transação ocorrida posteriormente na sequência.

- `highest_lsn_rcvd`: o LSN mais alto (mais recente) recebido pela instância de banco de dados da instância de banco de dados do gravador.
- `current_read_lsn`: o LSN do snapshot mais recente que foi aplicado a todos os leitores.

- `cur_replay_latency_in_usec`: o número de microssegundos que se espera para reproduzir o log no secundário.
- `active_txns`: o número de transações atualmente ativas.
- `is_current`: não usado.
- `last_transport_error`: código do último erro de replicação.
- `last_error_timestamp`: carimbo de data/hora do último erro de replicação.
- `last_update_timestamp`: carimbo de data/hora da última atualização para o status da réplica. No Aurora PostgreSQL 13.9, o valor `last_update_timestamp` da instância de banco de dados com que você tem conexão é definido como `NULL`.
- `feedback_xmin`: o feedback_xmin de standby a quente da réplica. O ID mínimo (mais antigo) de transação ativo usado pela instância de banco de dados.
- `feedback_epoch`: a época que a instância de banco de dados usa quando gera informações de standby a quente.
- `replica_lag_in_msec`: o tempo de atraso da instância do leitor em relação à instância do gravador, em milissegundos.
- `log_stream_speed_in_kib_per_second`: a taxa de transferência do fluxo de logs em kilobytes por segundo.
- `log_buffer_sequence_number`: o número da sequência do buffer de log.
- `oldest_read_view_trx_id`: não usado.
- `oldest_read_view_lsn`: o LSN mais antigo usado pela instância de banco de dados para fazer a leitura no armazenamento.
- `pending_read_ios`: a página pendente indica que ainda há pendência na réplica.
- `read_ios`: o número total de leituras de página na réplica.
- `iops`: não usado.
- `cpu`: uso da CPU do Aurora Storage Daemon para cada nó no cluster. Para obter mais informações sobre o uso da CPU pela instância, consulte Métricas no nível da instância do Amazon Aurora.

Observações de uso

Todas as versões do Aurora PostgreSQL atualmente disponíveis são compatíveis com essa função. A função `aurora_replica_status` retorna valores do gerenciador de status de réplica de um cluster de banco de dados do Aurora PostgreSQL. Você pode usar essa função para obter

informações sobre o status da replicação em seu cluster de banco de dados do Aurora PostgreSQL, como métricas para todas as instâncias de banco de dados no cluster de banco de dados do Aurora. Por exemplo, você pode fazer o seguinte:

- Obtenha informações sobre o tipo de instância (gravador, leitor) no cluster de banco de dados do Aurora PostgreSQL: você pode obter essas informações conferindo os valores das seguintes colunas:
  - `server_id`: contém o nome da instância que você especificou quando criou a instância. Em alguns casos, como para a instância principal (gravador), o nome geralmente é criado para você anexando -instância-1 ao nome criado para o cluster de banco de dados do Aurora PostgreSQL.
  - `session_id`: o campo `session_id` indica se a instância é um leitor ou um gravador. Para uma instância de gravador, `session_id` é sempre definido como `"MASTER_SESSION_ID"`. Para uma instância de leitor, `session_id` é definido como o `UUID` do leitor específico.
- Diagnosticar problemas comuns de replicação, como atraso de réplica: o atraso da réplica é o tempo em milissegundos em que o cache de página de uma instância de leitor está atrasado em relação à instância de gravador. Esse atraso ocorre porque os clusters do Aurora usam replicação assíncrona, conforme descrito em Replicação com o Amazon Aurora. Ele é mostrado na coluna `replica_lag_in_msec`, nos resultados retornados por essa função. O atraso também pode ocorrer quando uma consulta é cancelada devido a conflitos com a recuperação em um servidor em espera. Você pode conferir `pg_stat_database_conflicts()` para verificar se esse conflito está causando o atraso da réplica (ou não). Para obter mais informações, consulte Coletor de estatísticas na documentação do PostgreSQL. Para saber mais sobre alta disponibilidade e replicação, consulte Perguntas frequentes sobre o Amazon Aurora.

  O Amazon CloudWatch armazena resultados `replica_lag_in_msec` ao longo do tempo, como a métrica do `AuroraReplicaLag`. Para obter mais informações sobre o uso de métricas do CloudWatch para o Aurora, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.

Para saber mais sobre a solução de problemas de réplicas de leitura e reinicializações do Aurora, consulte Por que minha réplica de leitura do Amazon Aurora ficou atrasada e foi reiniciada? no AWS Support Center.

Exemplos

O exemplo a seguir mostra como obter o status de replicação de todas as instâncias em um cluster de banco de dados do Aurora PostgreSQL:

```
=> SELECT *
FROM aurora_replica_status();
```

O exemplo a seguir mostra a instância do gravador no cluster de banco de dados `docs-lab-apg-main` do Aurora PostgreSQL:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id = 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
 db-119-001-instance-01 | writer
```

O exemplo a seguir lista todas as instâncias de leitor em um cluster:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id <> 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
db-119-001-instance-02  | reader
db-119-001-instance-03  | reader
db-119-001-instance-04  | reader
db-119-001-instance-05  | reader
(4 rows)
```

O exemplo a seguir lista todas as instâncias, o atraso de cada instância em relação ao gravador e o tempo desde a última atualização:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
```

```
    END AS instance_role,
    replica_lag_in_msec AS replica_lag_ms,
    round(extract (epoch FROM (SELECT age(clock_timestamp(), last_update_timestamp))) *
 1000) AS last_update_age_ms
FROM aurora_replica_status()
ORDER BY replica_lag_in_msec NULLS FIRST;
       server_id        | instance_role | replica_lag_ms | last_update_age_ms
------------------------+---------------+----------------+--------------------
 db-124-001-instance-03 | writer        |         [NULL] |               1756
 db-124-001-instance-01 | reader        |             13 |               1756
 db-124-001-instance-02 | reader        |             13 |               1756
(3 rows)
```

aurora_stat_activity

Exibe uma linha por processo do servidor, mostrando informações relacionadas à atividade atual desse processo.

Sintaxe

```
aurora_stat_activity();
```

Argumentos

Nenhum

Tipo de retorno

Exibe uma linha por processo do servidor. Além das colunas `pg_stat_activity`, o seguinte campo é adicionado:

- planid: identificador do plano

Observações de uso

Uma visualização complementar para `pg_stat_activity` exibindo as mesmas colunas com uma coluna `plan_id` adicional que mostra o plano de execução da consulta atual.

`aurora_compute_plan_id` deve estar habilitado para que a visualização exiba um plan_id.

Essa função está disponível para o Aurora PostgreSQL versões 14.10, 15.5 e todas as outras versões posteriores.

Exemplos

O exemplo de consulta abaixo agrega a carga superior por query_id e plan_id.

```
db1=# select count(*), query_id, plan_id
db1-# from aurora_stat_activity() where state = 'active'
db1-# and pid <> pg_backend_pid()
db1-# group by query_id, plan_id
db1-# order by 1 desc;

count |  query_id             |  plan_id
-------+-----------------------+-------------
 11    | -5471422286312252535 | -2054628807
 3     | -6907107586630739258 | -815866029
 1     | 5213711845501580017  |  300482084
(3 rows)
```

Se o plano usado para query_id mudar, um novo plan_id será relatado por aurora_stat_activity.

```
count  |  query_id             |  plan_id
-------+-----------------------+-------------
 10    | -5471422286312252535 | 1602979607
 1     | -6907107586630739258 | -1809935983
 1     | -2446282393000597155 | -207532066
(3 rows)
```

aurora_stat_backend_waits

Exibe estatísticas da atividade de espera de um processo de backend específico.

Sintaxe

```
aurora_stat_backend_waits(pid)
```

Argumentos

`pid`: o ID do processo de backend. É possível obter os IDs de processos usando a visualização `pg_stat_activity`.

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `type_id`: um número que denota o tipo de evento de espera, como 1 para um bloqueio leve (`LWLock`), 3 para um bloqueio ou 6 para uma sessão de cliente, para citar alguns exemplos. Esses valores se tornam significativos ao unir os resultados dessa função com colunas da função `aurora_stat_wait_type`, como mostrado em Exemplos.
- `event_id`: um número de identificação para o evento de espera. Una esse valor com as colunas de `aurora_stat_wait_event` para obter nomes de eventos significativos.
- `waits`: uma contagem do número de esperas acumuladas para o ID do processo especificado.
- `wait_time`: tempo de espera em milissegundos.

Observações de uso

É possível usar essa função para analisar eventos de espera específicos de backend (sessão) que ocorreram desde que uma conexão foi aberta. Para obter informações mais significativas sobre os nomes e os tipos de evento de espera, combine as funções `aurora_stat_wait_type` e `aurora_stat_wait_event`, usando JOIN como mostrado nos exemplos.

Exemplos

Este exemplo mostra todas as esperas, os tipos e os nomes dos eventos de um processo de backend ID 3027.

```
=> SELECT type_name, event_name, waits, wait_time
        FROM aurora_stat_backend_waits(3027)
NATURAL JOIN aurora_stat_wait_type()
NATURAL JOIN aurora_stat_wait_event();
type_name |       event_name       | waits | wait_time
----------+------------------------+-------+-----------
 LWLock    | ProcArrayLock          |     3 |         27
 LWLock    | wal_insert             |   423 |      16336
 LWLock    | buffer_content         | 11840 |    1033634
 LWLock    | lock_manager           | 23821 |    5664506
```

```
 Lock      | tuple                 | 10258 |  152280165
 Lock      | transactionid         | 78340 | 1239808783
 Client    | ClientRead            | 34072 |   17616684
 IO        | ControlFileSyncUpdate |     2 |          0
 IO        | ControlFileWriteUpdate|     4 |         32
 IO        | RelationMapRead       |     2 |        795
 IO        | WALWrite              | 36666 |      98623
 IO        | XactSync              |  4867 |    7331963
```

Este exemplo mostra os tipos de espera atuais e cumulativos e de eventos de espera para todas as sessões ativas (`pg_stat_activity state <> 'idle'`), mas sem a sessão atual que está invocando a função (`pid <> pg_backend_pid()`).

```
=> SELECT a.pid,
            a.usename,
            a.app_name,
            a.current_wait_type,
            a.current_wait_event,
            a.current_state,
            wt.type_name AS wait_type,
            we.event_name AS wait_event,
            a.waits,
            a.wait_time
       FROM (SELECT pid,
                    usename,
                    left(application_name,16) AS app_name,
                    coalesce(wait_event_type,'CPU') AS current_wait_type,
                    coalesce(wait_event,'CPU') AS current_wait_event,
                    state AS current_state,
                    (aurora_stat_backend_waits(pid)).*
               FROM pg_stat_activity
              WHERE pid <> pg_backend_pid()
                AND state <> 'idle') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |       wait_event       | waits | wait_time
-------+----------+----------+-------------------+--------------------+---------------
+-----------+------------------------+-------+-----------
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1937 |     29975
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22903 |    760498
```

```
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | lock_manager           | 10012 |    223207
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
Lock      | tuple                  | 20315 |  63081529
.
.
.
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | WALWrite               | 93293 |    237440
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | XactSync               | 13010 |  19525143
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | ProcArrayLock          |     6 |        53
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | wal_insert             |  1913 |     25450
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | buffer_content         | 22874 |    778005
.
.
.
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | ProcArrayLock          |     3 |        71
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | wal_insert             |  1940 |     27741
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | buffer_content         | 22962 |    776352
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | lock_manager           |  9879 |    218826
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | tuple                  | 20401 |  63581306
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | transactionid          | 50769 | 211645008
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Client    | ClientRead             | 89901 |  44192439
```

Este exemplo mostra os três tipos principais de espera cumulativa e os eventos de espera de todas as sessões ativas (`pg_stat_activity state <> 'idle'`), exceto a sessão atual (`pid <>pg_backend_pid()`).

```
=> SELECT top3.*
       FROM (SELECT a.pid,
                    a.usename,
                    a.app_name,
```

```
                    a.current_wait_type,
                    a.current_wait_event,
                    a.current_state,
                    wt.type_name AS wait_type,
                    we.event_name AS wait_event,
                    a.waits,
                    a.wait_time,
                    RANK() OVER (PARTITION BY pid ORDER BY a.wait_time DESC)
               FROM (SELECT pid,
                            usename,
                            left(application_name,16) AS app_name,
                            coalesce(wait_event_type,'CPU') AS current_wait_type,
                            coalesce(wait_event,'CPU') AS current_wait_event,
                            state AS current_state,
                            (aurora_stat_backend_waits(pid)).*
                       FROM pg_stat_activity
                      WHERE pid <> pg_backend_pid()
                        AND state <> 'idle') a
             NATURAL JOIN aurora_stat_wait_type() wt
             NATURAL JOIN aurora_stat_wait_event() we) top3
 WHERE RANK <=3;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |   wait_event    |  waits  | wait_time  | rank
-------+----------+----------+-------------------+--------------------+---------------
+-----------+-----------------+---------+------------+------
 20567 | postgres | psql     | CPU               | CPU                | active        |
 LWLock    | wal_insert      |   25000 |   67512003 |    1
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | WALWrite        | 3071758 |    1016961 |    2
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | BufFileWrite    |   20750 |     184559 |    3
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237350 | 1265580011 |    1
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93641 |  341472318 |    2
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Client    | ClientRead      |  417556 |  204796837 |    3
 .
 .
 .
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | transactionid   |  238068 | 1265816822 |    1
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | tuple           |   93210 |  338312247 |    2
```

```
 27745 | postgres | pgbench  | IO               | XactSync          | active       |
 Client    | ClientRead      |  419157 |  207836533 |    3
 27746 | postgres | pgbench  | Lock             | transactionid     | active       |
 Lock      | transactionid   |  237621 | 1264528811 |    1
 27746 | postgres | pgbench  | Lock             | transactionid     | active       |
 Lock      | tuple           |   93563 |  339799310 |    2
 27746 | postgres | pgbench  | Lock             | transactionid     | active       |
 Client    | ClientRead      |  417304 |  208372727 |    3
```

aurora_stat_bgwriter

`aurora_stat_bgwriter` é uma exibição de estatísticas que mostra informações sobre gravações em cache do Optimized Reads.

Sintaxe

```
aurora_stat_bgwriter()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com todas as colunas `pg_stat_bgwriter` e as seguintes colunas adicionais. Para obter mais informações sobre as colunas `pg_stat_bgwriter`, consulte `pg_stat_bgwriter`.

Você pode redefinir as estatísticas dessa função usando `pg_stat_reset_shared("bgwriter")`.

- `orcache_blks_written`: número total de gravações em cache de blocos de dados do optimized reads.
- `orcache_blk_write_time`: se `track_io_timing` estiver ativado, ele rastreará o tempo total gasto gravando blocos de arquivos de dados em cache do optimized reads, em milissegundos. Para obter mais informações, consulte track_io_timing.

Observações de uso

Essa função está disponível nas seguintes versões do Aurora PostgreSQL:

- 15.4 e versões 15 posteriores

- 14.9 e versões 14 posteriores

Exemplos

```
=> select * from aurora_stat_bgwriter();
-[ RECORD 1 ]-----------------+-----------
orcache_blks_written          | 246522
orcache_blk_write_time        | 339276.404
```

aurora_stat_database

Ele carrega todas as colunas de pg_stat_database e acrescenta novas colunas no final.

Sintaxe

```
aurora_stat_database()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com todas as colunas `pg_stat_database` e as seguintes colunas adicionais. Para obter mais informações sobre as colunas `pg_stat_database`, consulte `pg_stat_database`.

- `storage_blks_read`: número total de blocos compartilhados lidos do armazenamento do aurora nesse banco de dados.
- `orcache_blks_hit`: número total de acessos ao cache do optimized reads nesse banco de dados.
- `local_blks_read`: número total de blocos locais lidos nesse banco de dados.
- `storage_blk_read_time`: se `track_io_timing` estiver ativado, ele rastreará o tempo total gasto lendo blocos de arquivos de dados do armazenamento do aurora, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.
- `local_blk_read_time`: se `track_io_timing` estiver ativado, ele rastreará o tempo total gasto lendo blocos de arquivos de dados locais, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.

- `orcache_blk_read_time`: se `track_io_timing` estiver ativado, ele rastreará o tempo total gasto lendo blocos de arquivos de dados do cache do optimized reads, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.

> **Note**
>
> O valor de `blks_read` é a soma de `storage_blks_read`, `orcache_blks_hit` e `local_blks_read`.
> O valor de `blk_read_time` é a soma de `storage_blk_read_time`, `orcache_blk_read_time` e `local_blk_read_time`.

Observações de uso

Essa função está disponível nas seguintes versões do Aurora PostgreSQL:

- 15.4 e versões 15 posteriores
- 14.9 e versões 14 posteriores

Exemplos

O exemplo a seguir mostra como ele carrega todas as colunas `pg_stat_database` e acrescenta seis novas colunas no final:

```
=> select * from aurora_stat_database() where datid=14717;
-[ RECORD 1 ]------------+------------------------------
datid                    | 14717
datname                  | postgres
numbackends              | 1
xact_commit              | 223
xact_rollback            | 4
blks_read                | 1059
blks_hit                 | 11456
tup_returned             | 27746
tup_fetched              | 5220
tup_inserted             | 165
tup_updated              | 42
tup_deleted              | 91
conflicts                | 0
```

```
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 3358.689
blk_write_time           | 0
session_time             | 1076007.997
active_time              | 3684.371
idle_in_transaction_time | 0
sessions                 | 10
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 0
stats_reset              | 2023-01-12 20:15:17.370601+00
orcache_blks_hit         | 425
orcache_blk_read_time    | 89.934
storage_blks_read        | 623
storage_blk_read_time    | 3254.914
local_blks_read          | 0
local_blk_read_time      | 0
```

aurora_stat_dml_activity

Relata a atividade cumulativa para cada tipo de operação de linguagem de manipulação de dados (DML) em um banco de dados em um cluster do Aurora PostgreSQL.

Sintaxe

```
aurora_stat_dml_activity(database_oid)
```

Argumentos

database_oid

O ID do objeto (OID) do banco de dados no cluster do Aurora PostgreSQL.

Tipo de retorno

Registro SETOF

Observações de uso

A função `aurora_stat_dml_activity` só está disponível com o Aurora PostgreSQL versão 3.1 compatível com o mecanismo do PostgreSQL 11.6 e posteriores.

Use essa função em clusters do Aurora PostgreSQL com um grande número de bancos de dados para identificar quais bancos de dados apresentam mais atividades de DML ou atividades de DML mais lentas ou ambas.

A função `aurora_stat_dml_activity` retorna o número de vezes que as operações foram executadas e a latência cumulativa em microssegundos para as operações SELECT, INSERT, UPDATE e DELETE. O relatório inclui apenas operações DML bem-sucedidas.

Você pode redefinir essa estatística usando a função de acesso de estatísticas do PostgreSQL `pg_stat_reset`. Você pode verificar a última vez que esta estatística foi redefinida usando a função `pg_stat_get_db_stat_reset_time`. Para obter mais informações sobre as funções de acesso a estatísticas do PostgreSQL, consulte Coletor de estatísticas na documentação do PostgreSQL.

Exemplos

O exemplo a seguir mostra como relatar estatísticas de atividades de DML para o banco de dados conectado.

```
--Define the oid variable from connected database by using \gset
=> SELECT oid,
          datname
     FROM pg_database
    WHERE datname=(select current_database()) \gset
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
select_count | select_latency_microsecs | insert_count | insert_latency_microsecs |
 update_count | update_latency_microsecs | delete_count | delete_latency_microsecs
--------------+--------------------------+--------------+--------------------------
+--------------+--------------------------+--------------+--------------------------
       178957 |                 66684115 |       171065 |                 28876649 |
    519538 |           1454579206167 |            1 |                    53027


-- Showing the same results with expanded display on
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
```

```
-[ RECORD 1 ]------------+--------------
select_count             | 178957
select_latency_microsecs | 66684115
insert_count             | 171065
insert_latency_microsecs | 28876649
update_count             | 519538
update_latency_microsecs | 1454579206167
delete_count             | 1
delete_latency_microsecs | 53027
```

O exemplo a seguir mostra estatísticas de atividades de DML para todos os bancos de dados no cluster do Aurora PostgreSQL. Esse cluster tem dois bancos de dados,`postgres` e `mydb`. A lista separada por vírgulas corresponde aos campos `select_count`, `select_latency_microsecs`, `insert_count`, `insert_latency_microsecs`, `update_count`, `update_latency_microsecs`, `delete_count` e `delete_latency_microsecs`.

O Aurora PostgreSQL cria e usa um banco de dados do sistema chamado `rdsadmin` para oferecer suporte a operações administrativas, como backups, restaurações, verificações de integridade, replicação e assim por diante. Essas operações de DML não têm impacto no cluster do Aurora PostgreSQL.

```
=> SELECT oid,
    datname,
    aurora_stat_dml_activity(oid)
    FROM pg_database;
oid  |    datname     |                   aurora_stat_dml_activity
-------+----------------
+-----------------------------------------------------------------
 14006 | template0      | (,,,,,,,)
 16384 | rdsadmin       | (2346623,1211703821,4297518,817184554,0,0,0,0)
     1 | template1      | (,,,,,,,)
 14007 | postgres       |
 (178961,66716329,171065,28876649,519538,1454579206167,1,53027)
 16401 | mydb           | (200246,64302436,200036,107101855,600000,83659417514,0,0)
```

O exemplo a seguir mostra estatísticas de atividade DML para todos os bancos de dados, organizadas em colunas para melhor legibilidade.

```
SELECT db.datname,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 1), '()') AS select_count,
```

```
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 2), '()') AS select_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 3), '()') AS insert_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 4), '()') AS insert_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 5), '()') AS update_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 6), '()') AS update_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 7), '()') AS delete_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 8), '()') AS delete_latency_microsecs
FROM  (SELECT datname,
              aurora_stat_dml_activity(oid) AS asdmla
         FROM pg_database
      ) AS db;

    datname     | select_count | select_latency_microsecs | insert_count |
 insert_latency_microsecs | update_count | update_latency_microsecs | delete_count |
 delete_latency_microsecs
---------------+--------------+--------------------------+--------------
+--------------------------+--------------+--------------------------+--------------
+--------------------------
 template0      |              |                          |              |
            |              |                          |              |
 rdsadmin       | 4206523      | 2478812333               | 7009414      | 1338482258
            | 0            | 0                        | 0            | 0
 template1      |              |                          |              |
            |              |                          |              |
 fault_test     | 66           | 452099                   | 0            | 0
            | 0            | 0                        | 0            | 0
 db_access_test | 1            | 5982                     | 0            | 0
            | 0            | 0                        | 0            | 0
 postgres       | 42035        | 95179203                 | 5752         | 2678832898
            | 21157        | 441883182488             | 2            | 1520
 mydb           | 71           | 453514                   | 0            | 0
            | 1            | 190                      | 1            | 152
```

O exemplo a seguir mostra a latência cumulativa média (latência cumulativa dividida por contagem) para cada operação de DML para o banco de dados com o OID 16401.

```
=> SELECT select_count,
          select_latency_microsecs,
          select_latency_microsecs/NULLIF(select_count,0) select_latency_per_exec,
          insert_count,
          insert_latency_microsecs,
          insert_latency_microsecs/NULLIF(insert_count,0) insert_latency_per_exec,
          update_count,
```

```
          update_latency_microsecs,
          update_latency_microsecs/NULLIF(update_count,0) update_latency_per_exec,
          delete_count,
          delete_latency_microsecs,
          delete_latency_microsecs/NULLIF(delete_count,0) delete_latency_per_exec
     FROM aurora_stat_dml_activity(16401);
-[ RECORD 1 ]------------+-------------
select_count             | 451312
select_latency_microsecs | 80205857
select_latency_per_exec  | 177
insert_count             | 451001
insert_latency_microsecs | 123667646
insert_latency_per_exec  | 274
update_count             | 1353067
update_latency_microsecs | 200900695615
update_latency_per_exec  | 148478
delete_count             | 12
delete_latency_microsecs | 448
delete_latency_per_exec  | 37
```

aurora_stat_get_db_commit_latency

Obtém a latência de confirmação cumulativa em microssegundos para bancos de dados do Aurora PostgreSQL. A confirmação da latência é medida como o tempo entre quando um cliente envia uma solicitação de confirmação e quando ele recebe a resposta afirmativa da confirmação.

Sintaxe

```
aurora_stat_get_db_commit_latency(database_oid)
```

Argumentos

database_oid

O ID do objeto (OID) do banco de dados Aurora PostgreSQL.

Tipo de retorno

Registro SETOF

Observações de uso

O Amazon CloudWatch usa essa função para calcular a latência média de confirmação. Para obter mais informações sobre as métricas do Amazon CloudWatch e como solucionar problemas de alta latência de confirmação, consulte Visualizar métricas no console do Amazon RDS e Como tomar melhores decisões sobre o Amazon RDS com métricas do Amazon CloudWatch.

Você pode redefinir essa estatística usando a função de acesso de estatísticas do PostgreSQL `pg_stat_reset`. Você pode verificar a última vez que esta estatística foi redefinida usando a função `pg_stat_get_db_stat_reset_time`. Para obter mais informações sobre as funções de acesso a estatísticas do PostgreSQL, consulte Coletor de estatísticas na documentação do PostgreSQL.

Exemplos

O exemplo a seguir obtém a latência de confirmação cumulativa para cada banco de dados no cluster `pg_database`.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database;

  oid  |    datname     | aurora_stat_get_db_commit_latency
-------+----------------+-----------------------------------
 14006 | template0      |                                 0
 16384 | rdsadmin       |                         654387789
     1 | template1      |                                 0
 16401 | mydb           |                            229556
 69768 | postgres       |                             22011
```

O exemplo a seguir obtém a latência de confirmação cumulativa para o banco de dados conectado no momento. Antes de chamar a função `aurora_stat_get_db_commit_latency`, o exemplo primeiro usa `\gset` para definir uma variável para o argumento `oid` e define seu valor do banco de dados conectado.

```
--Get the oid value from the connected database before calling
 aurora_stat_get_db_commit_latency
=> SELECT oid
     FROM pg_database
```

```
    WHERE datname=(SELECT current_database()) \gset
=> SELECT *
     FROM aurora_stat_get_db_commit_latency(:oid);

 aurora_stat_get_db_commit_latency
-----------------------------------
                        1424279160
```

O exemplo a seguir obtém a latência de confirmação cumulativa para o banco de dados mydb no cluster pg_database. Em seguida, ele redefine essa estatística usando a função pg_stat_reset e mostra os resultados. Finalmente, ele usa a função pg_stat_get_db_stat_reset_time para verificar a última vez que essa estatística foi redefinida.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database
    WHERE datname = 'mydb';

  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                           3320370


=> SELECT pg_stat_reset();
 pg_stat_reset
---------------


=> SELECT oid,
          datname,
          aurora_stat_get_db_commit_latency(oid)
     FROM pg_database
    WHERE datname = 'mydb';
  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                                 6


=> SELECT *
     FROM pg_stat_get_db_stat_reset_time(16427);

 pg_stat_get_db_stat_reset_time
```

```
-------------------------------
 2021-04-29 21:36:15.707399+00
```

aurora_stat_logical_wal_cache

Mostra o uso do log de gravação antecipada (WAL) por slot.

Sintaxe

```
SELECT * FROM aurora_stat_logical_wal_cache()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `name`: o nome do slot de replicação.
- `active_pid`: ID do processo walsender.
- `cache_hit`: o número total de acessos ao cache wal desde a última redefinição.
- `cache_miss`: o número total de perdas do cache wal desde a última redefinição.
- `blks_read`: o número total de solicitações de leitura do cache wal.
- `hit_rate`: a taxa de acertos do cache WAL (cache_hit/blks_read).
- `last_reset_timestamp`: última vez que o contador foi redefinido.

Observações de uso

Essa função está disponível para as versões a seguir do Aurora PostgreSQL.

- 15.2 e todas as versões posteriores.
- 14.7 e versões posteriores.
- 13.08 e versões posteriores
- 12.12 e versões posteriores

- 11.17 e versões posteriores

Exemplos

O exemplo a seguir mostra dois slots de replicação com somente uma função `aurora_stat_logical_wal_cache` ativa.

```
=> SELECT *
     FROM aurora_stat_logical_wal_cache();
    name    | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
------------+------------+-----------+------------+-----------+----------
+-------------------------------
 test_slot1 |      79183 |        24 |          0 |        24 | 100.00%  | 2022-08-05
 17:39:56.830635+00
 test_slot2 |            |         1 |          0 |         1 | 100.00%  | 2022-08-05
 17:34:04.036795+00
(2 rows)
```

aurora_stat_memctx_usage

Relata o uso do contexto de memória para cada processo do PostgreSQL.

Sintaxe

```
aurora_stat_memctx_usage()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `pid`: o ID do processo.
- `name`: o nome do contexto da memória.
- `allocated`: o número de bytes obtidos do subsistema de memória subjacente pelo contexto da memória.
- `used`: o número de bytes confirmados nos clientes do contexto de memória.

- `instances`: a contagem de contextos existentes desse tipo no momento.

Observações de uso

Essa função exibe o uso do contexto de memória para cada processo do PostgreSQL. Alguns processos são identificados como `anonymous`. Os processos não são expostos porque contêm palavras-chave restritas.

Essa função está disponível para as seguintes versões do Aurora PostgreSQL:

- 15.3 e versões 15 posteriores
- 14.8 e versões 14 posteriores
- 13.11 e versões 13 posteriores
- 12.15 e versões 12 posteriores
- 11.20 e versões 11 posteriores

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_stat_memctx_usage`.

```
=> SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                             | allocated |   used  | instances
-------+----------------------------------+-----------+---------+-----------
123864 | Miscellaneous                    |     19520 |   15064 |         3
123864 | Aurora File Context              |      8192 |     616 |         1
123864 | Aurora WAL Context               |      8192 |     296 |         1
123864 | CacheMemoryContext               |    524288 |  422600 |         1
123864 | Catalog tuple context            |     16384 |   13736 |         1
123864 | ExecutorState                    |     32832 |   28304 |         1
123864 | ExprContext                      |      8192 |    1720 |         1
123864 | GWAL record construction         |      1024 |     832 |         1
123864 | MdSmgr                           |      8192 |     296 |         1
123864 | MessageContext                   |    532480 |  353832 |         1
123864 | PortalHeapMemory                 |      1024 |     488 |         1
123864 | PortalMemory                     |      8192 |     576 |         1
123864 | printtup                         |      8192 |     296 |         1
123864 | RelCache hash table entries      |      8192 |    8152 |         1
```

```
123864 | RowDescriptionContext           |      8192 |    1344 |         1
123864 | smgr relation context           |      8192 |     296 |         1
123864 | Table function arguments        |      8192 |     352 |         1
123864 | TopTransactionContext           |      8192 |     632 |         1
123864 | TransactionAbortContext         |     32768 |     296 |         1
123864 | WAL record construction         |     50216 |   43904 |         1
123864 | hash table                      |     65536 |   52744 |         6
123864 | Relation metadata               |    191488 |  124240 |        87
104992 | Miscellaneous                   |      9280 |    7728 |         3
104992 | Aurora File Context             |      8192 |     376 |         1
104992 | Aurora WAL Context              |      8192 |     296 |         1
104992 ||Autovacuum Launcher             |      8192 |     296 |         1
104992 | Autovacuum database list        |     16384 |     744 |         2
104992 | CacheMemoryContext              |    262144 |  140288 |         1
104992 | Catalog tuple context           |      8192 |     296 |         1
104992 | GWAL record construction        |      1024 |     832 |         1
104992 | MdSmgr                          |      8192 |     296 |         1
104992 | PortalMemory                    |      8192 |     296 |         1
104992 | RelCache hash table entries     |      8192 |     296 |         1
104992 | smgr relation context           |      8192 |     296 |         1
104992 | Autovacuum start worker (tmp)   |      8192 |     296 |         1
104992 | TopTransactionContext           |     16384 |     592 |         2
104992 | TransactionAbortContext         |     32768 |     296 |         1
104992 | WAL record construction         |     50216 |   43904 |         1
104992 | hash table                      |     49152 |   34024 |         4
(39 rows)
```

Algumas palavras-chave restritas serão ocultadas, e a saída terá a seguinte aparência:

```
postgres=>SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                            | allocated |   used  | instances
-------+---------------------------------+-----------+---------+-----------
  5482 | anonymous                       |      8192 |     456 |         1
  5482 | anonymous                       |      8192 |     296 |         1
```

aurora_stat_optimized_reads_cache

Essa função mostra estatísticas de cache em camadas.

Sintaxe

```
aurora_stat_optimized_reads_cache()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- `total_size`: tamanho total do cache do optimized reads.
- `used_size`: tamanho de página usado em cache do optimized reads.

Observações de uso

Essa função está disponível nas seguintes versões do Aurora PostgreSQL:

- 15.4 e versões 15 posteriores
- 14.9 e versões 14 posteriores

Exemplos

Veja a seguir um exemplo de saída em uma instância r6gd.8xlarge:

```
=> select pg_size_pretty(total_size) as total_size, pg_size_pretty(used_size)
                as used_size from aurora_stat_optimized_reads_cache();
total_size | used_size
-----------+-----------
1054 GB    | 975 GB
```

aurora_stat_plans

Exibe uma linha para cada plano de execução monitorado.

Sintaxe

```
aurora_stat_plans(
```

```
    showtext
)
```

Argumentos

- showtext: mostra a consulta e o texto do plano. Os valores válidos são NULL, true ou false. True mostrará a consulta e o texto do plano.

Tipo de retorno

Exibe uma linha para cada plano monitorado que contém todas as colunas de `aurora_stat_statements` e as colunas adicionais a seguir.

- planid: identificador do plano.
- explain_plan: explica o texto do plano.
- plan_type:
  - `no plan`: nenhum plano foi capturado.
  - `estimate`: plano capturado com custos estimados.
  - `actual`: plano capturado com EXPLAIN ANALYZE.
- plan_captured_time: última vez que um plano foi capturado.

Observações de uso

`aurora_compute_plan_id` deve estar habilitado e `pg_stat_statements` deve estar em `shared_preload_libraries` para que os planos sejam monitorados.

O número de planos disponíveis é controlado pelo valor definido no parâmetro `pg_stat_statements.max`. Quando `aurora_compute_plan_id` está habilitado, é possível monitorar os planos até esse valor especificado em `aurora_stat_plans`.

Essa função está disponível para o Aurora PostgreSQL versões 14.10, 15.5 e todas as outras versões posteriores.

Exemplos

No exemplo abaixo, os dois planos que são para o identificador de consulta -5471422286312252535 são capturados e as estatísticas das declarações são monitoradas pelo planid.

```
db1=# select calls, total_exec_time, planid, plan_captured_time, explain_plan
db1-# from aurora_stat_plans(true)
db1-# where queryid = '-5471422286312252535'

calls    |  total_exec_time   |   planid    |      plan_captured_time       |
                explain_plan
---------+--------------------+-------------+-------------------------------
+-----------------------------------------------------------------
 1532632 |  3209846.097107853 |  1602979607 | 2023-10-31 03:27:16.925497+00 | Update on
 pgbench_branches                                       +
         |                    |             |                               |   ->
 Bitmap Heap Scan on pgbench_branches                      +
         |                    |             |                               |
 Recheck Cond: (bid = 76)                                  +
         |                    |             |                               |         -
>  Bitmap Index Scan on pgbench_branches_pkey          +
         |                    |             |                               |
     Index Cond: (bid = 76)
   61365 | 124078.18012200127 | -2054628807 | 2023-10-31 03:20:09.85429+00  | Update on
 pgbench_branches                                       +
         |                    |             |                               |   ->
 Index Scan using pgbench_branches_pkey on pgbench_branches+
         |                    |             |                               |
 Index Cond: (bid = 17)
```

aurora_stat_reset_wal_cache

Redefine o contador para o cache lógico wal.

Sintaxe

Para redefinir um slot específico

```
SELECT * FROM aurora_stat_reset_wal_cache('slot_name')
```

Para redefinir todos os slots

```
SELECT * FROM aurora_stat_reset_wal_cache(NULL)
```

Argumentos

`NULL` ou `slot_name`

Tipo de retorno

Mensagem de status, string de texto

- Redefina o contador lógico do cache wal: mensagem de êxito. Esse texto é retornado quando a função é bem-sucedida.
- Slot de replicação não encontrado. Tente novamente. – Mensagem de falha. Esse texto é retornado quando a função não é bem-sucedida.

Observações de uso

Essa função está disponível para as versões a seguir.

- Aurora PostgreSQL 14.5 e posteriores
- Aurora PostgreSQL versão 13.8 e posteriores
- Aurora PostgreSQL versão 12.12 e posteriores
- Aurora PostgreSQL versão 11.17 e posteriores

Exemplos

O exemplo a seguir usa a função `aurora_stat_reset_wal_cache` para redefinir um slot denominado `test_results` e, depois, tenta redefinir um slot que não existe.

```
=> SELECT *
     FROM aurora_stat_reset_wal_cache('test_slot');
aurora_stat_reset_wal_cache
------------------------------------
 Reset the logical wal cache counter.
(1 row)
=> SELECT *
     FROM aurora_stat_reset_wal_cache('slot-not-exist');
aurora_stat_reset_wal_cache
-----------------------------------------------
 Replication slot not found. Please try again.
(1 row)
```

aurora_stat_resource_usage

Relata a utilização de recursos em tempo real, que consiste em métricas de recursos de backend e no uso da CPU para todos os processos de backend do Aurora PostgreSQL.

Sintaxe

```
aurora_stat_resource_usage()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com colunas:

- pid: identificador de processo.
- allocated_memory: memória total alocada por processo em bytes.
- used_memory: memória realmente usada por processo em bytes.
- cpu_usage_percent: porcentagem de uso de CPU do processo.

Observações de uso

Essa função exibe o uso de recursos de backend para cada processo de backend do Aurora PostgreSQL.

Essa função está disponível para as seguintes versões do Aurora PostgreSQL:

- Aurora PostgreSQL 17.5 e versões 17 posteriores
- Aurora PostgreSQL 16.9 e versões 16 posteriores
- Aurora PostgreSQL 15.13 e versões 15 posteriores
- Aurora PostgreSQL 14.18 e versões 14 posteriores
- Aurora PostgreSQL 13.21 e versões 13 posteriores

Exemplos

O exemplo a seguir mostra a saída da função `aurora_stat_resource_usage`.

```
=> select * from aurora_stat_resource_usage();
 pid  | allocated_memory | used_memory |   cpu_usage_percent
------+------------------+-------------+-----------------------
  666 |          1074032 |      333544 |   0.00729274882897963
  667 |           787312 |      287360 | 0.0029263928146372746
  668 |          3076776 |     1563488 |  0.006013116835953961
  684 |           803744 |      307480 |  0.002226855426881142
 2401 |          1232992 |      943144 |                     0
  647 |             8000 |         944 |   0.48853387812429855
  659 |           319344 |      243000 | 0.0004135602076683591
  663 |           262000 |      185736 |  0.008181301476644002
  664 |             9024 |        1216 |   0.10992313082653653
(9 rows)
```

aurora_stat_statements

Exibe todas as colunas `pg_stat_statements` e adiciona mais colunas no final.

Sintaxe

```
aurora_stat_statements(showtext boolean)
```

Argumentos

showtext booliano

Tipo de retorno

Registro SETOF com todas as colunas `pg_stat_statements` e as seguintes colunas adicionais. Para obter mais informações sobre as colunas `pg_stat_statements`, consulte `pg_stat_statements`.

Você pode redefinir as estatísticas dessa função usando `pg_stat_statements_reset()`.

- `storage_blks_read`: número total de blocos compartilhados lidos do armazenamento do aurora de acordo com essa instrução.
- `orcache_blks_hit`: número total de acessos ao cache do optimized reads nessa instrução.
- `storage_blk_read_time`: se `track_io_timing` estiver habilitado, ele vai monitorar o tempo total gasto nessa declaração com a leitura de blocos compartilhados do armazenamento do

Aurora, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.

- `local_blk_read_time`: se `track_io_timing` estiver habilitado, ele vai monitorar o tempo total gasto nessa declaração com a leitura de blocos locais, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.
- `orcache_blk_read_time`: se `track_io_timing` estiver habilitado, ele vai monitorar o tempo total gasto nessa declaração com a leitura de blocos compartilhados do cache de leituras otimizadas, em milissegundos; caso contrário, o valor será zero. Para obter mais informações, consulte track_io_timing.
- `total_plan_peakmem`: soma total dos valores de pico de memória durante a fase de planejamento para todas as chamadas para essa declaração. Para ver o pico médio de memória durante o planejamento da declaração, divida esse valor pelo número de chamadas.
- `min_plan_peakmem`: o menor valor de pico de memória observado durante o planejamento em todas as chamadas para essa declaração.
- `max_plan_peakmem`: o maior valor de pico de memória durante o planejamento observado em todas as chamadas para essa declaração.
- `total_exec_peakmem`: soma total dos valores de pico de memória durante a fase de execução para todas as chamadas para essa declaração. Para ver o pico médio de memória durante a execução da declaração, divida esse valor pelo número de chamadas.
- `min_exec_peakmem`: o menor valor de pico de memória, em bytes, observado durante a execução em todas as chamadas para essa declaração.
- `max_exec_peakmem`: o maior valor de pico de memória, em bytes, observado durante a execução em todas as chamadas para essa declaração.

> **Note**
>
> `total_plan_peakmen`, `min_plan_peakmem` e `max_plan_peakmem` são monitorados apenas quando a configuração `pg_stat_statements.track_planning` está ativada.

Observações de uso

Para usar a função aurora_stat_statements(), é necessário incluir a extensão `pg_stat_statements` no parâmetro `shared_preload_libraries`.

Essa função está disponível nas seguintes versões do Aurora PostgreSQL:

- 15.4 e versões 15 posteriores
- 14.9 e versões 14 posteriores

As colunas que mostram o pico de memória estão disponíveis nas seguintes versões:

- 16.3 e versões posteriores
- 15.7 e versões posteriores
- 14.12 e versões posteriores

Exemplos

O exemplo a seguir mostra como ele carrega todas as colunas pg_stat_statements e inclui 11 novas colunas no final:

```
=> select * from aurora_stat_statements(true) where query like 'with window_max%';
-[ RECORD 1 ]----------
+-----------------------------------------------------------------------------------------------
userid                | 16409
dbid                  | 5
toplevel              | t
queryid               | -8347523682669847482
query                 | with window_max as (select custid, max(scratch) over (order by
 scratch rows between $1 preceding
and $2 following) wmax from ts) select sum(wmax), max(custid) from window_max
plans                 | 0
total_plan_time       | 0
min_plan_time         | 0
max_plan_time         | 0
mean_plan_time        | 0
stddev_plan_time      | 0
calls                 | 4
total_exec_time       | 254.105121
min_exec_time         | 57.503164000000005
max_exec_time         | 68.687418
mean_exec_time        | 63.52628025
stddev_exec_time      | 5.150765359979643
rows                  | 4
shared_blks_hit       | 200192
shared_blks_read      | 0
```

```
shared_blks_dirtied    | 0
shared_blks_written    | 0
local_blks_hit         | 0
local_blks_read        | 0
local_blks_dirtied     | 0
local_blks_written     | 0
temp_blks_read         | 0
temp_blks_written      | 0
blk_read_time          | 0
blk_write_time         | 0
temp_blk_read_time     | 0
temp_blk_write_time    | 0
wal_records            | 0
wal_fpi                | 0
wal_bytes              | 0
jit_functions          | 0
jit_generation_time    | 0
jit_inlining_count     | 0
jit_inlining_time      | 0
jit_optimization_count | 0
jit_optimization_time  | 0
jit_emission_count     | 0
jit_emission_time      | 0
storage_blks_read      | 0
orcache_blks_hit       | 0
storage_blk_read_time  | 0
local_blk_read_time    | 0
orcache_blk_read_time  | 0
total_plan_peakmem     | 0
min_plan_peakmem       | 0
max_plan_peakmem       | 0
total_exec_peakmem     | 6356224
min_exec_peakmem       | 1589056
max_exec_peakmem       | 1589056
```

aurora_stat_system_waits

Relata informações de eventos de espera para a instância de banco de dados Aurora PostgreSQL.

Sintaxe

```
aurora_stat_system_waits()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF

Observações de uso

Essa função retorna o número cumulativo de esperas e o tempo de espera cumulativo para cada evento de espera gerado pela instância de banco de dados à qual você está conectado no momento.

O conjunto de registros retornado inclui os seguintes campos:

- `type_id`: o ID do tipo do evento de espera.
- `event_id`: o ID do evento de espera.
- `waits`: o número de vezes que o evento de espera ocorreu.
- `wait_time`: a quantidade total de tempo em microssegundos gasto aguardando esse evento.

As estatísticas retornadas por essa função são redefinidas quando uma instância de banco de dados é reiniciada.

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_stat_system_waits`.

```
=> SELECT *
    FROM aurora_stat_system_waits();
 type_id | event_id  |   waits   |  wait_time
---------+-----------+-----------+--------------
       1 |  16777219 |        11 |        12864
       1 |  16777220 |       501 |       174473
       1 |  16777270 |     53171 |     23641847
       1 |  16777271 |        23 |       319668
       1 |  16777274 |        60 |        12759
.
.
.
```

```
      10 | 167772231 |    204596 |    790945212
      10 | 167772232 |         2 |        47729
      10 | 167772234 |         1 |          888
      10 | 167772235 |         2 |           64
```

O exemplo a seguir mostra como é possível usar essa função junto com `aurora_stat_wait_event` e `aurora_stat_wait_type` para produzir resultados mais legíveis.

```
=> SELECT type_name,
            event_name,
            waits,
            wait_time
       FROM aurora_stat_system_waits()
NATURAL JOIN aurora_stat_wait_event()
NATURAL JOIN aurora_stat_wait_type();

 type_name |       event_name       |  waits   |  wait_time
-----------+------------------------+----------+--------------
 LWLock    | XidGenLock             |       11 |        12864
 LWLock    | ProcArrayLock          |      501 |       174473
 LWLock    | buffer_content         |    53171 |     23641847
 LWLock    | rdsutils               |        2 |        12764
 Lock      | tuple                  |    75686 |   2033956052
 Lock      | transactionid          |  1765147 |  47267583409
 Activity  | AutoVacuumMain         |   136868 |  56305604538
 Activity  | BgWriterHibernate      |     7486 |  55266949471
 Activity  | BgWriterMain           |     7487 |   1508909964
.
.
.
 IO        | SLRURead               |        3 |        11756
 IO        | WALWrite               | 52544463 |    388850428
 IO        | XactSync               |   187073 |    597041642
 IO        | ClogRead               |        2 |        47729
 IO        | OutboundCtrlRead       |        1 |          888
 IO        | OutboundCtrlWrite      |        2 |           64
```

aurora_stat_wait_event

Lista todos os eventos de espera compatíveis para o Aurora PostgreSQL. Para obter informações sobre eventos de espera do Aurora PostgreSQL, consulte Eventos de espera do Amazon Aurora PostgreSQL.

Sintaxe

```
aurora_stat_wait_event()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- type_id: o ID do tipo do evento de espera.
- event_id: o ID do evento de espera.
- type_name: nome do tipo de espera
- event_name: nome do evento de espera

Observações de uso

Para ver nomes de eventos com tipos de eventos em vez de IDs, use essa função junto com outras funções, como `aurora_stat_wait_type` e `aurora_stat_system_waits`. Os nomes de eventos de espera retornados por esta função são os mesmos que os retornados pela função `aurora_wait_report`.

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_stat_wait_event`.

```
=>  SELECT *
    FROM aurora_stat_wait_event();

 type_id | event_id  |               event_name
---------+-----------+------------------------------------------
       1 |  16777216 | <unassigned:0>
       1 |  16777217 | ShmemIndexLock
       1 |  16777218 | OidGenLock
       1 |  16777219 | XidGenLock
.
.
.
```

```
       9 | 150994945 | PgSleep
       9 | 150994946 | RecoveryApplyDelay
      10 | 167772160 | BufFileRead
      10 | 167772161 | BufFileWrite
      10 | 167772162 | ControlFileRead
.
.
.
      10 | 167772226 | WALInitWrite
      10 | 167772227 | WALRead
      10 | 167772228 | WALSync
      10 | 167772229 | WALSyncMethodAssign
      10 | 167772230 | WALWrite
      10 | 167772231 | XactSync
.
.
.
      11 | 184549377 | LsnAllocate
```

O exemplo a seguir junta `aurora_stat_wait_type` e `aurora_stat_wait_event` para retornar nomes de tipos e nomes de eventos para melhorar a legibilidade.

```
=> SELECT *
    FROM aurora_stat_wait_type() t
    JOIN aurora_stat_wait_event() e
      ON t.type_id = e.type_id;

 type_id | type_name | type_id | event_id  |               event_name
---------+-----------+---------+-----------+-----------------------------------------
       1 | LWLock    |       1 |  16777216 | <unassigned:0>
       1 | LWLock    |       1 |  16777217 | ShmemIndexLock
       1 | LWLock    |       1 |  16777218 | OidGenLock
       1 | LWLock    |       1 |  16777219 | XidGenLock
       1 | LWLock    |       1 |  16777220 | ProcArrayLock
.
.
.
       3 | Lock      |       3 |  50331648 | relation
       3 | Lock      |       3 |  50331649 | extend
       3 | Lock      |       3 |  50331650 | page
       3 | Lock      |       3 |  50331651 | tuple
.
.
```

```
.
      10 | IO         |      10 | 167772214 | TimelineHistorySync
      10 | IO         |      10 | 167772215 | TimelineHistoryWrite
      10 | IO         |      10 | 167772216 | TwophaseFileRead
      10 | IO         |      10 | 167772217 | TwophaseFileSync
.
.
.
      11 | LSN        |      11 | 184549376 | LsnDurable
```

aurora_stat_wait_type

Lista todos os tipos de espera compatíveis para o Aurora PostgreSQL.

Sintaxe

```
aurora_stat_wait_type()
```

Argumentos

Nenhum

Tipo de retorno

Registro SETOF com as seguintes colunas:

- type_id: o ID do tipo do evento de espera.
- type_name: nome do tipo de espera.

Observações de uso

Para ver nomes de eventos de espera com tipos de eventos de espera em vez de IDs, use essa função junto com outras funções, como `aurora_stat_wait_event` e `aurora_stat_system_waits`. Os nomes de tipos de espera retornados por esta função são os mesmos que os retornados pela função `aurora_wait_report`.

Exemplos

O exemplo a seguir mostra os resultados de uma chamada à função `aurora_stat_wait_type`.

```
=> SELECT *
```

```
     FROM aurora_stat_wait_type();
 type_id | type_name
---------+-----------
       1 | LWLock
       3 | Lock
       4 | BufferPin
       5 | Activity
       6 | Client
       7 | Extension
       8 | IPC
       9 | Timeout
      10 | IO
      11 | LSN
```

aurora_version

Retorna o valor de string do número de versão do Amazon Aurora edição compatível com PostgreSQL.

Sintaxe

```
aurora_version()
```

Argumentos

Nenhum

Tipo de retorno

String CHAR ou VARCHAR

Observações de uso

Essa função exibe a versão do mecanismo de banco de dados do Amazon Aurora edição compatível com PostgreSQL. O número da versão é retornado como uma string formatada como *major*.*minor*.*patch*. Para obter mais informações sobre os número de versão do Aurora PostgreSQL, consulte Número de versão do Aurora.

Você pode escolher quando aplicar atualizações de versões secundárias definindo a janela de manutenção do cluster de banco de dados do Aurora PostgreSQL. Para saber como, consulte Manutenção de um cluster de banco de dados do Amazon Aurora.

A partir do lançamento das versões 13.3, 12.8, 11.13, 10.18 do Aurora PostgreSQL e para todas as outras posteriores, os números de versão do Aurora seguem os números de versão do PostgreSQL. Para obter mais informações sobre todas as versões do Aurora PostgreSQL, consulte Atualizações do Aurora PostgreSQL nas Notas de lançamento do Aurora PostgreSQL.

Exemplos

O exemplo a seguir mostra os resultados da chamada da função `aurora_version` em um cluster de banco de dados do Aurora PostgreSQL que executa o PostgreSQL 12.7, Aurora PostgreSQL versão 4.2, e depois executa a mesma função em um cluster que executa o Aurora PostgreSQL versão 13.3.

```
=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
```

Este exemplo mostra como usar a função com várias opções para obter mais detalhes sobre a versão do Aurora PostgreSQL. Este exemplo tem um número de versão do Aurora distinto do número de versão do PostgreSQL.

```

=> SHOW SERVER_VERSION;
 server_version
----------------
 12.7
(1 row)

=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
(1 row)

=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 12.7
```

```
(1 row)

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
------------------------------------------------------------------------------------------------
 PostgreSQL 12.7 on aarch64-unknown-linux-gnu, compiled by aarch64-unknown-linux-gnu-
gcc (GCC) 7.4.0, 64-bit
(1 row)

=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-------------------------------------------------
 Aurora: 4.2.2 Compatible with PostgreSQL: 12.7
(1 row)
```

Este próximo exemplo usa a função com as mesmas opções no exemplo anterior. Este exemplo não tem um número de versão do Aurora distinto do número de versão do PostgreSQL.

```
=> SHOW SERVER_VERSION;
server_version
----------------
 13.3

=> SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
---------------------------
 13.3

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
=> SELECT 'Aurora: '
    || aurora_version()
```

```
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-------------------------------------------------------
 Aurora: 13.3.0 Compatible with PostgreSQL: 13.3
```

aurora_volume_logical_start_lsn

Retorna o número de sequência de log (LSN) usado para identificar o início de um registro no stream lógico do log de gravação antecipada (WAL) do volume de cluster do Aurora.

Sintaxe

```
aurora_volume_logical_start_lsn()
```

Argumentos

Nenhum

Tipo de retorno

`pg_lsn`

Observações de uso

Essa função identifica o início do registro no stream lógico de WAL para determinado volume de cluster do Aurora. Você pode usar essa função enquanto realiza um upgrade de versão principal usando a replicação lógica e a clonagem rápida do Aurora para determinar o LSN em que um snapshot ou clone de banco de dados é gerado. Depois, você pode usar a replicação lógica para transmitir continuamente os dados mais recentes registrados depois do LSN e sincronizar as alterações entre publicador e assinante.

Para obter mais informações sobre como usar a replicação lógica para um upgrade de versão principal, consulte Usar replicação lógica para realizar uma atualização de versão principal do Aurora PostgreSQL.

Essa função está disponível nas seguintes versões do Aurora PostgreSQL:

- Versão 15.2 e versões 15 posteriores
- Versão 14.3 e versões 14 posteriores
- Versão 13.6 e versões 13 posteriores

- Versão 12.10 e versões 12 posteriores
- Versão 11.15 e versões 11 posteriores
- Versão 10.20 e versões 10 posteriores

Exemplos

Você pode obter o número de sequência de log (LSN) usando a seguinte consulta:

```
postgres=> SELECT aurora_volume_logical_start_lsn();

aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

aurora_wait_report

Essa função mostra a atividade de eventos de espera por um período.

Sintaxe

```
aurora_wait_report([time])
```

Argumentos

time (opcional)

A hora em segundos. O padrão é 10 segundos.

Tipo de retorno

Registro SETOF com as seguintes colunas:

- type_name: nome do tipo de espera
- event_name: nome do evento de espera
- wait: número de esperas
- wait_time: tempo de espera em milissegundos
- ms_per_wait: média de milissegundos pelo número de uma espera

- waits_per_xact: média de esperas pelo número de uma transação
- ms_per_wait: média de milissegundos pelo número de transações

Observações de uso

Essa função está disponível a partir do Aurora PostgreSQL versão 1.1 compatível com PostgreSQL 9.6.6 e versões posteriores.

Para usar essa função, você precisa primeiro criar a extensão `aurora_stat_utils` do Aurora PostgreSQL, da seguinte forma:

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Para obter mais informações sobre as versões de extensão do Aurora PostgreSQL disponíveis, consulte Versões de extensão do Amazon Aurora PostgreSQL em Notas de lançamento do Aurora PostgreSQL.

Essa função calcula os eventos de espera no nível da instância comparando dois snapshots de dados estatísticos da função aurora_stat_system_waits() e das visualizações estatísticas pg_stat_database do PostgreSQL.

Para obter mais informações sobre `aurora_stat_system_waits()` e `pg_stat_database`, consulte Coletor de estatísticas na documentação do PostgreSQL.

Quando executada, essa função tira um snapshot inicial, aguarda o número de segundos especificado e, depois, tira um segundo snapshot. A função compara os dois snapshots e retorna a diferença. Essa diferença representa a atividade da instância para esse intervalo de tempo.

Na instância de gravador, a função também exibe o número de transações confirmadas e TPS (transações por segundo). Essa função retorna informações no nível da instância e inclui todos os bancos de dados na instância.

Exemplos

Este exemplo mostra como criar a extensão aurora_stat_utils para poder usar a função aurora_wait_report.

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Este exemplo mostra como conferir o relatório de espera por 10 segundos.

```
=> SELECT *
     FROM aurora_wait_report();
NOTICE:  committed 34 transactions in 10 seconds (tps 3)
 type_name |    event_name     | waits | wait_time | ms_per_wait | waits_per_xact |
 ms_per_xact
-----------+-------------------+-------+-----------+-------------+----------------
+-------------
 Client    | ClientRead        |    26 |  30003.00 |    1153.961 |           0.76 |
 882.441
 Activity  | WalWriterMain     |    50 |  10051.32 |     201.026 |           1.47 |
 295.627
 Timeout   | PgSleep           |     1 |  10049.52 |   10049.516 |           0.03 |
 295.574
 Activity  | BgWriterHibernate |     1 |  10048.15 |   10048.153 |           0.03 |
 295.534
 Activity  | AutoVacuumMain    |    18 |   9941.66 |     552.314 |           0.53 |
 292.402
 Activity  | BgWriterMain      |     1 |    201.09 |     201.085 |           0.03 |
   5.914
 IO        | XactSync          |    15 |     25.34 |       1.690 |           0.44 |
   0.745
 IO        | RelationMapRead   |    12 |      0.54 |       0.045 |           0.35 |
   0.016
 IO        | WALWrite          |    84 |      0.21 |       0.002 |           2.47 |
   0.006
 IO        | DataFileExtend    |     1 |      0.02 |       0.018 |           0.03 |
   0.001
```

Este exemplo mostra como conferir o relatório de espera por 60 segundos.

```
=> SELECT *
     FROM aurora_wait_report(60);
NOTICE:  committed 1544 transactions in 60 seconds (tps 25)
 type_name |       event_name       |  waits  | wait_time | ms_per_wait |
 waits_per_xact | ms_per_xact
-----------+------------------------+---------+-----------+-------------
+----------------+-------------
 Lock      | transactionid          |    6422 | 477000.53 |      74.276 |
 4.16 |     308.938
 Client    | ClientRead             |    8265 | 270752.99 |      32.759 |
 5.35 |     175.358
```

```
Activity  | CheckpointerMain       |       1 |  60100.25 |   60100.246 |
0.00 |      38.925
Timeout   | PgSleep                |       1 |  60098.49 |   60098.493 |
0.00 |      38.924
Activity  | WalWriterMain          |     296 |  60010.99 |     202.740 |
0.19 |      38.867
Activity  | AutoVacuumMain         |     107 |  59827.84 |     559.139 |
0.07 |      38.749
Activity  | BgWriterMain           |     290 |  58821.83 |     202.834 |
0.19 |      38.097
IO        | XactSync               |    1295 |  55220.13 |      42.641 |
0.84 |      35.764
IO        | WALWrite               | 6602259 |  47810.94 |       0.007 |
4276.07 |      30.966
Lock      | tuple                  |     473 |  29880.67 |      63.173 |
0.31 |      19.353
LWLock    | buffer_mapping         |     142 |   3540.13 |      24.930 |
0.09 |       2.293
Activity  | BgWriterHibernate      |     290 |   1124.15 |       3.876 |
0.19 |       0.728
IO        | BufFileRead            |    7615 |    618.45 |       0.081 |
4.93 |       0.401
LWLock    | buffer_content         |      73 |    345.93 |       4.739 |
0.05 |       0.224
LWLock    | lock_manager           |      62 |    191.44 |       3.088 |
0.04 |       0.124
IO        | RelationMapRead        |      72 |      5.16 |       0.072 |
0.05 |       0.003
LWLock    | ProcArrayLock          |       1 |      2.01 |       2.008 |
0.00 |       0.001
IO        | ControlFileWriteUpdate |       2 |      0.03 |       0.013 |
0.00 |       0.000
IO        | DataFileExtend         |       1 |      0.02 |       0.018 |
0.00 |       0.000
IO        | ControlFileSyncUpdate  |       1 |      0.00 |       0.000 |
0.00 |       0.000
```

Amazon Aurora PostgreSQL parameters

Você gerencia o cluster de bancos de dados Amazon Aurora da mesma maneira que gerencia outras instâncias de banco de dados do Amazon RDS, usando parâmetros em um grupo de parâmetros de banco de dados. No entanto, o Amazon Aurora difere do Amazon RDS porque um cluster de bancos de dados Aurora tem várias instâncias de banco de dados. Dessa forma, alguns dos parâmetros que

você usa para gerenciar seu cluster de bancos de dados Amazon Aurora se aplicam a todo o cluster, enquanto outros parâmetros se aplicam apenas a uma instância de banco de dados particular no cluster de banco de dados, como o seguinte:

- Grupo de parâmetros do cluster de banco de dados: um grupo de parâmetros do cluster de banco de dados contém o conjunto de parâmetros de configuração do mecanismo que se aplicam em todo o cluster de bancos de dados Aurora. Por exemplo, o gerenciamento de cache de cluster é um recurso de um cluster de bancos de dados Aurora que é controlado pelo parâmetro `apg_ccm_enabled` que faz parte do grupo de parâmetros de cluster de banco de dados. O grupo de parâmetros do cluster de banco de dados também contém configurações padrão para o grupo de parâmetros de banco de dados das instâncias de banco de dados que compõem o cluster.
- Grupo de parâmetros de banco de dados: um grupo de parâmetros de banco de dados é o conjunto de valores de configuração do mecanismo que se aplicam a uma instância de banco de dados específica desse tipo de mecanismo. Os grupos de parâmetros de banco de dados para o mecanismo de banco de dados do PostgreSQL são usados por uma instância de banco de dados do RDS para PostgreSQL e um cluster de bancos de dados do Aurora PostgreSQL. Essas definições de configuração aplicam-se a propriedades que podem variar entre as instâncias de banco de dados em um cluster do Aurora, como os tamanhos dos buffers de memória.

Você gerencia parâmetros em nível de cluster em grupos de parâmetros de cluster de banco de dados. Você gerencia parâmetros em nível de instância em grupos de parâmetros de banco de dados. Você pode gerenciar parâmetros usando o console do Amazon RDS, a AWS CLI ou a API do Amazon RDS. Há comandos separados para gerenciar parâmetros no nível do cluster e parâmetros no nível da instância.

- Para gerenciar parâmetros no nível do cluster em um grupo de parâmetros do cluster de banco de dados, use o comando modify-db-cluster-parameter-group da AWS CLI.
- Para gerenciar parâmetros no nível da instância em um grupo de parâmetros de banco de dados de uma instância de banco de dados em um cluster de banco de dados, use o comando modify-db-parameter-group da AWS CLI.

Para saber mais sobre a AWS CLI, consulte Usar a AWS CLI no Guia do usuário do AWS Command Line Interface.

Para obter mais informações sobre grupos de parâmetros, consulte Grupos de parâmetros para Amazon Aurora.

Exibindo parâmetros de banco de dados e de cluster de bancos de dados Aurora PostgreSQL

Você pode exibir todos os grupos de parâmetros padrão disponíveis para instâncias de banco de dados do RDS para PostgreSQL e para clusters de bancos de dados Aurora PostgreSQL no Console de gerenciamento da AWS. Os grupos de parâmetros padrão para todos os mecanismos de banco de dados e tipos e versões de cluster de banco de dados são listados para cada região da AWS. Todos os grupos de parâmetros personalizados também estão listados.

Em vez de visualizar no Console de gerenciamento da AWS, você também pode listar parâmetros contidos em grupos de parâmetros de cluster de banco de dados e grupos de parâmetros de banco de dados usando a AWS CLI ou a API do Amazon RDS. Por exemplo, para listar os parâmetros em um grupo de parâmetros do cluster de banco de dados, use o comando da AWS CLI describe-db-cluster-parameters.

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12
```

O comando retorna descrições JSON detalhadas de cada parâmetro. Para reduzir a quantidade de informações retornadas, você pode especificar o que deseja usando a opção `--query`. Por exemplo, você pode obter o nome do parâmetro, sua descrição e os valores permitidos para o grupo de parâmetro de cluster de banco de dados padrão do Aurora PostgreSQL 12 da seguinte forma:

Para Linux, macOS ou Unix:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Para Windows:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Um grupo de parâmetros de cluster de bancos de dados Aurora inclui o grupo de parâmetros da instância de banco de dados e os valores padrão para um determinado mecanismo de banco de dados do Aurora Você pode obter a lista de parâmetros de banco de dados do mesmo grupo de

parâmetros padrão do Aurora PostgreSQL usando o comando da AWS CLI describe-db-parameters como mostrado a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Para Windows:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Os comandos anteriores retornam listas de parâmetros do cluster de banco de dados ou grupo de parâmetros de banco de dados com descrições e outros detalhes especificados na consulta. Veja a seguir uma resposta de exemplo.

```
[
    [
        {
            "ParameterName": "apg_enable_batch_mode_function_execution",
            "ApplyType": "dynamic",
            "Description": "Enables batch-mode functions to process sets of rows at a
 time.",
            "AllowedValues": "0,1"
        }
    ],
    [
        {
            "ParameterName": "apg_enable_correlated_any_transform",
            "ApplyType": "dynamic",
            "Description": "Enables the planner to transform correlated ANY Sublink
 (IN/NOT IN subquery) to JOIN when possible.",
            "AllowedValues": "0,1"
        }
    ],...
```

A seguir estão as tabelas que contêm valores para o parâmetro de cluster de banco de dados padrão e o parâmetro de banco de dados para o Aurora PostgreSQL versão 14.

Parâmetros no nível do cluster do Aurora PostgreSQL

É possível visualizar parâmetros no nível do cluster disponíveis para uma versão do Aurora PostgreSQL usando o Console de Gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS. Para obter informações sobre como visualizar os parâmetros em grupos de parâmetros de cluster de banco de dados do Aurora PostgreSQL no console do RDS, consulte Visualizar valores de parâmetros de um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Alguns parâmetros no nível de cluster não estão disponíveis em todas as versões e alguns estão sendo descontinuados. Para obter informações sobre como visualizar os parâmetros de uma versão específica do Aurora PostgreSQL, consulte Exibindo parâmetros de banco de dados e de cluster de bancos de dados Aurora PostgreSQL.

Por exemplo, a tabela a seguir lista os parâmetros disponíveis no grupo de parâmetros de cluster de banco de dados padrão para o Aurora PostgreSQL versão 14. Se você criar um cluster de bancos de dados Aurora PostgreSQL sem especificar seu próprio grupo de parâmetros de banco de dados personalizado, seu cluster de banco de dados será criado usando o grupo de parâmetros de cluster de banco de dados padrão do Aurora para a versão escolhida, como `default.aurora-postgresql14`, `default.aurora-postgresql13` e assim por diante.

Para obter uma lista dos parâmetros de instância de banco de dados para o mesmo grupo de parâmetros padrão do cluster de banco de dados, consulte Parâmetros no nível da instância do Aurora PostgreSQL.

Nome do parâmetro	Descrição	Padrão
ansi_constraint_trigger_ordering	Altere a ordem de disparo dos acionadores de restrição para ser compatível com o padrão ANSI SQL.	–
ansi_force_foreign_key_checks	Certifique-se de que ações referenciais, como exclusão em cascata ou atualização em cascata, sempre ocorrerão independentemente dos vários contextos de acionadores existentes para a ação.	–
ansi_qualified_update_set_target	Suporta qualificadores de tabela e esquema em UPDATE... Instruções SET.	–

Nome do parâmetro	Descrição	Padrão
apg_ccm_enabled	Habilita ou desabilita o gerenciamento de cache de cluster para o cluster.	–
apg_enable_batch_mode_function_execution	Permite que as funções em modo em lote processem conjuntos de linhas de cada vez.	–
apg_enable_correlated_any_transform	Permite que o planejador transforme ANY Sublink correlacionado (subconsulta IN/NOT IN) para JOIN quando possível.	–
apg_enable_function_migration	Permite que o planejador migre funções escalares elegíveis para a cláusula FROM.	–
apg_enable_not_in_transform	Permite que o planejador transforme a subconsulta NOT IN em ANTI JOIN quando possível.	–
apg_enable_remove_redundant_inner_joins	Permite que o planejador remova junções internas redundantes.	–
apg_enable_semijoin_push_down	Permite o uso de filtros de semijunção para junções de hash.	–
apg_plan_mgmt.capture_plan_baselines	Modo de linha de base do plano de captura. manual: habilita a captura de plano para qualquer instrução SQL, desativado: desabilita captura de plano, automático: habilita a captura de plano para instruções em pg_stat_statement que atendam aos critérios de elegibilidade.	desligar
apg_plan_mgmt.max_databases	Define o número máximo de bancos de dados que podem gerenciar consultas usando apg_plan_mgmt.	10

Nome do parâmetro	Descrição	Padrão
apg_plan_mgmt.max_plans	Define o número máximo de planos que podem ser armazenados em cache por apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Número máximo de dias desde que um plano foi last_used antes de um plano ser excluído automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Custo total estimado do plano abaixo do qual um plano não aprovado será executado.	0
apg_plan_mgmt.use_plan_baselines	Usa somente planos aprovados ou fixos para instruções gerenciadas.	false
application_name	Define o nome da aplicação a ser informada em estatísticas e logs.	–
array_nulls	Permite entrada de elementos NULL em arrays.	–
aurora_compute_plan_id	Monitora planos de execução de consultas para detectar planos de execução que contribuem com a carga de banco de dados atual e para monitorar as estatísticas de desempenho de planos de execução ao longo do tempo. Consulte mais informações em Monitoring query execution plans for Aurora PostgreSQL.	ativado
authentication_timeout	(s) Define o tempo máximo permitido para concluir a autenticação de cliente.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE para registro em log de planos.	–

Nome do parâmetro	Descrição	Padrão
auto_explain.log_buffers	Uso de buffers de log.	–
auto_explain.log_format	Formato EXPLAIN a ser usado para registro em log de planos.	–
auto_explain.log_min_duration	Define o tempo de execução mínimo acima do qual as instruções serão registradas em log.	–
auto_explain.log_nested_statements	Registrar instruções aninhadas.	–
auto_explain.log_timing	Coleta dados de tempo, não apenas contagens de linhas.	–
auto_explain.log_triggers	Inclui estatísticas do acionador nos planos.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE para registro em log de planos.	–
auto_explain.sample_rate	Fração de consultas a serem processadas.	–
autovacuum	Inicia o subprocesso de autovacuum.	–
autovacuum_analyze_scale_factor	Número de inserções, atualizações ou exclusões de tuplas anteriores à análise, como uma fração de reltuplas.	0.05
autovacuum_analyze_threshold	Número mínimo de inserções, atualizações ou exclusões de tuplas antes da análise.	–
autovacuum_freeze_max_age	Idade na qual o autovacuum de uma tabela deve ocorrer para evitar a conclusão do ID de transação.	–

Nome do parâmetro	Descrição	Padrão
autovacuum_max_workers	Define o número máximo de processos de trabalho de autovacuum em execução simultaneamente.	GREATEST(DBInstanceClassMemory/64371566592,3)
autovacuum_multixact_freeze_max_age	Idade multixact na qual o autovacuum de uma tabela deve ocorrer para evitar o wraparound multixact.	–
autovacuum_naptime	(s) Tempo de suspensão entre execuções de autovacuum.	5
autovacuum_vacuum_cost_delay	(ms) Atraso de custo de vacuum, em milissegundos, para autovacuum.	5
autovacuum_vacuum_cost_limit	Valor do custo de vacuum disponível antes da suspensão, para autovacuum.	GREATEST(log(DBInstanceClassMemory/21474836480)*600,200)
autovacuum_vacuum_insert_scale_factor	Número de inserções de tupla antes do vacuum como uma fração de reltuplas.	–
autovacuum_vacuum_insert_threshold	Número mínimo de inserções de tupla antes do vacuum ou -1 para desabilitar os vacuums de inserção.	–
autovacuum_vacuum_scale_factor	Número de atualizações ou exclusões de tuplas antes de vacuum, como uma fração de reltuplas.	0.1
autovacuum_vacuum_threshold	Número mínimo de atualizações ou exclusões de tuplas antes do vacuum.	–
autovacuum_work_mem	(kB) Define a memória máxima a ser usada por cada processo de operador de autovacuum.	GREATEST(DBInstanceClassMemory/32768,131072)

Nome do parâmetro	Descrição	Padrão
babelfishpg_tds.default_server_name	O nome padrão do servidor Babelfish	Microsoft SQL Server
babelfishpg_tds.listen_addresses	Define o nome do host ou um ou mais endereços IP para ouvir o TDS.	*
babelfishpg_tds.port	Define a porta TCP TDS na qual o servidor escuta.	1433
babelfishpg_tds.tds_debug_log_level	Define o nível de registro em log no TDS, 0 desabilita o registro em log	1
babelfishpg_tds.tds_default_numeric_precision	Define a precisão padrão do tipo numérico a ser enviada nos metadados da coluna TDS se o mecanismo não especificar uma.	38
babelfishpg_tds.tds_default_numeric_scale	Define a escala padrão do tipo numérico a ser enviada nos metadados da coluna TDS se o mecanismo não especificar uma.	8
babelfishpg_tds.tds_default_packet_size	Define o tamanho do pacote padrão para todos os clientes do SQL Server sendo conectados	4096
babelfishpg_tds.tds_default_protocol_version	Define uma versão padrão do protocolo TDS para todos os clientes sendo conectados	DEFAULT
babelfishpg_tds.tds_ssl_encrypt	Define a opção Criptografia SSL	0
babelfishpg_tds.tds_ssl_max_protocol_version	Define a versão máxima do protocolo SSL/TLS a ser utilizada na sessão TDS.	TLSv1.2

Nome do parâmetro	Descrição	Padrão
babelfishpg_tds.tds_ssl_min_protocol_version	Define a versão mínima do protocolo SSL/TLS a ser utilizada na sessão TDS.	TLSv1.2 do Aurora PostgreSQL versão 16, TLSv1 para versões anteriores ao Aurora PostgreSQL versão 16
babelfishpg_tsql.default_locale	Localidade padrão a ser usada para agrupamentos criados por CREATE COLLATION.	en-US
babelfishpg_tsql.migration_mode	Define se há suporte para vários bancos de dados de usuários.	multi-db do Aurora PostgreSQL versão 16, single-db para versões anteriores ao Aurora PostgreSQL versão 16
babelfishpg_tsql.server_collation_name	Nome do agrupamento de servidores padrão	sql_latin1_general_cp1_ci_as
babelfishpg_tsql.version	Define a saída da variável @@VERSION.	padrão
backend_flush_after	(8Kb) Número de páginas após as quais as gravações executadas anteriormente são liberadas para o disco.	–
backslash_quote	Define se \\ é permitido em literais de string.	–
backtrace_functions	Registre o backtrace em busca de erros nessas funções.	–
bytea_output	Define o formato da saída para bytes.	–
check_function_bodies	Verifica corpos de funções durante CREATE FUNCTION.	–

Nome do parâmetro	Descrição	Padrão
client_connection_check_interval	Define o intervalo de tempo entre as verificações de desconexão durante a execução de consultas.	–
client_encoding	Define a codificação do conjunto de caracteres do cliente.	UTF8
client_min_messages	Define os níveis de mensagem enviados ao cliente.	–
compute_query_id	Calcula identificadores de consulta.	auto
config_file	Define o arquivo de configuração principal dos servidores.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Permite que o planejador use restrições para otimizar consultas.	–
cpu_index_tuple_cost	Define a estimativa do planejador sobre o custo do processamento de cada entrada de índice durante uma verificação de índice.	–
cpu_operator_cost	Define a estimativa do planejador sobre o custo do processamento de cada operador ou chamada de função.	–
cpu_tuple_cost	Define a estimativa do planejador sobre o custo do processamento de cada tupla (linha).	–
cron.database_name	Define o banco de dados para armazenar tabelas de metadados pg_cron	postgres
cron.log_run	Registra todos os trabalhos executados na tabela job_run_details	ativado
cron.log_statement	Registre todas as instruções cron antes da execução.	desligar

Nome do parâmetro	Descrição	Padrão
cron.max_running_jobs	O número máximo de trabalhos que podem ser executados simultaneamente.	5
cron.use_background_workers	Habilita operadores em segundo plano para pg_cron	ativado
cursor_tuple_fraction	Define a estimativa do planejador sobre a fração de linhas de um cursor que serão recuperadas.	–
data_directory	Define o diretório de dados do servidor.	/rdsdbdata/db
datestyle	Define o formato de exibição de valores de data e hora.	–
db_user_namespace	Ativa nomes de usuário por banco de dados.	–
deadlock_timeout	(ms) Define o tempo de espera em um bloqueio antes de verificar a existência de um deadlock.	–
debug_pretty_print	Recua exibições de árvores de análise e plano.	–
debug_print_parse	Registra a árvore de análise de cada consulta.	–
debug_print_plan	Registra o plano de execução de cada consulta.	–
debug_print_rewritten	Registra a árvore de análise regravada de cada consulta.	–
default_statistics_target	Define o destino de estatísticas padrão.	–
default_tablespace	Define o espaço de tabela padrão no qual criar tabelas e índices.	–
default_toast_compression	Define o método de compactação padrão para valores compactáveis.	–

Nome do parâmetro	Descrição	Padrão
default_transaction_deferrable	Define o status padrão postergável de novas transações.	–
default_transaction_isolation	Define o nível de isolamento de transação de cada nova transação.	–
default_transaction_read_only	Define o status padrão somente leitura de novas transações.	–
effective_cache_size	(8kB) Define a pressuposição do planejador sobre o tamanho do cache de disco.	SUM(DBInstanceClassMemory/12038,-50003)
effective_io_concurrency	Número de solicitações simultâneas que podem ser manipuladas de forma eficiente pelo subsistema de disco.	–
enable_async_append	Habilita o uso do planejador de planos de anexação assíncronos.	–
enable_bitmapscan	Habilita o uso do planejador de planos de verificação de bitmap.	–
enable_gathermerge	Habilita o uso do planejador de planos de junção de mesclagem.	–
enable_hashagg	Habilita o uso do planejador de planos de agregação em hash.	–
enable_hashjoin	Habilita o uso do planejador de planos de junção hash.	–
enable_incremental_sort	Habilita o uso do planejador de etapas de classificação incremental.	–
enable_indexonlyscan	Habilita o uso do planejador de planos de verificação somente de índice.	–

Nome do parâmetro	Descrição	Padrão
enable_indexscan	Habilita o uso do planejador de planos de verificação de índice.	–
enable_material	Habilita o uso do planejador da materialização.	–
enable_memoize	Habilita o uso do planejador da memoização	–
enable_mergejoin	Habilita o uso do planejador de planos de junção de mesclagem.	–
enable_nestloop	Habilita o uso do planejador de planos de junção de loop aninhado.	–
enable_parallel_append	Habilita o uso do planejador de planos de anexação paralelos.	–
enable_parallel_hash	Habilita o uso do planejador de planos de hash paralelos.	–
enable_partition_pruning	Habilita a remoção de partição em tempo de planejamento e tempo de execução.	–
enable_partitionwise_aggregate	Habilita agregação e agrupamento por partição.	–
enable_partitionwise_join	Habilita a junção por partição.	–
enable_seqscan	Habilita o uso do planejador de planos de verificação sequencial.	–
enable_sort	Habilita o uso do planejador de etapas de classificação explícitas.	–
enable_tidscan	Habilita o uso do planejador de planos de verificação TID.	–

Nome do parâmetro	Descrição	Padrão
escape_string_warning	Avisa sobre escapes de barra invertida (\) em literais de string comuns.	–
exit_on_error	Encerra a sessão em qualquer erro.	–
extra_float_digits	Define o número de dígitos exibidos para valores de ponto flutuante.	–
force_parallel_mode	Força o uso de instalações de consulta paralela.	–
from_collapse_limit	Define o tamanho da lista FROM além do qual subconsultas não são recolhidas.	–
geqo	Permite a otimização de consultas genéticas.	–
geqo_effort	GEQO: esforço é usado para definir o padrão para outros parâmetros GEQO.	–
geqo_generations	GEQO: número de iterações do algoritmo.	–
geqo_pool_size	GEQO: número de indivíduos na população.	–
geqo_seed	GEQO: propagação para seleção de caminho aleatório.	–
geqo_selection_bias	GEQO: pressão seletiva dentro da população.	–
geqo_threshold	Define o limite de itens FROM além do qual o GEQO é usado.	–
gin_fuzzy_search_limit	Define o resultado máximo permitido para pesquisa exata por GIN.	–
gin_pending_list_limit	(kB) Define o tamanho máximo da lista pendente para o índice GIN.	–

Nome do parâmetro	Descrição	Padrão
hash_mem_multiplier	Múltiplos de work_mem para usar em tabelas de hash.	–
hba_file	Define o arquivo de configuração do hba dos servidores.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Permite feedback de um standby a quente para o primário que evitará conflitos de consulta.	ativado
huge_pages	Reduz a sobrecarga quando uma instância de banco de dados está trabalhando com grandes blocos contíguos de memória, como os usados por buffers compartilhados. Ele é ativado por padrão para todas as classes de instância de banco de dados, exceto as classes de instância t3.medium, db.t3.large, db.t4g.medium e db.t4g.large.	ativado
ident_file	Define o arquivo de configuração de ident dos servidores.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Define a duração máxima permitida de qualquer transação inativa.	86400000
idle_session_timeout	Encerra qualquer sessão que tenha ficado ociosa (ou seja, aguardando uma consulta do cliente), mas não em uma transação aberta, por um tempo maior que o especificado.	–
intervalstyle	Define o formato de exibição para valores de intervalo.	–
join_collapse_limit	Define o tamanho da lista FROM além do qual constructos JOIN não são nivelados.	–

Nome do parâmetro	Descrição	Padrão
krb_caseins_users	Define se os nomes de usuário da GSSAPI (Generic Security Service API) devem ser tratados sem distinção entre maiúsculas e minúsculas (true) ou não. Por padrão, esse parâmetro é definido como false, então o Kerberos espera que os nomes de usuário diferenciem maiúsculas e minúsculas. Para obter mais informações, consulte GSSAPI Authentication (Autenticação da GSSAPI) na documentação do PostgreSQL.	false
lc_messages	Define o idioma em que as mensagens são exibidas.	–
lc_monetary	Define a localidade para a formatação de valores monetários.	–
lc_numeric	Define a localidade para a formatação de números.	–
lc_time	Define a localidade para a formatação de valores de data e hora.	–
listen_addresses	Define o nome do host ou os endereços IP para escutar.	*
lo_compat_privileges	Habilita o modo de compatibilidade com versões anteriores para verificações de privilégi os em objetos grandes.	0
log_autovacuum_min_duration	(ms) Define o tempo de execução mínimo acima do qual as ações de autovacuum serão registradas em log.	10000
log_connections	Registra cada conexão bem-sucedida.	–

Nome do parâmetro	Descrição	Padrão
log_destination	Define o destino para a saída de log do servidor.	stderr
log_directory	Define o diretório de destino para arquivos de log.	/rdsdbdata/log/error
log_disconnections	Registra o fim de uma sessão, incluindo a duração.	–
log_duration	Registra a duração de cada instrução SQL concluída.	–
log_error_verbosity	Define a verbosidade das mensagens registradas.	–
log_executor_stats	Grava estatísticas de desempenho do executor no log do servidor.	–
log_file_mode	Define as permissões de arquivo para arquivos de log.	0644
log_filename	Define o padrão de nome de arquivo para arquivos de log.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Inicia um subprocesso para capturar a saída stderr e/ou csvlogs em arquivos de log.	1
log_hostname	Registra o nome do host nos logs de conexão.	0
logical_decoding_work_mem	(kB) Essa quantidade de memória pode ser usada por cada buffer de reordenamento interno antes de espalhar para o disco.	–
log_line_prefix	Controla informações prefixadas para cada linha de log.	%t:%r:%u@%d:%p]:
log_lock_waits	Registra esperas de bloqueio longas.	–

Nome do parâmetro	Descrição	Padrão
log_min_duration_sample	(ms) Define o tempo de execução mínimo acima do qual as instruções serão registradas em log. A amostragem é determinada por log_statement_sample_rate.	–
log_min_duration_statement	(ms) Define o tempo de execução mínimo acima do qual as instruções serão registradas em log.	–
log_min_error_statement	Faz com que todas as instruções que geram um erro igual ou acima desse nível sejam registradas em log.	–
log_min_messages	Define os níveis de mensagem registrados.	–
log_parameter_max_length	(B) Quando as instruções de registro em log limitam os valores de parâmetros registrados em log aos primeiros N bytes.	–
log_parameter_max_length_on_error	(B) Quando o relato de um erro limita os valores de parâmetros registrados em log aos primeiros N bytes.	–
log_parser_stats	Grava estatísticas de desempenho do analisador no log do servidor.	–
log_planner_stats	Grava estatísticas de desempenho do planejador no log do servidor.	–
log_replication_commands	Registra cada comando de replicação.	–
log_rotation_age	(min) A alternância automática de arquivos de log ocorrerá depois de N minutos.	60
log_rotation_size	(kB) A alternância automática de arquivos de log ocorrerá depois de N kilobytes.	100000

Nome do parâmetro	Descrição	Padrão
log_statement	Define o tipo de instruções registradas.	–
log_statement_sample_rate	Fração de instruções que excedem log_min_duration_sample a serem registradas em log.	–
log_statement_stats	Grava estatísticas de desempenho cumulativas no log do servidor.	–
log_temp_files	(kB) Registra o uso de arquivos temporários maiores do que esse número de kilobytes.	–
log_timezone	Define o fuso horário a ser usado em mensagens de log.	UTC
log_transaction_sample_rate	Define a fração de transações para registrar novas transações.	–
log_truncate_on_rotation	Trunca os arquivos de log existentes com o mesmo nome durante a alternância do log.	0
maintenance_io_concurrency	Uma variante de effective_io_concurrency usada para trabalhos de manutenção.	1
maintenance_work_mem	(kB) Define a memória máxima a ser usada para operações de manutenção.	GREATEST(DBInstanceClassMemory/63963136*1024, 65536)
max_connections	Define o número máximo de conexões simultâneas.	LEAST(DBInstanceClassMemory/9531392, 5000)
max_files_per_process	Define o número máximo de arquivos abertos simultaneamente para cada processo do servidor.	–

Nome do parâmetro	Descrição	Padrão
max_locks_per_transaction	Define o número máximo de bloqueios por transação.	64
max_logical_replication_workers	Número máximo de processos do operador de replicação lógica.	–
max_parallel_maintenance_workers	Define o número máximo de processos em paralelo por operação de manutenção.	–
max_parallel_workers	Define o número máximo de operadores em paralelo que podem estar ativos de uma só vez.	GREATEST($DBInstanceVCPU/2, 8)
max_parallel_workers_per_gather	Define o número máximo de processos em paralelo por nó do executor.	–
max_pred_locks_per_page	Define o número máximo de tuplas bloqueadas por predicado por página.	–
max_pred_locks_per_relation	Define o número máximo de páginas e tuplas bloqueadas por predicado por relação.	–
max_pred_locks_per_transaction	Define o número máximo de bloqueios de predicado por transação.	–
max_prepared_transactions	Define o número máximo de transações simultaneamente preparadas.	0
max_replication_slots	Define o número máximo de slots de replicação que o servidor pode suportar.	20
max_slot_wal_keep_size	(MB) Os slots de replicação serão marcados como falhos e segmentos liberados para exclusão ou reciclagem se esse espaço for ocupado pelo WAL no disco.	–

Nome do parâmetro	Descrição	Padrão
max_stack_depth	(kB) Define a profundidade máxima da pilha, em kilobytes.	6144
max_standby_streaming_delay	(ms) Define o atraso máximo antes de cancelar consultas quando um servidor em standby a quente está processando dados do WAL em transmissão.	14000
max_sync_workers_per_subscription	Número máximo de operadores de sincronização por assinatura	2
max_wal_senders	Define o número máximo de processos do remetente WAL em execução simultânea.	10
max_worker_processes	Define o número máximo de processos de operadores simultâneos.	GREATEST($DBInstanceVCPU*2, 8)
min_dynamic_shared_memory	(MB) Quantidade de memória compartilhada dinâmica reservada na inicialização.	–
min_parallel_index_scan_size	(8kB) Define a quantidade mínima de dados de índice para uma varredura paralela.	–
min_parallel_table_scan_size	(8kB) Define a quantidade mínima de dados de tabela para uma varredura paralela.	–
old_snapshot_threshold	(min) O tempo antes de um snapshot ser muito antigo para ler as páginas alteradas depois que o snapshot foi tirado.	–
orafce.nls_date_format	Emula o comportamento de saída de data oracles.	–
orafce.timezone	Especifique o fuso horário utilizado para a função sysdate.	–

Nome do parâmetro	Descrição	Padrão
parallel_leader_participation	Controla se Reunir e Reunir mesclagem também executam subplanos.	–
parallel_setup_cost	Define a estimativa dos planejadores do custo de inicialização de processos de operadores para consulta paralela.	–
parallel_tuple_cost	Define a estimativa dos planejadores do custo de passar cada tupla (linha) do operador para o backend primário.	–
password_encryption	Criptografa senhas.	–
pgaudit.log	Especifica quais classes de instruções serão registradas pelo registro em log de auditoria de sessão.	–
pgaudit.log_catalog	Especifica que o registro em log da sessão deve ser habilitado no caso em que todas as relações em uma instrução estão em pg_catalog.	–
pgaudit.log_level	Especifica o nível de log que será usado para entradas de log.	–
pgaudit.log_parameter	Especifica que o registro em log de auditoria deve incluir os parâmetros que foram passados com a instrução.	–
pgaudit.log_relation	Especifica se o registro em log de auditoria de sessão deve criar uma entrada de log separada para cada relação (TABLE, VIEW etc.) referenciada em uma instrução SELECT ou DML.	–

Nome do parâmetro	Descrição	Padrão
pgaudit.log_statement_once	Especifica se o registro incluirá o texto e os parâmetros da instrução com a primeira entrada de log para uma combinação de instrução/subinstrução ou com cada entrada.	–
pgaudit.role	Especifica a função primária a ser usada para o registro em log de auditoria de objetos.	–
pg_bigm.enable_recheck	Ele especifica se deseja executar a Reverificação, que é um processo interno de pesquisa de texto completo.	ativado
pg_bigm.gin_key_limit	Ele especifica o número máximo de 2 gramas da palavra-chave de pesquisa a ser usada para pesquisa de texto completo.	0
pg_bigm.last_update	Ele relata a última data atualizada do módulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Ele especifica o limite mínimo usado pela pesquisa por similaridade.	0.3
pg_hint_plan.debug_print	Registra os resultados da análise de dicas.	–
pg_hint_plan.enable_hint	Força o planejador a usar planos especificados no comentário de dica anterior à consulta.	–
pg_hint_plan.enable_hint_table	Força o planejador a não receber dicas usando pesquisas de tabela.	–
pg_hint_plan.message_level	Nível da mensagem de mensagens de depuração.	–
pg_hint_plan.parse_messages	Nível de mensagem de erros de análise.	–

Nome do parâmetro	Descrição	Padrão
pglogical.batch_inserts	Inserções em lote, se possível	–
pglogical.conflict_log_level	Define o nível de log usado para registrar em log conflitos resolvidos.	–
pglogical.conflict_resolution	Define o método usado para resolução de conflitos para conflitos resolvíveis.	–
pglogical.extra_connection_options	opções de conexão para adicionar a todas as conexões de nó de pares	–
pglogical.synchronous_commit	Valor de confirmação síncrona específica do pglogical	–
pglogical.use_spi	Usa SPI em vez de API de baixo nível para aplicar alterações	–
pgtle.clientauth_databases_to_skip	Lista de bancos de dados a serem ignorados para o recurso clientauth.	–
pgtle.clientauth_db_name	Controla qual banco de dados é usado para o recurso clientauth.	–
pgtle.clientauth_num_parallel_workers	Número de operadores em segundo plano usados para o recurso clientauth.	–
pgtle.clientauth_users_to_skip	Lista de usuários a serem ignorados para o recurso clientauth.	–
pgtle.enable_clientauth	Ativa o recurso clientauth.	–
pgtle.passcheck_db_name	Define qual banco de dados é usado para o recurso de verificação de acesso em todo o cluster.	–

Nome do parâmetro	Descrição	Padrão
pg_prewarm.autoprewarm	Inicia o trabalhador de reaquecimento automático.	–
pg_prewarm.autoprewarm_interval	Define o intervalo entre despejos de buffers compartilhados	–
pg_similarity.block_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.block_threshold	Define o limite usado pela função de similaridade de bloco.	–
pg_similarity.block_tokenizer	Define o tokenizador para a função de similaridade de bloco.	–
pg_similarity.cosine_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.cosine_threshold	Define o limite usado pela função de similaridade de cosseno.	–
pg_similarity.cosine_tokenizer	Define o tokeniziador para a função de similaridade de cosseno.	–
pg_similarity.dice_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.dice_threshold	Define o limite usado pela medida de similaridade de dados.	–
pg_similarity.dice_tokenizer	Define o tokenizador para a medida de similaridade de dados.	–
pg_similarity.euclidean_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.euclidean_threshold	Define o limite usado pela medida de similaridade euclidiana.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.euclidean_tokenizer	Define o tokenizador para a medida de similaridade euclidiana.	–
pg_similarity.hamming_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.hamming_threshold	Define o limite usado pela métrica de similaridade de bloco.	–
pg_similarity.jaccard_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jaccard_threshold	Define o limite usado pela medida de similaridade Jaccard.	–
pg_similarity.jaccard_tokenizer	Define o tokenizador para a medida de similaridade Jaccard.	–
pg_similarity.jaro_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jaro_threshold	Define o limite usado pela medida de similaridade Jaro.	–
pg_similarity.jarowinkler_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jarowinkler_threshold	Define o limite usado pela medida de similaridade Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.levenshtein_threshold	Define o limite usado pela medida de similaridade Levenshtein.	–
pg_similarity.matching_is_normalized	Define se o valor do resultado é normalizado ou não.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.matching_threshold	Define o limite usado pela medida de similaridade de Coeficiente de correspondência.	–
pg_similarity.matching_tokenizer	Define o tokenizador para a medida de similaridade de Coeficiente de correspondência.	–
pg_similarity.mongeelkan_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.mongeelkan_threshold	Define o limite usado pela medida de similaridade Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Define o tokenizador para a medida de similaridade Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Define a penalidade de lacuna usada pela medida de similaridade Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.nw_threshold	Define o limite usado pela medida de similaridade Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.overlap_threshold	Define o limite usado pela medida de similaridade do Coeficiente de sobreposição.	–
pg_similarity.overlap_tokenizer	Define o tokenizador para a medida de similaridade de coeficientes de sobreposição.	–
pg_similarity.qgram_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.qgram_threshold	Define o limite usado pela medida de similaridade Q-Gram.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.qgram_tokenizer	Define o tokenizador para a medida Q-Gram.	–
pg_similarity.swg_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.swg_threshold	Define o limite usado pela medida de similaridade Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.sw_threshold	Define o limite usado pela medida de similaridade Smith-Waterman.	–
pg_stat_statements.max	Define o número máximo de instruções rastreadas por pg_stat_statements.	–
pg_stat_statements.save	Salva as estatísticas pg_stat_statements em todas os desligamentos do servidor.	–
pg_stat_statements.track	Seleciona quais instruções são rastreadas por pg_stat_statements.	–
pg_stat_statements.track_planning	Seleciona se a duração do planejamento é rastreada por pg_stat_statements.	–
pg_stat_statements.track_utility	Seleciona se os comandos do utilitário são rastreados por pg_stat_statements.	–
plan_cache_mode	Controla a seleção do planejador de plano personalizado ou genérico.	–
porta	Define a porta TCP na qual o servidor escuta.	EndPointPort
postgis.gdal_enabled_drivers	Habilita ou desabilita drivers GDAL usados com PostGIS no Postgres 9.3.5 e posteriores.	ENABLE_ALL

Nome do parâmetro	Descrição	Padrão
quote_all_identifiers	Ao gerar fragmentos SQL, cita todos os identificadores.	–
random_page_cost	Define a estimativa do planejador sobre o custo de uma página de disco não sequencialmente buscada.	–
rdkit.dice_threshold	Limite inferior de similaridade de dados. Moléculas com similaridade menor que o limite não são semelhantes pela operação #.	–
rdkit.do_chiral_sss	A estereoquímica deve ser levada em consideração na correspondência de subestruturas. Se falsa, nenhuma informação estereoquímica é usada em correspondências de subestrutura.	–
rdkit.tanimoto_threshold	Limite inferior de similaridade de Tanimoto. Moléculas com similaridade menor que o limite não são semelhantes pela operação %.	–
rds.accepted_password_auth_method	Forçar a autenticação para conexões com senha armazenada localmente.	md5+scram
rds.adaptive_autovacuum	Parâmetro RDS para habilitar/desabilitar o autovacuum adaptativo.	1
rds.babelfish_status	Parâmetro RDS para habilitar/desabilitar o Babelfish for Aurora PostgreSQL.	desligar
rds.enable_plan_management	Habilita ou desabilita a extensão apg_plan_mgmt.	0

Nome do parâmetro	Descrição	Padrão
rds.extensions	Lista de extensões fornecidas pelo RDS	address_standardizer, address_standardizer_data_us, apg_plan_mgmt, aurora_stat_utils, amcheck, autoinc, aws_commons, aws_ml, aws_s3, aws_lambda, bool_plperl, bloom, btree_gin, btree_gist, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hll, hstore, hstore_plperl, insert_username, intagg, intarray, ip4r, isn, jsonb_plperl, lo, log_fdw, ltree, moddatetime, old_snapshot, oracle_fdw, orafce, pgaudit, pgcrypto, pglogical, pgrouting, pgrowlocks, pgstattuple, pgtap, pg_bigm, pg_buffercache, pg_cron, pg_freespacemap, pg_hint_plan, pg_partman, pg_prewarm, pg_proctab, pg_repack, pg_simila

Nome do parâmetro	Descrição	Padrão
		rity, pg_stat_s tatements, pg_trgm, pg_visibility, plcoffee, plls, plperl, plpgsql, plprofiler, pltcl, plv8, postgis, postgis_t iger_geocoder, postgis_raster, postgis_topology, postgres_fdw, prefix, rdkit, rds_tools, refint, sslinfo, tablefunc, tds_fdw, test_parser, tsm_system_rows, tsm_system_time, unaccent, uuid-ossp
rds.force_admin_lo gging_level	Consulta mensagens de log para ações de usuário administrador do RDS em bancos de dados de clientes.	–
rds.force_autovacu um_logging_level	Consulta mensagens de log relacionadas às operações de autovacuum.	WARNING
rds.force_ssl	Força conexões SSL.	0

Nome do parâmetro	Descrição	Padrão
rds.global_db_rpo	(s) Limite do objetivo do ponto de recuperação em segundos que bloqueia confirmações do usuário quando ele é violado. **Important** Esse parâmetro é destinado aos bancos de dados globais baseados no Aurora PostgreSQL. Para um banco de dados não global, deixe-o com o valor padrão. Para obter mais informaçõ es sobre como usar esse parâmetro, consulte the section called “Gerenciar RPOs (Aurora PostgreSQL)”.	–
rds.logical_replication	Permite decodificação lógica.	0
rds.logically_repl icate_unlogged_tables	As tabelas não registradas em log são replicadas logicamente.	1
rds.log_retention_ period	O Amazon RDS excluirá o log do PostgreSQL com mais de N minutos.	4320
rds.pg_stat_ramdis k_size	Tamanho das estatísticas ramdisk em MB. Um valor diferente de zero configurará o ramdisk. Esse parâmetro só está disponível no Aurora PostgreSQL 14 e versões anteriores.	0
rds.rds_superuser_ reserved_connections	Define o número de slots de conexão reservados para rds_superusers. Esse parâmetro só está disponível nas versões 15 e posterior. Para ter mais informações, consulte reserved connections na documentação do PostgreSQL.	2

Nome do parâmetro	Descrição	Padrão
rds.restrict_password_commands	Restringe comandos relacionados a senha a membros do rds_password	–
rds.superuser_variables	Lista de variáveis somente superusuário para as quais elevamos as instruções de modificação rds_superuser.	session_replication_role
recovery_init_sync_method	Define o método para sincronizar o diretório de dados antes da recuperação de falhas.	syncfs
remove_temp_files_after_crash	Remove arquivos temporários após falha de backend.	0
restart_after_crash	Reinicializa o servidor após falha de backend.	–
row_security	Habilita a segurança de linha.	–
search_path	Define a ordem de pesquisa do esquema de nomes que não são qualificados para esquema.	–
seq_page_cost	Define a estimativa do planejador sobre o custo de uma página de disco sequencialmente buscada.	–
session_replication_role	Define o comportamento de sessões para disparadores e regras de regravação.	–
shared_buffers	(8kB) Define o número de buffers de memória compartilhada usados pelo servidor.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Lista bibliotecas compartilhadas para pré-carregar no servidor.	pg_stat_statements
ssl	Habilita conexões SSL.	1

Nome do parâmetro	Descrição	Padrão
ssl_ca_file	Localização do arquivo de autoridade do servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Localização do arquivo de certificado do servidor SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_ciphers	Define a lista de criptografias TLS permitidas em conexões seguras.	–
ssl_crl_dir	Localização do diretório da lista de revogação de certificados SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Localização do arquivo de chave privada do servidor SSL	/rdsdbdata/rds-metadata/server-key.pem
ssl_max_protocol_version	Define a versão máxima do protocolo SSL/TLS permitida	–
ssl_min_protocol_version	Define a versão mínima do protocolo SSL/TLS permitida	TLSv1.2
standard_conforming_strings	Faz com que strings ... tratem barras invertidas literalmente.	–
statement_timeout	(ms) Define a duração máxima permitida de qualquer instrução.	–
stats_temp_directory	Grava arquivos de estatísticas temporárias no diretório especificado.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Define o número de slots de conexão reservados para superusuários.	3
synchronize_seqscans	Habilita varreduras sequenciais sincronizadas.	–
synchronous_commit	Define o nível de sincronização de transações atual.	ativado

Nome do parâmetro	Descrição	Padrão
tcp_keepalives_count	Número máximo de retransmissões de keepalives TCP.	–
tcp_keepalives_idle	(s) Tempo entre a emissão de keepalives TCP.	–
tcp_keepalives_interval	(s) Tempo entre retransmissões de keepalives TCP.	–
temp_buffers	(8kB) Define o número máximo de buffers temporários usado por cada sessão.	–
temp_file_limit	Restringe a quantidade total de espaço em disco em kilobytes que um determinado processo PostgreSQL pode usar para arquivos temporários, excluindo o espaço usado para tabelas temporárias explícitas	-1
temp_tablespaces	Define os espaços de tabela a serem usados para tabelas temporárias e arquivos de classificação.	–
timezone	Define o fuso horário para exibir e interpretar carimbos de data/hora.	UTC
track_activities	Coleta informações sobre a execução de comandos.	–
track_activity_query_size	Define o tamanho reservado para pg_stat_activity.current_query, em bytes.	4096
track_commit_timestamp	Coleta o tempo de confirmação da transação.	–
track_counts	Coleta estatísticas sobre a atividade do banco de dados.	–

Nome do parâmetro	Descrição	Padrão
track_functions	Coleta estatísticas em nível de função sobre a atividade do banco de dados.	pl
track_io_timing	Coleta estatísticas de tempo sobre atividades de E/S do banco de dados.	1
track_wal_io_timing	Coleta estatísticas de tempo da atividade de WAL.	–
transform_null_equals	Trata expr=NULL como expr IS NULL.	–
update_process_title	Atualiza o título do processo para mostrar o comando SQL ativo.	–
vacuum_cost_delay	(ms) Atraso de custo de vacuum em milissegundos.	–
vacuum_cost_limit	Valor do custo de vacuum disponível antes da suspensão.	–
vacuum_cost_page_dirty	Custo de vacuum para uma página suja por vacuum.	–
vacuum_cost_page_hit	Custo de vacuum para uma página encontrada no cache do buffer.	–
vacuum_cost_page_miss	Custo de vacuum para uma página não encontrada no cache do buffer.	0
vacuum_defer_cleanup_age	Número de transações pelas quais a limpeza VACUUM e HOT deve ser adiada, se houver.	–
vacuum_failsafe_age	Idade na qual o VACUUM deve ser acionado à prova de falha para evitar uma interrupção de wraparound.	1200000000
vacuum_freeze_min_age	Idade mínima na qual o VACUUM deve congelar uma linha de tabela.	–

Nome do parâmetro	Descrição	Padrão
vacuum_freeze_table_age	Idade na qual o VACUUM deve varrer uma tabela inteira para congelar tuplas.	–
vacuum_multixact_failsafe_age	Idade Multixact na qual o VACUUM deve ser acionado à prova de falha para evitar uma interrupção de wraparound.	1200000000
vacuum_multixact_freeze_min_age	Idade mínima na qual o VACUUM deve congelar uma linha de tabela MultiXactId.	–
vacuum_multixact_freeze_table_age	Idade multiexact na qual o VACUUM deve varrer uma tabela inteira para congelar tuplas.	–
wal_buffers	(8 kB) Define o número de buffers da página de disco na memória compartilhada para WAL.	–
wal_receiver_create_temp_slot	Define se um receptor WAL deve criar um slot de replicação temporário se nenhum slot permanente estiver configurado.	0
wal_receiver_status_interval	(s) Define o intervalo máximo entre os relatórios de status do receptor WAL para o primário.	–
wal_receiver_timeout	(ms) Define o tempo máximo de espera para receber dados do primário.	30000
wal_sender_timeout	(ms) Define o tempo máximo para aguardar a replicação do WAL.	–
work_mem	(kB) Define o máximo de memória a ser usada para espaços de trabalho de consulta.	–
xmlbinary	Define como valores binários devem ser codificados em XML.	–

Nome do parâmetro	Descrição	Padrão
xmloption	Define se dados XML em operações de análise e serialização implícitas são considerados documentos ou fragmentos de conteúdo.	–

Parâmetros no nível da instância do Aurora PostgreSQL

É possível visualizar parâmetros no nível da instância disponíveis para uma versão do Aurora PostgreSQL usando o Console de Gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS. Para obter informações sobre como visualizar os parâmetros em grupos de parâmetros de banco de dados do Aurora PostgreSQL no console do RDS, consulte Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados no Amazon Aurora.

Alguns parâmetros no nível de instância não estão disponíveis em todas as versões e alguns estão sendo descontinuados. Para obter informações sobre como visualizar os parâmetros de uma versão específica do Aurora PostgreSQL, consulte Exibindo parâmetros de banco de dados e de cluster de bancos de dados Aurora PostgreSQL.

Por exemplo, a tabela a seguir lista os parâmetros que se aplicam a uma instância de banco de dados específica em um cluster de bancos de dados do Aurora PostgreSQL. Esta lista foi gerada executando o comando da AWS CLI describe-db-parameters com `default.aurora-postgresql14` para o valor `--db-parameter-group-name`.

Para obter uma lista dos parâmetros de cluster de banco de dados para o mesmo grupo de parâmetros padrão do banco de dados, consulte Parâmetros no nível do cluster do Aurora PostgreSQL.

Nome do parâmetro	Descrição	Padrão
apg_enable_batch_mode_function_execution	Permite que as funções em modo em lote processem conjuntos de linhas de cada vez.	–
apg_enable_correlated_any_transform	Permite que o planejador transforme ANY Sublink correlacionado (subconsulta IN/NOT IN) para JOIN quando possível.	–
apg_enable_function_migration	Permite que o planejador migre funções escalares elegíveis para a cláusula FROM.	–
apg_enable_not_in_transform	Permite que o planejador transforme a subconsulta NOT IN em ANTI JOIN quando possível.	–

Nome do parâmetro	Descrição	Padrão
apg_enable_remove_redundant_inner_joins	Permite que o planejador remova junções internas redundantes.	–
apg_enable_semijoin_push_down	Permite o uso de filtros de semijunção para junções de hash.	–
apg_plan_mgmt.capture_plan_baselines	Modo de linha de base do plano de captura. manual: habilita a captura de plano para qualquer instrução SQL, desativado: desabilita captura de plano, automático: habilita a captura de plano para instruções em pg_stat_statement que atendam aos critérios de elegibilidade.	desligar
apg_plan_mgmt.max_databases	Define o número máximo de bancos de dados que podem gerenciar consultas usando apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Define o número máximo de planos que podem ser armazenados em cache por apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Número máximo de dias desde que um plano foi last_used antes de um plano ser excluído automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Custo total estimado do plano abaixo do qual um plano não aprovado será executado.	0
apg_plan_mgmt.use_plan_baselines	Usa somente planos aprovados ou fixos para instruções gerenciadas.	false
application_name	Define o nome da aplicação a ser informada em estatísticas e logs.	–

Nome do parâmetro	Descrição	Padrão
aurora_compute_plan_id	Monitora planos de execução de consultas para detectar planos de execução que contribuem com a carga de banco de dados atual e para monitorar as estatísticas de desempenho de planos de execução ao longo do tempo. Consulte mais informações em Monitoring query execution plans for Aurora PostgreSQL.	ativado
aurora_temp_space_size	(MB) Define o tamanho do espaço alocado para objetos temporários habilitados para leituras otimizadas em clusters do Aurora otimizados para E/S com classes de instância compatíveis.	DBInstanceClassMemory/524288
authentication_timeout	(s) Define o tempo máximo permitido para concluir a autenticação de cliente.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE para registro em log de planos.	–
auto_explain.log_buffers	Uso de buffers de log.	–
auto_explain.log_format	Formato EXPLAIN a ser usado para registro em log de planos.	–
auto_explain.log_min_duration	Define o tempo de execução mínimo acima do qual as instruções serão registradas em log.	–
auto_explain.log_nested_statements	Registrar instruções aninhadas.	–
auto_explain.log_timing	Coleta dados de tempo, não apenas contagens de linhas.	–

Nome do parâmetro	Descrição	Padrão
auto_explain.log_triggers	Inclui estatísticas do acionador nos planos.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE para registro em log de planos.	–
auto_explain.sample_rate	Fração de consultas a serem processadas.	–
babelfishpg_tds.listen_addresses	Define o nome do host ou um ou mais endereços IP para ouvir o TDS.	*
babelfishpg_tds.tds_debug_log_level	Define o nível de registro em log no TDS, 0 desabilita o registro em log	1
backend_flush_after	(8 Kb) Número de páginas após as quais as gravações executadas anteriormente são liberadas para o disco.	–
bytea_output	Define o formato da saída para bytes.	–
check_function_bodies	Verifica corpos de funções durante CREATE FUNCTION.	–
client_connection_check_interval	Define o intervalo de tempo entre as verificações de desconexão durante a execução de consultas.	–
client_min_messages	Define os níveis de mensagem enviados ao cliente.	–
config_file	Define o arquivo de configuração principal dos servidores.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Permite que o planejador use restrições para otimizar consultas.	–

Nome do parâmetro	Descrição	Padrão
cpu_index_tuple_cost	Define a estimativa do planejador sobre o custo do processamento de cada entrada de índice durante uma verificação de índice.	–
cpu_operator_cost	Define a estimativa do planejador sobre o custo do processamento de cada operador ou chamada de função.	–
cpu_tuple_cost	Define a estimativa do planejador sobre o custo do processamento de cada tupla (linha).	–
cron.database_name	Define o banco de dados para armazenar tabelas de metadados pg_cron	postgres
cron.log_run	Registra todos os trabalhos executados na tabela job_run_details	ativado
cron.log_statement	Registre todas as instruções cron antes da execução.	desligar
cron.max_running_jobs	O número máximo de trabalhos que podem ser executados simultaneamente.	5
cron.use_background_workers	Habilita operadores em segundo plano para pg_cron	ativado
cursor_tuple_fraction	Define a estimativa do planejador sobre a fração de linhas de um cursor que serão recuperadas.	–
db_user_namespace	Ativa nomes de usuário por banco de dados.	–
deadlock_timeout	(ms) Define o tempo de espera em um bloqueio antes de verificar a existência de um deadlock.	–
debug_pretty_print	Recua exibições de árvores de análise e plano.	–
debug_print_parse	Registra a árvore de análise de cada consulta.	–

Nome do parâmetro	Descrição	Padrão
debug_print_plan	Registra o plano de execução de cada consulta.	–
debug_print_rewritten	Registra a árvore de análise regravada de cada consulta.	–
default_statistics_target	Define o destino de estatísticas padrão.	–
default_transaction_deferrable	Define o status padrão postergável de novas transações.	–
default_transaction_isolation	Define o nível de isolamento de transação de cada nova transação.	–
default_transaction_read_only	Define o status padrão somente leitura de novas transações.	–
effective_cache_size	(8 kB) Define a pressuposição do planejador sobre o tamanho do cache de disco.	SUM(DBInstanceClassMemory/12038,-50003
effective_io_concurrency	Número de solicitações simultâneas que podem ser manipuladas de forma eficiente pelo subsistema de disco.	–
enable_async_append	Habilita o uso do planejador de planos de anexação assíncronos.	–
enable_bitmapscan	Habilita o uso do planejador de planos de verificação de bitmap.	–
enable_gathermerge	Habilita o uso do planejador de planos de junção de mesclagem.	–
enable_hashagg	Habilita o uso do planejador de planos de agregação em hash.	–

Nome do parâmetro	Descrição	Padrão
enable_hashjoin	Habilita o uso do planejador de planos de junção hash.	–
enable_incremental_sort	Habilita o uso do planejador de etapas de classificação incremental.	–
enable_indexonlyscan	Habilita o uso do planejador de planos de verificação somente de índice.	–
enable_indexscan	Habilita o uso do planejador de planos de verificação de índice.	–
enable_material	Habilita o uso do planejador da materialização.	–
enable_memoize	Habilita o uso do planejador da memoização	–
enable_mergejoin	Habilita o uso do planejador de planos de junção de mesclagem.	–
enable_nestloop	Habilita o uso do planejador de planos de junção de loop aninhado.	–
enable_parallel_append	Habilita o uso do planejador de planos de anexação paralelos.	–
enable_parallel_hash	Habilita o uso do planejador de planos de hash paralelos.	–
enable_partition_pruning	Habilita a remoção de partição em tempo de planejamento e tempo de execução.	–
enable_partitionwise_aggregate	Habilita agregação e agrupamento por partição.	–
enable_partitionwise_join	Habilita a junção por partição.	–

Nome do parâmetro	Descrição	Padrão
enable_seqscan	Habilita o uso do planejador de planos de verificação sequencial.	–
enable_sort	Habilita o uso do planejador de etapas de classificação explícitas.	–
enable_tidscan	Habilita o uso do planejador de planos de verificação TID.	–
escape_string_warning	Avisa sobre escapes de barra invertida (\) em literais de string comuns.	–
exit_on_error	Encerra a sessão em qualquer erro.	–
force_parallel_mode	Força o uso de instalações de consulta paralela.	–
from_collapse_limit	Define o tamanho da lista FROM além do qual subconsultas não são recolhidas.	–
geqo	Permite a otimização de consultas genéticas.	–
geqo_effort	GEQO: esforço é usado para definir o padrão para outros parâmetros GEQO.	–
geqo_generations	GEQO: número de iterações do algoritmo.	–
geqo_pool_size	GEQO: número de indivíduos na população.	–
geqo_seed	GEQO: propagação para seleção de caminho aleatório.	–
geqo_selection_bias	GEQO: pressão seletiva dentro da população.	–
geqo_threshold	Define o limite de itens FROM além do qual o GEQO é usado.	–

Nome do parâmetro	Descrição	Padrão
gin_fuzzy_search_limit	Define o resultado máximo permitido para pesquisa exata por GIN.	–
gin_pending_list_limit	(kB) Define o tamanho máximo da lista pendente para o índice GIN.	–
hash_mem_multiplier	Múltiplos de work_mem para usar em tabelas de hash.	–
hba_file	Define o arquivo de configuração do hba dos servidores.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Permite feedback de um standby a quente para o primário que evitará conflitos de consulta.	ativado
ident_file	Define o arquivo de configuração de ident dos servidores.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Define a duração máxima permitida de qualquer transação inativa.	86400000
idle_session_timeout	Encerra qualquer sessão que tenha ficado ociosa (ou seja, aguardando uma consulta do cliente), mas não em uma transação aberta por um tempo maior que o especificado.	–
join_collapse_limit	Define o tamanho da lista FROM além do qual constructos JOIN não são nivelados.	–
lc_messages	Define o idioma em que as mensagens são exibidas.	–
listen_addresses	Define o nome do host ou os endereços IP para escutar.	*

Nome do parâmetro	Descrição	Padrão
lo_compat_privileges	Habilita o modo de compatibilidade com versões anteriores para verificações de privilégios em objetos grandes.	0
log_connections	Registra cada conexão bem-sucedida.	–
log_destination	Define o destino para a saída de log do servidor.	stderr
log_directory	Define o diretório de destino para arquivos de log.	/rdsdbdata/log/error
log_disconnections	Registra o fim de uma sessão, incluindo a duração.	–
log_duration	Registra a duração de cada instrução SQL concluída.	–
log_error_verbosity	Define a verbosidade das mensagens registradas.	–
log_executor_stats	Grava estatísticas de desempenho do executor no log do servidor.	–
log_file_mode	Define as permissões de arquivo para arquivos de log.	0644
log_filename	Define o padrão de nome de arquivo para arquivos de log.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Inicia um subprocesso para capturar a saída stderr e/ou csvlogs em arquivos de log.	1
log_hostname	Registra o nome do host nos logs de conexão.	0
logical_decoding_work_mem	(kB) Essa quantidade de memória pode ser usada por cada buffer de reordenamento interno antes de espalhar para o disco.	–

Nome do parâmetro	Descrição	Padrão
log_line_prefix	Controla informações prefixadas para cada linha de log.	%t:%r:%u@%d:%p]:
log_lock_waits	Registra esperas de bloqueio longas.	–
log_min_duration_sample	(ms) Define o tempo de execução mínimo acima do qual as declarações serão registradas em log. A amostragem é determinada por log_statement_sample_rate.	–
log_min_duration_statement	(ms) Define o tempo de execução mínimo acima do qual as declarações serão registradas em log.	–
log_min_error_statement	Faz com que todas as instruções que geram um erro igual ou acima desse nível sejam registradas em log.	–
log_min_messages	Define os níveis de mensagem registrados.	–
log_parameter_max_length	(B) Quando as declarações de registro em log limitam os valores de parâmetros registrados em log aos primeiros N bytes.	–
log_parameter_max_length_on_error	(B) Quando o relato de um erro limita os valores de parâmetros registrados em log aos primeiros N bytes.	–
log_parser_stats	Grava estatísticas de desempenho do analisador no log do servidor.	–
log_planner_stats	Grava estatísticas de desempenho do planejador no log do servidor.	–
log_replication_commands	Registra cada comando de replicação.	–

Nome do parâmetro	Descrição	Padrão
log_rotation_age	(min) A alternância automática de arquivos de log ocorrerá depois de N minutos.	60
log_rotation_size	(kB) A alternância automática de arquivos de log ocorrerá depois de N kilobytes.	100000
log_statement	Define o tipo de instruções registradas.	–
log_statement_sample_rate	Fração de instruções que excedem log_min_duration_sample a serem registradas em log.	–
log_statement_stats	Grava estatísticas de desempenho cumulativas no log do servidor.	–
log_temp_files	(kB) Registra o uso de arquivos temporários maiores do que esse número de kilobytes.	–
log_timezone	Define o fuso horário a ser usado em mensagens de log.	UTC
log_truncate_on_rotation	Trunca os arquivos de log existentes com o mesmo nome durante a alternância do log.	0
maintenance_io_concurrency	Uma variante de effective_io_concurrency usada para trabalhos de manutenção.	1
maintenance_work_mem	(kB) Define a memória máxima a ser usada para operações de manutenção.	GREATEST(DBInstanceClassMemory/63963136*1024, 65536
max_connections	Define o número máximo de conexões simultâneas.	LEAST(DBInstanceClassMemory/9531392, 5000

Nome do parâmetro	Descrição	Padrão
max_files_per_process	Define o número máximo de arquivos abertos simultaneamente para cada processo do servidor.	–
max_locks_per_transaction	Define o número máximo de bloqueios por transação.	64
max_parallel_maintenance_workers	Define o número máximo de processos em paralelo por operação de manutenção.	–
max_parallel_workers	Define o número máximo de operadores em paralelo que podem estar ativos de uma só vez.	GREATEST($DBInstanceVCPU/2, 8
max_parallel_workers_per_gather	Define o número máximo de processos em paralelo por nó do executor.	–
max_pred_locks_per_page	Define o número máximo de tuplas bloqueadas por predicado por página.	–
max_pred_locks_per_relation	Define o número máximo de páginas e tuplas bloqueadas por predicado por relação.	–
max_pred_locks_per_transaction	Define o número máximo de bloqueios de predicado por transação.	–
max_slot_wal_keep_size	(MB) Os slots de replicação serão marcados como falhos e segmentos liberados para exclusão ou reciclagem se esse espaço for ocupado pelo WAL no disco.	–
max_stack_depth	(kB) Define a profundidade máxima da pilha, em kilobytes.	6144

Nome do parâmetro	Descrição	Padrão
max_standby_streaming_delay	(ms) Define o atraso máximo antes de cancelar consultas quando um servidor em standby a quente está processando dados do WAL em transmissão.	14000
max_worker_processes	Define o número máximo de processos de operadores simultâneos.	GREATEST($DBInstanceVCPU*2, 8
min_dynamic_shared_memory	(MB) Quantidade de memória compartilhada dinâmica reservada na inicialização.	–
min_parallel_index_scan_size	(8kB) Define a quantidade mínima de dados de índice para uma varredura paralela.	–
min_parallel_table_scan_size	(8kB) Define a quantidade mínima de dados de tabela para uma varredura paralela.	–
old_snapshot_threshold	(min) O tempo antes de um snapshot ser muito antigo para ler as páginas alteradas depois que o snapshot foi tirado.	–
parallel_leader_participation	Controla se Reunir e Reunir mesclagem também executam subplanos.	–
parallel_setup_cost	Define a estimativa dos planejadores do custo de inicialização de processos de operadores para consulta paralela.	–
parallel_tuple_cost	Define a estimativa dos planejadores do custo de passar cada tupla (linha) do operador para o backend primário.	–
pgaudit.log	Especifica quais classes de instruções serão registradas pelo registro em log de auditoria de sessão.	–

Nome do parâmetro	Descrição	Padrão
pgaudit.log_catalog	Especifica que o registro em log da sessão deve ser habilitado no caso em que todas as relações em uma instrução estão em pg_catalog.	–
pgaudit.log_level	Especifica o nível de log que será usado para entradas de log.	–
pgaudit.log_parameter	Especifica que o registro em log de auditoria deve incluir os parâmetros que foram passados com a instrução.	–
pgaudit.log_relation	Especifica se o registro em log de auditoria de sessão deve criar uma entrada de log separada para cada relação (TABLE, VIEW etc.) referenciada em uma declaração SELECT ou DML.	–
pgaudit.log_statement_once	Especifica se o registro incluirá o texto e os parâmetros da instrução com a primeira entrada de log para uma combinação de instrução/subinstrução ou com cada entrada.	–
pgaudit.role	Especifica a função primária a ser usada para o registro em log de auditoria de objetos.	–
pg_bigm.enable_recheck	Ele especifica se deseja executar a Reverificação, que é um processo interno de pesquisa de texto completo.	ativado
pg_bigm.gin_key_limit	Ele especifica o número máximo de 2 gramas da palavra-chave de pesquisa a ser usada para pesquisa de texto completo.	0
pg_bigm.last_update	Ele relata a última data atualizada do módulo pg_bigm.	2013.11.22

Nome do parâmetro	Descrição	Padrão
pg_bigm.similarity_limit	Ele especifica o limite mínimo usado pela pesquisa por similaridade.	0.3
pg_hint_plan.debug_print	Registra os resultados da análise de dicas.	–
pg_hint_plan.enable_hint	Força o planejador a usar planos especificados no comentário de dica anterior à consulta.	–
pg_hint_plan.enable_hint_table	Força o planejador a não receber dicas usando pesquisas de tabela.	–
pg_hint_plan.message_level	Nível da mensagem de mensagens de depuração.	–
pg_hint_plan.parse_messages	Nível de mensagem de erros de análise.	–
pglogical.batch_inserts	Inserções em lote, se possível	–
pglogical.conflict_log_level	Define o nível de log usado para registrar em log conflitos resolvidos.	–
pglogical.conflict_resolution	Define o método usado para resolução de conflitos para conflitos resolvíveis.	–
pglogical.extra_connection_options	opções de conexão para adicionar a todas as conexões de nó de pares	–
pglogical.synchronous_commit	Valor de confirmação síncrona específica do pglogical	–
pglogical.use_spi	Usa SPI em vez de API de baixo nível para aplicar alterações	–
pg_similarity.block_is_normalized	Define se o valor do resultado é normalizado ou não.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.block_threshold	Define o limite usado pela função de similaridade de bloco.	–
pg_similarity.block_tokenizer	Define o tokenizador para a função de similaridade de bloco.	–
pg_similarity.cosine_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.cosine_threshold	Define o limite usado pela função de similaridade de cosseno.	–
pg_similarity.cosine_tokenizer	Define o tokeniziador para a função de similaridade de cosseno.	–
pg_similarity.dice_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.dice_threshold	Define o limite usado pela medida de similaridade de dados.	–
pg_similarity.dice_tokenizer	Define o tokenizador para a medida de similaridade de dados.	–
pg_similarity.euclidean_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.euclidean_threshold	Define o limite usado pela medida de similaridade euclidiana.	–
pg_similarity.euclidean_tokenizer	Define o tokenizador para a medida de similaridade euclidiana.	–
pg_similarity.hamming_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.hamming_threshold	Define o limite usado pela métrica de similaridade de bloco.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.jaccard_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jaccard_threshold	Define o limite usado pela medida de similaridade Jaccard.	–
pg_similarity.jaccard_tokenizer	Define o tokenizador para a medida de similaridade Jaccard.	–
pg_similarity.jaro_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jaro_threshold	Define o limite usado pela medida de similaridade Jaro.	–
pg_similarity.jarowinkler_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.jarowinkler_threshold	Define o limite usado pela medida de similaridade Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.levenshtein_threshold	Define o limite usado pela medida de similaridade Levenshtein.	–
pg_similarity.matching_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.matching_threshold	Define o limite usado pela medida de similaridade de Coeficiente de correspondência.	–
pg_similarity.matching_tokenizer	Define o tokenizador para a medida de similaridade de Coeficiente de correspondência.	–
pg_similarity.mongeelkan_is_normalized	Define se o valor do resultado é normalizado ou não.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.mongeelkan_threshold	Define o limite usado pela medida de similaridade Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Define o tokenizador para a medida de similaridade Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Define a penalidade de lacuna usada pela medida de similaridade Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.nw_threshold	Define o limite usado pela medida de similaridade Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.overlap_threshold	Define o limite usado pela medida de similaridade do Coeficiente de sobreposição.	–
pg_similarity.overlap_tokenizer	Define o tokenizador para a medida de similaridade de coeficientes de sobreposição.	–
pg_similarity.qgram_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.qgram_threshold	Define o limite usado pela medida de similaridade Q-Gram.	–
pg_similarity.qgram_tokenizer	Define o tokenizador para a medida Q-Gram.	–
pg_similarity.swg_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.swg_threshold	Define o limite usado pela medida de similaridade Smith-Waterman-Gotoh.	–

Nome do parâmetro	Descrição	Padrão
pg_similarity.sw_is_normalized	Define se o valor do resultado é normalizado ou não.	–
pg_similarity.sw_threshold	Define o limite usado pela medida de similaridade Smith-Waterman.	–
pg_stat_statements.max	Define o número máximo de instruções rastreadas por pg_stat_statements.	–
pg_stat_statements.save	Salva as estatísticas pg_stat_statements em todas os desligamentos do servidor.	–
pg_stat_statements.track	Seleciona quais instruções são rastreadas por pg_stat_statements.	–
pg_stat_statements.track_planning	Seleciona se a duração do planejamento é rastreada por pg_stat_statements.	–
pg_stat_statements.track_utility	Seleciona se os comandos do utilitário são rastreados por pg_stat_statements.	–
postgis.gdal_enabled_drivers	Habilita ou desabilita drivers GDAL usados com PostGIS no Postgres 9.3.5 e posteriores.	ENABLE_ALL
quote_all_identifiers	Ao gerar fragmentos SQL, cita todos os identificadores.	–
random_page_cost	Define a estimativa do planejador sobre o custo de uma página de disco não sequencialmente buscada.	–

Nome do parâmetro	Descrição	Padrão
rds.enable_memory_management	Melhora os recursos de gerenciamento de memória no Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 e versões posteriores, evitando problemas de estabilidade e reinicializações do banco de dados decorrentes de memória livre insuficiente. Para obter mais informaçõ es, consulte Gerenciamento aprimorado de memória no Aurora PostgreSQL.	Verdadeiro
rds.force_admin_logging_level	Consulta mensagens de log para ações de usuário administrador do RDS em bancos de dados de clientes.	–
rds.log_retention_period	O Amazon RDS excluirá o log do PostgreSQL com mais de N minutos.	4320
rds.memory_allocation_guard	Melhora os recursos de gerenciamento de memória no Aurora PostgreSQL 11.21, 12.16, 13.12, 14.9, 15.4 e versões anteriores, evitando problemas de estabilidade e reinicializações do banco de dados decorrentes de memória livre insuficiente. Para obter mais informaçõ es, consulte Gerenciamento aprimorado de memória no Aurora PostgreSQL.	Falso
rds.pg_stat_ramdisk_size	Tamanho das estatísticas ramdisk em MB. Um valor diferente de zero configurará o ramdisk.	0
rds.rds_superuser_reserved_connections	Define o número de slots de conexão reservados para rds_superusers. Esse parâmetro só está disponível nas versões 15 e posterior. Para ter mais informações, consulte reserved connections na documentação do PostgreSQL.	2

Nome do parâmetro	Descrição	Padrão
rds.superuser_variables	Lista de variáveis somente superusuário para as quais elevamos as instruções de modificação rds_superuser.	session_replication_role
remove_temp_files_after_crash	Remove arquivos temporários após falha de backend.	0
restart_after_crash	Reinicializa o servidor após falha de backend.	–
row_security	Habilita a segurança de linha.	–
search_path	Define a ordem de pesquisa do esquema de nomes que não são qualificados para esquema.	–
seq_page_cost	Define a estimativa do planejador sobre o custo de uma página de disco sequencialmente buscada.	–
session_replication_role	Define o comportamento de sessões para disparadores e regras de regravação.	–
shared_buffers	(8kB) Define o número de buffers de memória compartilhada usados pelo servidor.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Lista bibliotecas compartilhadas para pré-carregar no servidor.	pg_stat_statements
ssl_ca_file	Localização do arquivo de autoridade do servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Localização do arquivo de certificado do servidor SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_crl_dir	Localização do diretório da lista de revogação de certificados SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/

Nome do parâmetro	Descrição	Padrão
ssl_key_file	Localização do arquivo de chave privada do servidor SSL	/rdsdbdata/rds-metadata/server-key.pem
standard_conforming_strings	Faz com que strings ... tratem barras invertidas literalmente.	–
statement_timeout	(ms) Define a duração máxima permitida de qualquer declaração.	–
stats_temp_directory	Grava arquivos de estatísticas temporárias no diretório especificado.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Define o número de slots de conexão reservados para superusuários.	3
synchronize_seqscans	Habilita varreduras sequenciais sincronizadas.	–
tcp_keepalives_count	Número máximo de retransmissões de keepalives TCP.	–
tcp_keepalives_idle	(s) Tempo entre a emissão de keepalives TCP.	–
tcp_keepalives_interval	(s) Tempo entre retransmissões de keepalives TCP.	–
temp_buffers	(8kB) Define o número máximo de buffers temporários usado por cada sessão.	–
temp_file_limit	Restringe a quantidade total de espaço em disco em kilobytes que um determinado processo PostgreSQL pode usar para arquivos temporários, excluindo o espaço usado para tabelas temporárias explícitas	-1

Nome do parâmetro	Descrição	Padrão
temp_tablespaces	Define os espaços de tabela a serem usados para tabelas temporárias e arquivos de classific ação.	–
track_activities	Coleta informações sobre a execução de comandos.	–
track_activity_que ry_size	Define o tamanho reservado para pg_stat_a ctivity.current_query, em bytes.	4096
track_counts	Coleta estatísticas sobre a atividade do banco de dados.	–
track_functions	Coleta estatísticas em nível de função sobre a atividade do banco de dados.	pl
track_io_timing	Coleta estatísticas de tempo sobre atividades de E/S do banco de dados.	1
transform_–_equals	Trata expr=– como expr IS –.	–
update_process_title	Atualiza o título do processo para mostrar o comando SQL ativo.	–
wal_receiver_statu s_interval	(s) Define o intervalo máximo entre os relatório s de status do receptor WAL para o primário.	–
work_mem	(kB) Define o máximo de memória a ser usada para espaços de trabalho de consulta.	–
xmlbinary	Define como valores binários devem ser codificados em XML.	–
xmloption	Define se dados XML em operações de análise e serialização implícitas são considerados documentos ou fragmentos de conteúdo.	–

Eventos de espera do Amazon Aurora PostgreSQL

Os seguintes são alguns eventos de espera comuns do Aurora PostgreSQL. Para saber mais sobre eventos de espera e como ajustar um cluster de banco de dados do Aurora PostgreSQL, consulte Ajustar com eventos de espera do Aurora PostgreSQL.

Activity:ArchiverMain

O processo do arquivador está aguardando atividades.

Activity:AutoVacuumMain

O processo do launcher de autovacuum está aguardando atividades.

Activity:BgWriterHibernate

O processo do gravador em segundo plano está hibernando enquanto aguarda atividades.

Activity:BgWriterMain

O processo do gravador em segundo plano está aguardando atividades.

Activity:CheckpointerMain

O processo do ponteiro de verificação está aguardando atividades.

Activity:LogicalApplyMain

O processo da aplicação de replicação lógica está aguardando atividades.

Activity:LogicalLauncherMain

O processo do launcher de replicação lógica está aguardando atividades.

Activity:PgStatMain

O processo do coletor de estatísticas está aguardando atividades.

Activity:RecoveryWalAll

Um processo está aguardando o log de gravação antecipada (WAL) de um fluxo na recuperação.

Activity:RecoveryWalStream

O processo de inicialização está aguardando a chegada do log de gravação antecipada (WAL) durante a recuperação da transmissão.

Activity:SysLoggerMain

O processo do syslogger está aguardando atividades.

Activity:WalReceiverMain

O processo do receptor do log de gravação antecipada (WAL) está aguardando atividades.

Activity:WalSenderMain

O processo do emissor do log de gravação antecipada (WAL) está aguardando atividades.

Activity:WalWriterMain

O processo do gravador do log de gravação antecipada (WAL) está aguardando atividades.

BufferPin:BufferPin

Um processo está aguardando para adquirir um pino exclusivo em um buffer.

Client:GSSOpenServer

Um processo está aguardando para ler dados do cliente enquanto estabelece uma sessão Generic Security Service Application Program Interface (GSSAPI).

Client:ClientRead

Um processo de backend está aguardando para receber dados de um cliente PostgreSQL. Para obter mais informações, consulte Client:ClientRead.

Client:ClientWrite

Um processo de backend está aguardando para enviar mais dados a um cliente PostgreSQL. Para obter mais informações, consulte Client:ClientWrite.

Client:LibPQWalReceiverConnect

Um processo está aguardando no receptor do log de gravação antecipada (WAL) para estabelecer a conexão com o servidor remoto.

Client:LibPQWalReceiverReceive

Um processo está aguardando no receptor do log de gravação antecipada (WAL) para receber dados do servidor remoto.

Client:SSLOpenServer

Um processo está aguardando o Secure Sockets Layer (SSL) ao tentar a conexão.

Client:WalReceiverWaitStart

Um processo está aguardando o processo de inicialização enviar dados iniciais para replicação de transmissão.

Client:WalSenderWaitForWAL

Um processo está aguardando o log de gravação antecipada (WAL) ser liberado no processo do emissor do WAL.

Client:WalSenderWriteData

Um processo está aguardando qualquer atividade ao processar respostas do receptor do log de gravação antecipada (WAL) no processo do emissor do WAL.

CPU

Um processo de backend está ativo ou está aguardando a CPU. Para obter mais informações, consulte CPU.

Extension:extension

Um processo de backend está aguardando uma condição definida por uma extensão ou um módulo.

IO:AuroraEnhancedLogicalWALRead

Um processo de backend está buscando registros de log do volume de captura de dados de alteração (CDC).

IO:AuroraOptimizedReadsCacheRead

Um processo está aguardando uma leitura do cache em camadas do Optimized Reads porque a página não está disponível na memória compartilhada.

IO:AuroraOptimizedReadsCacheSegmentTruncate

Um processo está aguardando o truncamento de um arquivo de segmento de cache em camadas do Optimized Reads.

IO:AuroraOptimizedReadsCacheWrite

O processo de gravação em segundo plano está aguardando o cache em camadas do Optimized Reads

IO:AuroraStorageLogAllocate

Uma sessão está alocando metadados e se preparando para uma gravação do log de transações.

IO:BufFileRead

Quando as operações exigem mais memória do que a quantidade definida por parâmetros de memória de trabalho, o mecanismo cria arquivos temporários no disco. Esse evento de espera ocorre quando as operações leem dos arquivos temporários. Para obter mais informações, consulte IO:BufFileRead and IO:BufFileWrite.

IO:BufFileWrite

Quando as operações exigem mais memória do que a quantidade definida por parâmetros de memória de trabalho, o mecanismo cria arquivos temporários no disco. Esse evento de espera ocorre quando as operações gravam nos arquivos temporários. Para obter mais informações, consulte IO:BufFileRead and IO:BufFileWrite.

IO:ControlFileRead

Um processo está aguardando uma leitura do arquivo `pg_control`.

IO:ControlFileSync

Um processo está aguardando o arquivo `pg_control` alcançar o armazenamento durável.

IO:ControlFileSyncUpdate

Um processo está aguardando uma atualização do arquivo `pg_control` alcançar o armazenamento durável.

IO:ControlFileWrite

Um processo está aguardando uma gravação no arquivo `pg_control`.

IO:ControlFileWriteUpdate

Um processo está aguardando uma gravação atualizar o arquivo `pg_control`.

IO:CopyFileRead

Um processo está aguardando uma leitura durante uma operação de cópia de arquivos.

IO:CopyFileWrite

Um processo está aguardando uma gravação durante uma operação de cópia de arquivos.

IO:DataFileExtend

Um processo está aguardando a extensão de um arquivo de dados de relação.

IO:DataFileFlush

Um processo está aguardando um arquivo de dados de relação alcançar o armazenamento durável.

IO:DataFileImmediateSync

Um processo está aguardando uma sincronização imediata de um arquivo de dados de relação com o armazenamento durável.

IO:DataFilePrefetch

Um processo está aguardando uma pré-busca assíncrona de um arquivo de dados de relação.

IO:DataFileSync

Um processo está aguardando alterações em um arquivo de dados de relação alcançarem o armazenamento durável.

IO:DataFileRead

Um processo de backend tentou encontrar uma página nos buffers compartilhados, não a encontrou e, portanto, fez a sua leitura no armazenamento. Para obter mais informações, consulte IO:DataFileRead.

IO:DataFileTruncate

Um processo está aguardando o truncamento de um arquivo de dados de relação.

IO:DataFileWrite

Um processo está aguardando uma gravação em um arquivo de dados de relação.

IO:DSMFillZeroWrite

Um processo está aguardando para gravar zero bytes em um arquivo dinâmico de backup de memória compartilhada.

IO:LockFileAddToDataDirRead

Um processo está aguardando uma leitura enquanto adiciona uma linha ao arquivo de bloqueio do diretório de dados.

IO:LockFileAddToDataDirSync

Um processo está aguardando os dados alcançarem o armazenamento durável enquanto adiciona uma linha ao arquivo de bloqueio do diretório de dados.

IO:LockFileAddToDataDirWrite

Um processo está aguardando uma gravação enquanto adiciona uma linha ao arquivo de bloqueio do diretório de dados.

IO:LockFileCreateRead

Um processo está aguardando uma leitura enquanto cria o arquivo de bloqueio do diretório de dados.

IO:LockFileCreateSync

Um processo está aguardando os dados alcançarem o armazenamento durável enquanto cria o arquivo de bloqueio do diretório de dados.

IO:LockFileCreateWrite

Um processo está aguardando uma gravação enquanto cria o arquivo de bloqueio do diretório de dados.

IO:LockFileReCheckDataDirRead

Um processo está aguardando uma leitura durante a reverificação do arquivo de bloqueio do diretório de dados.

IO:LogicalRewriteCheckpointSync

Um processo está aguardando os mapeamentos de regravação lógica alcançarem o armazenamento durável durante um ponto de verificação.

IO:LogicalRewriteMappingSync

Um processo está aguardando os dados de mapeamento alcançarem o armazenamento durável durante uma regravação lógica.

IO:LogicalRewriteMappingWrite

Um processo está aguardando uma gravação de dados de mapeamento durante uma regravação lógica.

IO:LogicalRewriteSync

Um processo está aguardando os mapeamentos de regravação lógica alcançarem o armazenamento durável.

IO:LogicalRewriteTruncate

Um processo está aguardando o truncamento de dados de mapeamento durante uma regravação lógica.

IO:LogicalRewriteWrite

Um processo está aguardando uma gravação de mapeamentos de regravação lógica.

IO:RelationMapRead

Um processo está aguardando uma leitura do arquivo de mapa de relação.

IO:RelationMapSync

Um processo está aguardando o arquivo de mapa de relação alcançar o armazenamento durável.

IO:RelationMapWrite

Um processo está aguardando uma gravação no arquivo de mapa de relação.

IO:ReorderBufferRead

Um processo está aguardando uma leitura durante o gerenciamento do buffer de reordenação.

IO:ReorderBufferWrite

Um processo está aguardando uma gravação durante o gerenciamento do buffer de reordenação.

IO:ReorderLogicalMappingRead

Um processo está aguardando uma leitura de um mapeamento lógico durante o gerenciamento do buffer de reordenação.

IO:ReplicationSlotRead

Um processo está aguardando uma leitura de um arquivo de controle de slot de replicação.

IO:ReplicationSlotRestoreSync

Um processo está aguardando um arquivo de controle de slot de replicação alcançar o armazenamento durável enquanto o restaura na memória.

IO:ReplicationSlotSync

Um processo está aguardando um arquivo de controle de slot de replicação alcançar o armazenamento durável.

IO:ReplicationSlotWrite

Um processo está aguardando uma gravação em um arquivo de controle de slot de replicação.

IO:SLRUFlushSync

Um processo está aguardando os dados simples menos recentemente utilizados (SLRU) alcançarem o armazenamento durável durante um ponto de verificação ou desligamento do banco de dados.

IO:SLRURead

Um processo está aguardando uma leitura de uma página simples menos recentemente utilizada (SLRU).

IO:SLRUSync

Um processo está aguardando os dados simples menos recentemente utilizados (SLRU) alcançarem o armazenamento durável após uma gravação de página.

IO:SLRUWrite

Um processo está aguardando uma gravação de uma página simples menos recentemente utilizada (SLRU).

IO:SnapbuildRead

Um processo está aguardando uma leitura de um snapshot de catálogo histórico serializado.

IO:SnapbuildSync

Um processo está aguardando um snapshot de catálogo histórico serializado alcançar o armazenamento durável.

IO:SnapbuildWrite

Um processo está aguardando uma gravação de um snapshot de catálogo histórico serializado.

IO:TimelineHistoryFileSync

Um processo está aguardando um arquivo de histórico da linha do tempo recebido via replicação de transmissão alcançar o armazenamento durável.

IO:TimelineHistoryFileWrite

Um processo está aguardando uma gravação de um arquivo de histórico da linha do tempo recebido via replicação de transmissão.

IO:TimelineHistoryRead

Um processo está aguardando uma leitura de um arquivo de histórico da linha do tempo.

IO:TimelineHistorySync

Um processo está aguardando um arquivo de histórico da linha do tempo recém-criado alcançar o armazenamento durável.

IO:TimelineHistoryWrite

Um processo está aguardando uma gravação de um arquivo de histórico da linha do tempo recém-criado.

IO:TwophaseFileRead

Um processo está aguardando uma leitura de um arquivo de estado de duas fases.

IO:TwophaseFileSync

Um processo está aguardando um arquivo de estado de duas fases alcançar o armazenamento durável.

IO:TwophaseFileWrite

Um processo está aguardando uma gravação de um arquivo de estado de duas fases.

IO:WALBootstrapSync

Um processo está aguardando o log de gravação antecipada (WAL) alcançar o armazenamento durável durante o bootstrapping.

IO:WALBootstrapWrite

Um processo está aguardando uma gravação de uma página do log de gravação antecipada (WAL) durante o bootstrapping.

IO:WALCopyRead

Um processo está aguardando uma leitura ao criar um novo segmento do log de gravação antecipada (WAL) por meio da cópia de um existente.

IO:WALCopySync

Um processo está aguardando um novo segmento do log de gravação antecipada (WAL), criado por meio da cópia de um existente, alcançar o armazenamento durável.

IO:WALCopyWrite

Um processo está aguardando uma gravação ao criar um novo segmento do log de gravação antecipada (WAL) por meio da cópia de um existente.

IO:WALInitSync

Um processo está aguardando um arquivo do log de gravação antecipada (WAL) recém-inicializado alcançar um armazenamento durável.

IO:WALInitWrite

Um processo está aguardando uma gravação ao inicializar um novo arquivo do log de gravação antecipada (WAL).

IO:WALRead

Um processo está aguardando uma leitura de um arquivo do log de gravação antecipada (WAL).

IO:WALSenderTimelineHistoryRead

Um processo está aguardando uma leitura de um arquivo de histórico da linha do tempo durante um comando de linha do tempo do emissor do WAL.

IO:WALSync

Um processo está aguardando um arquivo do log de gravação antecipada (WAL) alcançar o armazenamento durável.

IO:WALSyncMethodAssign

Um processo está aguardando os dados alcançarem o armazenamento durável enquanto atribui um novo método de sincronização do log de gravação antecipada (WAL).

IO:WALWrite

Um processo está aguardando uma gravação em um arquivo do log de gravação antecipada (WAL).

IO:XactSync

Um processo de backend está aguardando o subsistema de armazenamento do Aurora fazer a confirmação de uma transação regular ou a confirmação ou reversão de uma transação preparada. Para obter mais informações, consulte IO:XactSync.

IPC:AuroraLogicalSchemaUpdate

Dois processos de backend estão tentando inserir a mesma entrada no cache do esquema. Um processo continuará enquanto o outro aguarda sua conclusão.

IPC:AuroraOptimizedReadsCacheWriteStop

Um processo está aguardando o gravador em segundo plano interromper a gravação no cache em níveis do Optimized Reads.

IPC:BackupWaitWalArchive

Um processo está aguardando os arquivos do log de gravação antecipada (WAL) necessários para que um backup seja arquivado com êxito.

IPC:BgWorkerShutdown

Um processo está aguardando o encerramento de um operador em segundo plano.

IPC:BgWorkerStartup

Um processo está aguardando o início de um operador em segundo plano.

IPC:BtreePage

Um processo está aguardando o número de página necessário para continuar uma varredura paralela de árvore B para ficar disponível.

IPC:CheckpointDone

Um processo está aguardando a conclusão de um ponto de verificação.

IPC:CheckpointStart

Um processo está aguardando o início de um ponto de verificação.

IPC:ClogGroupUpdate

Um processo está aguardando o líder do grupo atualizar o status da transação no final de uma transação.

IPC:DamRecordTxAck

Um processo de backend gerou um evento de fluxos de atividades de banco de dados e está aguardando esse evento se tornar durável. Para obter mais informações, consulte IPC:DamRecordTxAck.

IPC:ExecuteGather

Um processo está aguardando atividades de um processo filho durante a execução de um nó do plano Gather.

IPC:Hash/Batch/Allocating

Um processo está aguardando um participante de hash paralelo eleito alocar uma tabela de hash.

IPC:Hash/Batch/Electing

Um processo está elegendo um participante de hash paralelo para alocar uma tabela de hash.

IPC:Hash/Batch/Loading

Um processo está aguardando outros participantes de hash paralelos terminarem de carregar uma tabela de hash.

IPC:Hash/Build/Allocating

Um processo está aguardando um participante de hash paralelo eleito alocar a tabela de hash inicial.

IPC:Hash/Build/Electing

Um processo está elegendo um participante de hash paralelo para alocar a tabela de hash inicial.

IPC:Hash/Build/HashingInner

Um processo está aguardando outros participantes de hash paralelos terminarem o hash da relação interna.

IPC:Hash/Build/HashingOuter

Um processo está aguardando outros participantes de hash paralelos terminarem de particionar a relação externa.

IPC:Hash/GrowBatches/Allocating

Um processo está aguardando um participante de hash paralelo eleito alocar mais lotes.

IPC:Hash/GrowBatches/Deciding

Um processo está elegendo um participante de hash paralelo para decidir sobre o crescimento futuro do lote.

IPC:Hash/GrowBatches/Electing

Um processo está elegendo um participante de hash paralelo para alocar mais lotes.

IPC:Hash/GrowBatches/Finishing

Um processo está aguardando um participante de hash paralelo eleito decidir sobre o crescimento futuro do lote.

IPC:Hash/GrowBatches/Repartitioning

Um processo está aguardando outros participantes de hash paralelos terminarem o reparticionamento.

IPC:Hash/GrowBuckets/Allocating

Um processo está aguardando um participante de hash paralelo eleito terminar a alocação de mais buckets.

IPC:Hash/GrowBuckets/Electing

Um processo está elegendo um participante de hash paralelo para alocar mais buckets.

IPC:Hash/GrowBuckets/Reinserting

Um processo está aguardando outros participantes de hash paralelos terminarem de inserir tuplas em novos buckets.

IPC:HashBatchAllocate

Um processo está aguardando um participante de hash paralelo eleito alocar uma tabela de hash.

IPC:HashBatchElect

Um processo está aguardando para eleger um participante de hash paralelo para alocar uma tabela de hash.

IPC:HashBatchLoad

Um processo está aguardando outros participantes de hash paralelos terminarem de carregar uma tabela de hash.

IPC:HashBuildAllocate

Um processo está aguardando um participante de hash paralelo eleito alocar a tabela de hash inicial.

IPC:HashBuildElect

Um processo está aguardando para eleger um participante de hash paralelo para alocar a tabela de hash inicial.

IPC:HashBuildHashInner

Um processo está aguardando outros participantes de hash paralelos terminarem o hash da relação interna.

IPC:'HashBuildHashOuter

Um processo está aguardando outros participantes de hash paralelos terminarem de particionar a relação externa.

IPC:HashGrowBatchesAllocate

Um processo está aguardando um participante de hash paralelo eleito alocar mais lotes.

IPC:'HashGrowBatchesDecide

Um processo está aguardando para eleger um participante de hash paralelo para decidir sobre o crescimento futuro do lote.

IPC:HashGrowBatchesElect

Um processo está aguardando para eleger um participante de hash paralelo para alocar mais lotes.

IPC:HashGrowBatchesFinish

Um processo está aguardando um participante de hash paralelo eleito decidir sobre o crescimento futuro do lote.

IPC:HashGrowBatchesRepartition

Um processo está aguardando outros participantes de hash paralelos terminarem o reparticionamento.

IPC:HashGrowBucketsAllocate

Um processo está aguardando um participante de hash paralelo eleito terminar a alocação de mais buckets.

IPC:HashGrowBucketsElect

Um processo está aguardando para eleger um participante de hash paralelo para alocar mais buckets.

IPC:HashGrowBucketsReinsert

Um processo está aguardando outros participantes de hash paralelos terminarem de inserir tuplas em novos buckets.

IPC:LogicalSyncData

Um processo está aguardando um servidor remoto de replicação lógica enviar dados para sincronização da tabela inicial.

IPC:LogicalSyncStateChange

Um processo está aguardando um servidor remoto de replicação lógica mudar de estado.

IPC:MessageQueueInternal

Um processo está aguardando outro processo ser anexado a uma fila de mensagens compartilhadas.

IPC:MessageQueuePutMessage

Um processo está aguardando para gravar uma mensagem de protocolo em uma fila de mensagens compartilhadas.

IPC:MessageQueueReceive

Um processo está aguardando para receber bytes de uma fila de mensagens compartilhadas.

IPC:MessageQueueSend

Um processo está aguardando para enviar bytes para uma fila de mensagens compartilhadas.

IPC:ParallelBitmapScan

Um processo está aguardando a inicialização de uma varredura de bitmap paralela.

IPC:ParallelCreateIndexScan

Um processo está aguardando operadores paralelos de CREATE INDEX terminarem uma varredura de heap.

IPC:ParallelFinish

Um processo está aguardando operadores paralelos terminarem o cálculo.

IPC:ProcArrayGroupUpdate

Um processo está aguardando o líder do grupo limpar o ID da transação no final de uma transação.

IPC:ProcSignalBarrier

Um processo está aguardando um evento de barreira ser processado por todos os backends.

IPC:Promote

Um processo está aguardando uma promoção para espera.

IPC:RecoveryConflictSnapshot

Um processo está aguardando a resolução de um conflito de recuperação para uma limpeza de vacuum.

IPC:RecoveryConflictTablespace

Um processo está aguardando a resolução de um conflito de recuperação para descartar um espaço de tabela.

IPC:RecoveryPause

Um processo está aguardando a retomada da recuperação.

IPC:ReplicationOriginDrop

Um processo está aguardando uma origem de replicação tornar-se inativa para que ela possa ser descartada.

IPC:ReplicationSlotDrop

Um processo está aguardando um slot de replicação tornar-se inativo para que ele possa ser descartado.

IPC:SafeSnapshot

Um processo está aguardando para obter um snapshot válido para uma transação READ ONLY DEFERRABLE.

IPC:SyncRep

Um processo está aguardando confirmação de um servidor remoto durante a replicação síncrona.

IPC:XactGroupUpdate

Um processo está aguardando o líder do grupo atualizar o status da transação no final de uma transação.

Lock:advisory

Um processo de backend solicitou um bloqueio de consultoria e está aguardando por ele. Para obter mais informações, consulte Lock:advisory.

Lock:extend

Um processo de backend está aguardando um bloqueio ser liberado para poder estender uma relação. Esse bloqueio é necessário porque apenas um processo de backend pode estender uma relação por vez. Para obter mais informações, consulte Lock:extend.

Lock:frozenid

Um processo está aguardando para atualizar `pg_database.datfrozenxid` e `pg_database.datminmxid`.

Lock:object

Um processo está aguardando para obter um bloqueio em um objeto de banco de dados sem relação.

Lock:page

Um processo está aguardando para obter um bloqueio em uma página de uma relação.

Lock:Relation

Um processo de backend aguarda para adquirir um bloqueio em uma relação bloqueada por outra transação. Para obter mais informações, consulte Lock:Relation.

Lock:spectoken

Um processo está aguardando para obter um bloqueio de inserção especulativo.

Lock:speculative token

Um processo está aguardando para adquirir um bloqueio de inserção especulativo.

Lock:transactionid

Uma transação está aguardando um bloqueio em nível de linha. Para obter mais informações, consulte Lock:transactionid.

Lock:tuple

Um processo de backend está aguardando para adquirir um bloqueio em uma tupla enquanto outro processo de backend mantém um bloqueio conflitante na mesma tupla. Para obter mais informações, consulte Lock:tuple.

Lock:userlock

Um processo está aguardando para obter um bloqueio de usuário.

Lock:virtualxid

Um processo está aguardando para obter um bloqueio de ID de transação virtual.

LWLock:AddinShmemInit

Um processo está aguardando para gerenciar a alocação de espaço de uma extensão na memória compartilhada.

LWLock:AddinShmemInitLock

Um processo está aguardando para gerenciar a alocação de espaço na memória compartilhada.

LWLock:async

Um processo está aguardando E/S em um buffer assíncrono (notificação).

LWLock:AsyncCtlLock

Um processo está aguardando para ler ou atualizar um estado de notificação compartilhado.

LWLock:AsyncQueueLock

Um processo está aguardando para ler ou atualizar mensagens de notificação.

LWLock:AuroraOptimizedReadsCacheMapping

Um processo está aguardando para associar um bloco de dados a uma página no cache do Optimized Reads.

LWLock:AutoFile

Um processo está aguardando para atualizar o arquivo `postgresql.auto.conf`.

LWLock:AutoFileLock

Um processo está aguardando para atualizar o arquivo `postgresql.auto.conf`.

LWLock:Autovacuum

Um processo está aguardando para ler ou atualizar o estado atual de operadores de autovacuum.

LWLock:AutovacuumLock

Um operador ou launcher de autovacuum está aguardando para atualizar ou ler o estado atual de operadores de autovacuum.

LWLock:AutovacuumSchedule

Um processo está aguardando para garantir que uma tabela selecionada para autovacuum ainda requer aplicação de vacuum.

LWLock:AutovacuumScheduleLock

Um processo está aguardando para garantir que a tabela que ele selecionou para um vacuum ainda requer aplicação de vacuum.

LWLock:BackendRandomLock

Um processo está aguardando para gerar um número aleatório.

LWLock:BackgroundWorker

Um processo está aguardando para ler ou atualizar o estado do operador em segundo plano.

LWLock:BackgroundWorkerLock

Um processo está aguardando para ler ou atualizar o estado do operador em segundo plano.

LWLock:BtreeVacuum

Um processo está aguardando para ler ou atualizar informações relacionadas a vacuum para um índice de árvores B.

LWLock:BtreeVacuumLock

Um processo está aguardando para ler ou atualizar informações relacionadas a vacuum para um índice de árvores B.

LWLock:buffer_content

Um processo de backend está aguardando para adquirir um bloqueio leve no conteúdo de um buffer de memória compartilhada. Para obter mais informações, consulte LWLock:buffer_content (BufferContent).

LWLock:buffer_mapping

Um processo de backend está aguardando para associar um bloqueio de dados a um buffer no grupo de buffer compartilhado. Para obter mais informações, consulte LWLock:buffer_mapping.

LWLock:BufferIO

Um processo de backend deseja ler uma página na memória compartilhada. O processo está aguardando outros processos terminarem sua E/S para a página. Para obter mais informações, consulte LWLock:BufferIO (IPC:BufferIO).

LWLock:Checkpoint

Um processo está aguardando para iniciar um ponto de verificação.

LWLock:CheckpointLock

Um processo está aguardando para realizar o ponto de verificação.

LWLock:CheckpointerComm

Um processo está aguardando para gerenciar solicitações `fsync`.

LWLock:CheckpointerCommLock

Um processo está aguardando para gerenciar solicitações `fsync`.

LWLock:clog

Um processo está aguardando E/S em um buffer de entupimento (status da transação).

LWLock:CLogControlLock

Um processo está aguardando para ler ou atualizar o status da transação.

LWLock:CLogTruncationLock

Um processo está aguardando para executar `txid_status` ou atualizar o ID de transação mais antigo disponível para ele.

LWLock:commit_timestamp

Um processo está aguardando E/S em um buffer de carimbo de data/hora de confirmação.

LWLock:CommitTs

Um processo está aguardando para ler ou atualizar o último valor definido para um carimbo de data/hora de confirmação de transação.

LWLock:CommitTsBuffer

Um processo está aguardando E/S em um buffer simples menos recentemente utilizado (SLRU) para um carimbo de data/hora de confirmação.

LWLock:CommitTsControlLock

Um processo está aguardando para ler ou atualizar carimbos de data/hora de confirmação de transação.

LWLock:CommitTsLock

Um processo está aguardando para ler ou atualizar o último valor definido para o carimbo de data/hora da transação.

LWLock:CommitTsSLRU

Um processo está aguardando para acessar o cache simples menos recentemente utilizado (SLRU) para um carimbo de data/hora de confirmação.

LWLock:ControlFile

Um processo está aguardando para ler ou atualizar o arquivo `pg_control` ou criar um novo arquivo de log de gravação antecipada (WAL).

LWLock:ControlFileLock

Um processo está aguardando para ler ou atualizar o arquivo de controle ou a criação de um novo arquivo de log de gravação antecipada (WAL).

LWLock:DynamicSharedMemoryControl

Um processo está aguardando para ler ou atualizar informações de alocação de memória compartilhada dinâmica.

LWLock:DynamicSharedMemoryControlLock

Um processo está aguardando para ler ou atualizar o estado de memória compartilhada dinâmica.

LWLock:lock_manager

Um processo de backend está aguardando para adicionar ou examinar bloqueios para processos de backend. Ou está aguardando para entrar ou sair de um grupo de bloqueio utilizado pela consulta paralela. Para obter mais informações, consulte LWLock:lock_manager.

LWLock:LockFastPath

Um processo está aguardando para ler ou atualizar as informações de bloqueio de caminho rápido de um processo.

LWLock:LogicalRepWorker

Um processo está aguardando para ler ou atualizar o estado dos operadores de replicação lógica.

LWLock:LogicalRepWorkerLock

Um processo está aguardando a conclusão de uma ação em um operador de replicação lógica.

LWLock:LogicalSchemaCache

Um processo modificou o cache do esquema.

LWLock:multixact_member

Um processo está aguardando E/S em um buffer multixact_member.

LWLock:multixact_offset

Um processo está aguardando E/S em um buffer de deslocamento multixact.

LWLock:MultiXactGen

Um processo está aguardando para ler ou atualizar o estado multixact compartilhado.

LWLock:MultiXactGenLock

Um processo está aguardando para ler ou atualizar um estado multixact compartilhado.

LWLock:MultiXactMemberBuffer

Um processo está aguardando a E/S em um buffer simples menos recentemente utilizado (SLRU) para um membro multixact. Para obter mais informações, consulte LWLock:MultiXact.

LWLock:MultiXactMemberControlLock

Um processo está aguardando para ler ou atualizar mapeamentos de membros multixact.

LWLock:MultiXactMemberSLRU

Um processo está aguardando para acessar o cache simples menos recentemente utilizado (SLRU) para um membro multixact. Para obter mais informações, consulte LWLock:MultiXact.

LWLock:MultiXactOffsetBuffer

Um processo está aguardando a E/S em um buffer simples menos recentemente utilizado (SLRU) para um desvio multixact. Para obter mais informações, consulte LWLock:MultiXact.

LWLock:MultiXactOffsetControlLock

Um processo está aguardando para ler ou atualizar mapeamentos de deslocamento multixact.

LWLock:MultiXactOffsetSLRU

Um processo está aguardando para acessar o cache simples menos recentemente utilizado (SLRU) para um desvio multixact. Para obter mais informações, consulte LWLock:MultiXact.

LWLock:MultiXactTruncation

Um processo está aguardando para ler ou truncar informações multixact.

LWLock:MultiXactTruncationLock

Um processo está aguardando para ler ou truncar informações multixact.

LWLock:NotifyBuffer

Um processo está aguardando a E/S no buffer simples menos recentemente utilizado (SLRU) para uma mensagem NOTIFY.

LWLock:NotifyQueue

Um processo está aguardando para ler ou atualizar mensagens NOTIFY.

LWLock:NotifyQueueTail

Um processo está aguardando para atualizar um limite no armazenamento de mensagens NOTIFY.

LWLock:NotifyQueueTailLock

Um processo está aguardando para atualizar o limite no armazenamento de mensagens de notificação.

LWLock:NotifySLRU

Um processo está aguardando para acessar o cache simples menos frequentemente utilizado (SLRU) para uma mensagem NOTIFY.

LWLock:OidGen

Um processo está aguardando para alocar um novo ID de objeto (OID).

LWLock:OidGenLock

Um processo está aguardando para alocar ou atribuir um ID de objeto (OID).

LWLock:oldserxid

Um processo está aguardando E/S em um buffer oldserxid.

LWLock:OldSerXidLock

Um processo está aguardando para ler ou registrar transações serializáveis conflitantes.

LWLock:OldSnapshotTimeMap

Um processo está aguardando para ler ou atualizar informações antigas de controle de snapshots.

LWLock:OldSnapshotTimeMapLock

Um processo está aguardando para ler ou atualizar informações antigas de controle de snapshots.

LWLock:parallel_append

Um processo está aguardando para escolher o próximo subplano durante a execução do plano de anexação em paralelo.

LWLock:parallel_hash_join

Um processo está aguardando para alocar ou trocar um bloco de memória ou atualizar contadores durante a execução de um plano de hash paralelo.

LWLock:parallel_query_dsa

Um processo está aguardando um bloqueio na alocação de memória compartilhada dinâmica para uma consulta paralela.

LWLock:ParallelAppend

Um processo está aguardando para escolher o próximo subplano durante a execução do plano de anexação em paralelo.

LWLock:ParallelHashJoin

Um processo está aguardando para sincronizar os operadores durante a execução do plano para uma junção de hash paralela.

Lwlock:ParallelQueryDSA

Um processo está aguardando a alocação dinâmica de memória compartilhada para uma consulta paralela.

Lwlock:PerSessionDSA

Um processo está aguardando a alocação dinâmica de memória compartilhada para uma consulta paralela.

Lwlock:PerSessionRecordType

Um processo está aguardando para acessar informações de uma consulta paralela sobre tipos compostos.

Lwlock:PerSessionRecordTypmod

Um processo está aguardando para acessar informações de uma consulta paralela sobre modificadores de tipo que identificam tipos de registros anônimos.

Lwlock:PerXactPredicateList

Um processo está aguardando para acessar a lista de bloqueios de predicados mantidos pela transação serializável atual durante uma consulta paralela.

Lwlock:predicate_lock_manager

Um processo está aguardando para adicionar ou examinar informações de bloqueio de predicado.

Lwlock:PredicateLockManager

Um processo está aguardando para acessar informações de bloqueio de predicado utilizadas por transações serializáveis.

Lwlock:proc

Um processo está aguardando para ler ou atualizar informações de bloqueio de caminho rápido.

LWLock:ProcArray

Um processo está aguardando para acessar as estruturas de dados compartilhadas por processo (em geral, para obter um snapshot ou relatar o ID da transação de uma sessão).

LWLock:ProcArrayLock

Um processo está aguardando para obter um snapshot ou apagar um ID de transação ao final de uma transação.

LWLock:RelationMapping

Um processo está aguardando para ler ou atualizar um arquivo `pg_filenode.map` (usado para rastrear atribuições de nó de arquivo de determinados catálogos do sistema).

LWLock:RelationMappingLock

Um processo está aguardando para atualizar o arquivo de mapa de relação utilizado para armazenar o mapeamento de catálogo para nó de arquivo.

LWLock:RelCacheInit

Um processo está aguardando para ler ou atualizar um arquivo `pg_internal.init` (um arquivo de inicialização do cache de relação).

LWLock:RelCacheInitLock

Um processo está aguardando para ler ou gravar um arquivo de inicialização do cache de relação.

LWLock:replication_origin

Um processo está aguardando para ler ou atualizar o progresso da replicação.

LWLock:replication_slot_io

Um processo está aguardando E/S em um slot de replicação.

LWLock:ReplicationOrigin

Um processo está aguardando para criar, descartar ou utilizar uma origem de replicação.

LWLock:ReplicationOriginLock

Um processo está aguardando para configurar, descartar ou utilizar uma origem de replicação.

LWLock:ReplicationOriginState

Um processo está aguardando para ler ou atualizar o progresso de uma origem de replicação.

LWLock:ReplicationSlotAllocation

Um processo está aguardando para alocar ou liberar um slot de replicação.

LWLock:ReplicationSlotAllocationLock

Um processo está aguardando para alocar ou liberar um slot de replicação.

LWLock:ReplicationSlotControl

Um processo está aguardando para ler ou atualizar um estado de slot de replicação.

LWLock:ReplicationSlotControlLock

Um processo está aguardando para ler ou atualizar o estado do slot de replicação.

LWLock:ReplicationSlotIO

Um processo está aguardando E/S em um slot de replicação.

LWLock:SerialBuffer

Um processo está aguardando E/S em um buffer simples menos recentemente utilizado (SLRU) para um conflito de transação serializável.

LWLock:SerializableFinishedList

Um processo está aguardando para acessar a lista de transações serializáveis concluídas.

LWLock:SerializableFinishedListLock

Um processo está aguardando para acessar a lista de transações serializáveis concluídas.

LWLock:SerializablePredicateList

Um processo está aguardando para acessar a lista de bloqueios de predicados mantidos por transações serializáveis.

LWLock:SerializablePredicateLockListLock

Um processo está aguardando para executar uma operação em uma lista de bloqueios mantidos por transações serializáveis.

LWLock:SerializableXactHash

Um processo está aguardando para ler ou atualizar informações sobre transações serializáveis.

LWLock:SerializableXactHashLock

Um processo está aguardando para recuperar ou armazenar informações sobre transações serializáveis.

LWLock:SerialSLRU

Um processo está aguardando para acessar o cache simples menos recentemente utilizado (SLRU) para um conflito de transação serializável.

LWLock:SharedTidBitmap

Um processo está aguardando para acessar um bitmap de identificador de tupla compartilhado (TID) durante uma varredura paralela de índice de bitmap.

LWLock:SharedTupleStore

Um processo está aguardando para acessar um armazenamento de tuplas compartilhado durante uma consulta paralela.

LWLock:ShmemIndex

Um processo está aguardando para encontrar ou alocar espaço na memória compartilhada.

LWLock:ShmemIndexLock

Um processo está aguardando para encontrar ou alocar espaço na memória compartilhada.

LWLock:SInvalRead

Um processo está aguardando para recuperar mensagens da fila de anulação de catálogo compartilhado.

LWLock:SInvalReadLock

Um processo está aguardando para recuperar ou remover mensagens de uma fila de invalidação compartilhada.

LWLock:SInvalWrite

Um processo está aguardando para adicionar uma mensagem à fila de anulação de catálogo compartilhada.

LWLock:SInvalWriteLock

Um processo está aguardando para adicionar uma mensagem em uma fila de invalidação compartilhada.

LWLock:SyncRep

Um processo está aguardando para ler ou atualizar informações sobre o estado da replicação síncrona.

LWLock:SyncRepLock

Um processo está aguardando para ler ou atualizar informações sobre réplicas síncronas.

LWLock:SyncScan

Um processo está aguardando para selecionar a localização inicial de uma varredura de tabela sincronizada.

LWLock:SyncScanLock

Um processo está aguardando para obter a localização inicial de uma varredura em uma tabela para varreduras sincronizadas.

LWLock:TablespaceCreate

Um processo está aguardando para criar ou descartar um espaço de tabelas.

LWLock:TablespaceCreateLock

Um processo está aguardando para criar ou descartar o espaço de tabelas.

LWLock:tbm

Um processo está aguardando um bloqueio de iterador compartilhado em um bitmap de árvore (TBM).

LWLock:TwoPhaseState

Um processo está aguardando para ler ou atualizar o estado de transações preparadas.

LWLock:TwoPhaseStateLock

Um processo está aguardando para ler ou atualizar o estado de transações preparadas.

LWLock:wal_insert

Um processo está aguardando para inserir o log de gravação antecipada (WAL) em um buffer de memória.

LWLock:WALBufMapping

Um processo está aguardando para substituir uma página em buffers do log de gravação antecipada (WAL).

LWLock:WALBufMappingLock

Um processo está aguardando para substituir uma página em buffers do log de gravação antecipada (WAL).

LWLock:WALInsert

Um processo está aguardando para inserir dados do log de gravação antecipada (WAL) em um buffer de memória.

LWLock:WALWrite

Um processo está aguardando os buffers do log de gravação antecipada (WAL) serem gravados no disco.

LWLock:WALWriteLock

Um processo está aguardando os buffers do log de gravação antecipada (WAL) serem gravados no disco.

LWLock:WrapLimitsVacuum

Um processo está aguardando para atualizar limites sobre o ID de transação e o consumo multixact.

LWLock:WrapLimitsVacuumLock

Um processo está aguardando para atualizar limites sobre o ID de transação e o consumo multixact.

LWLock:XactBuffer

Um processo está aguardando a E/S em um buffer simples menos recentemente utilizado (SLRU) para um status de transação.

LWLock:XactSLRU

Um processo está aguardando para acessar o cache simples menos frequentemente utilizado (SLRU) para um status de transação.

LWLock:XactTruncation

Um processo está aguardando para executar pg_xact_status ou atualizar o ID de transação mais antigo disponível para ele.

LWLock:XidGen

Um processo está aguardando para alocar um novo ID de transação.

LWLock:XidGenLock

Um processo está aguardando para alocar ou atribuir um ID de transação.

Timeout:BaseBackupThrottle

Um processo está aguardando durante o backup básico ao controlar a utilização da atividade.

Tempo limite:PgSleep

Um processo de backend chamou a função pg_sleep e está aguardando o tempo limite de suspensão expirar. Para obter mais informações, consulte Tempo limite:PgSleep.

Timeout:RecoveryApplyDelay

Um processo está aguardando para aplicar o log de gravação antecipada (WAL) durante a recuperação devido a uma configuração de atraso.

Timeout:RecoveryRetrieveRetryInterval

Um processo está aguardando durante a recuperação quando os dados do log de gravação antecipada (WAL) não estão disponíveis em nenhuma origem (pg_wal, arquivo ou fluxo).

Timeout:VacuumDelay

Um processo está aguardando em um ponto de atraso de vacuum baseado em custos.

Para obter uma lista completa de eventos de espera do PostgreSQL, consulte The Statistics Collector > Wait Event tables (Coletor de estatísticas > Tabela de eventos de espera), na documentação do PostgreSQL.

Atualizações do mecanismo de banco de dados do Amazon Aurora PostgreSQL

Depois disso, é possível encontrar informações sobre as atualizações e as versões do mecanismo Amazon Aurora PostgreSQL. Descubra também como atualizar a versão Amazon Aurora PostgreSQL do mecanismo. Para obter mais informações sobre as versões do Aurora em geral, consulte Versões do Amazon Aurora.

> **Tip**
>
> Você pode minimizar o tempo de inatividade necessário para a atualização do cluster de banco de dados usando uma implantação azul/verde. Para obter mais informações, consulte Usar implantações azul/verde para atualizações de banco de dados.

Tópicos

- Identificar as versões do Amazon Aurora PostgreSQL
- Versões Amazon Aurora PostgreSQL e versões do mecanismo
- Versões de extensão para o Amazon Aurora PostgreSQL
- Como atualizar os clusters de banco de dados de Amazon Aurora MySQL
- Usar uma versão de suporte de longo prazo (LTS) do Aurora PostgreSQL

Identificar as versões do Amazon Aurora PostgreSQL

O Amazon Aurora contém determinados recursos que são gerais para o Aurora e estão disponíveis para todos os clusters de bancos de dados Aurora. O Aurora contém outros recursos que são específicos de um mecanismo de banco de dados compatível com o Aurora. Esses recursos estão disponíveis somente para esses clusters de bancos de dados Aurora que usam o mecanismo de banco de dados, como o Aurora PostgreSQL.

Um banco de dados do Aurora normalmente tem dois números de versão, o número da versão do mecanismo de banco de dados e o número da versão do Aurora. Se uma versão do Aurora PostgreSQL tiver um número de versão do Aurora, ela será incluída após o número da versão do mecanismo na listagem Versões Amazon Aurora PostgreSQL e versões do mecanismo.

Tópicos

- Número de versão do Aurora
- Número de versão do mecanismo do PostgreSQL

Número de versão do Aurora

Os números de versão do Aurora usam o esquema de nomeação *major*.*minor*.*patch*. Uma versão do patch do Aurora inclui correções de bugs importantes adicionadas a uma versão secundária após o lançamento. Para saber mais sobre as versões principal, secundária e patch do Amazon Aurora, consulte Versões principais do Amazon Aurora, Versões secundárias do Amazon Aurora e Versões de patch do Amazon Aurora.

Você pode descobrir o número de versão do Aurora da instância de banco de dados Aurora PostgreSQL com a consulta SQL a seguir:

```
postgres=> SELECT aurora_version();
```

A partir das versões 13.3, 12.8, 11.13, 10.18 do PostgreSQL e para todas as outras posteriores, os números de versão do Aurora se alinham mais de perto com a versão do mecanismo do PostgreSQL. Por exemplo, consultar um cluster de bancos de dados Aurora PostgreSQL 13.3 retorna o seguinte:

```
aurora_version
----------------
 13.3.1
(1 row)
```

Versões anteriores, como o cluster de bancos de dados Aurora PostgreSQL 10.14, retornam números de versão semelhantes aos seguintes:

```
aurora_version
----------------
 2.7.3
(1 row)
```

Número de versão do mecanismo do PostgreSQL

A partir do PostgreSQL 10, as versões do mecanismo de banco de dados PostgreSQL usam um esquema de numeração *major*.*minor* para todas as versões. Alguns exemplos incluem PostgreSQL 10.18, PostgreSQL 12.7 e PostgreSQL 13.3.

Versões anteriores ao PostgreSQL 10 usavam um esquema de numeração *major*.*major*.*minor* em que os dois primeiros dígitos compõem o número da versão principal e um terceiro dígito denota uma versão menor. Por exemplo, o PostgreSQL 9.6 era uma versão principal, com versões secundárias 9.6.21 ou 9.6.22 indicadas pelo terceiro dígito.

> **Note**
>
> O mecanismo PostgreSQL versão 9.6 não tem mais suporte. Para atualizar, consulte Como atualizar os clusters de banco de dados de Amazon Aurora MySQL. Para obter versões, políticas e cronogramas de lançamento, consulte Por quanto tempo as versões principais do Amazon Aurora permanecem disponíveis.

Você pode descobrir o número da versão do mecanismo de banco de dados PostgreSQL com a seguinte consulta SQL:

```
postgres=> SELECT version();
```

Para um cluster de banco de dados Aurora PostgreSQL 13.3, os resultados são os seguintes:

```
version
------------------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
(1 row)
```

Versões Amazon Aurora PostgreSQL e versões do mecanismo

As versões da edição do Amazon Aurora compatível com PostgreSQL são atualizadas regularmente. Essas atualizações são aplicadas a clusters de banco de dados do Aurora PostgreSQL durante as janelas de manutenção do sistema. A aplicação das atualizações depende do tipo de atualização, da Região da AWS e da configuração da janela de manutenção para o cluster de banco de dados. Muitas das versões listadas incluem um número de versão do PostgreSQL e um número de versão do Amazon Aurora. Contudo, a partir das versões 13.3, 12.8, 11.13, 10.18 do PostgreSQL e para todas as outras versões posteriores, os números de versão do Aurora não são utilizados. Para identificar os números de versão de seu banco de dados do Aurora PostgreSQL, consulte Identificar as versões do Amazon Aurora PostgreSQL.

Para obter informações sobre extensões e módulos, consulte Versões de extensão para o Amazon Aurora PostgreSQL.

> **Note**
>
> Para obter informações sobre as versões, as políticas e os cronogramas de lançamento do Amazon Aurora, consulte Por quanto tempo as versões principais do Amazon Aurora permanecem disponíveis.

Para obter informações sobre o suporte ao Amazon Aurora, consulte Perguntas frequentes sobre o Amazon RDS.

Para determinar quais versões do mecanismo PostgreSQL estão disponíveis em uma Região da AWS, use o comando describe-db-engine-versions da AWS CLI. Por exemplo:

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region aws-region
```

Para obter uma lista de Regiões da AWS, consulte Disponibilidade de regiões do Aurora PostgreSQL.

Para obter detalhes sobre as versões do PostgreSQL disponíveis no Aurora PostgreSQL, consulte Notas de lançamento do Aurora PostgreSQL.

Versões de extensão para o Amazon Aurora PostgreSQL

Você pode instalar e configurar várias extensões do PostgreSQL para serem utilizadas com clusters de banco de dados do Aurora PostgreSQL. Por exemplo, você pode usar a extensão `pg_partman` do PostgreSQL para automatizar a criação e a manutenção de partições de tabelas. Para saber mais sobre essa e outras extensões disponíveis para o Aurora PostgreSQL, consulte Trabalhar com extensões e invólucros de dados externos.

Para obter detalhes sobre as extensões do PostgreSQL compatíveis com o Aurora PostgreSQL, consulte Versões de extensão para o Amazon Aurora PostgreSQL em Notas de lançamento do Aurora PostgreSQL.

Como atualizar os clusters de banco de dados de Amazon Aurora MySQL

O Amazon Aurora disponibiliza novas versões do mecanismo de banco de dados do PostgreSQL em Regiões da AWS somente após testes extensivos. Você poderá atualizar seus clusters de banco de dados do Aurora PostgreSQL para a nova versão quando ela estiver disponível em sua região.

Dependendo da versão do Aurora PostgreSQL em que seu cluster de banco de dados está em execução no momento, uma atualização para a nova versão é secundária ou principal. Por exemplo, a atualização de um cluster de banco de dados do Aurora PostgreSQL 11.15 para o Aurora PostgreSQL 13.6 é uma atualização da versão principal. Por exemplo, a atualização de um cluster de banco de dados do Aurora PostgreSQL 11.15 para o Aurora PostgreSQL 13.6 é uma atualização da versão secundária. Nos tópicos a seguir, você encontrará informações sobre como realizar os dois tipos de atualizações.

Sumário

- Visão geral dos processos de atualização do Aurora PostgreSQL
- Obter uma lista de versões disponíveis em sua Região da AWS
- Realizar uma atualização da versão principal
  - Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal
  - Recomendações pós-atualização
  - Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal
    - Atualizações principais de bancos de dados globais
- Realizar uma atualização da versão secundária
  - Antes de realizar um upgrade da versão secundária
  - Como realizar atualizações de versão secundária e aplicar patches
  - Atualizações de versões secundárias e patches com tempo de inatividade zero
  - Limitações da aplicação de patches com tempo de inatividade zero
  - Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão secundária
- Atualizar extensões do PostgreSQL
- Técnica alternativa de atualização azul-verde

Visão geral dos processos de atualização do Aurora PostgreSQL

As diferenças entre atualizações de versão principal e secundária são as seguintes:

Patches e atualizações de versões secundárias

Patches e atualizações de versões secundárias incluem somente alterações compatíveis com versões anteriores das aplicações existentes. Patches e atualizações de versões secundárias ficam disponíveis para você somente depois que o Aurora PostgreSQL os testa e os aprova.

O Aurora pode aplicar atualizações de versões secundárias automaticamente para você. Quando você cria um cluster de banco de dados do PostgreSQL do Aurora, a opção Habilitar o upgrade automático da versão secundária é habilitada por padrão. A menos que você desative manualmente essa opção, o Aurora aplicará atualizações automáticas de versões secundárias durante a janela de manutenção programada. Para obter mais informações sobre a opção de atualização automática de versão secundária (AmVU) e como modificar o cluster de banco de dados do Aurora para usá-lo, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.

Se a atualização automática de versões secundárias não estiver habilitada para seu cluster de banco de dados do Aurora PostgreSQL, o Aurora PostgreSQL não será atualizado automaticamente para a nova versão secundária. Em vez disso, quando uma nova versão secundária é lançada em sua Região da AWS e o cluster de banco de dados do Aurora PostgreSQL está executando uma versão secundária mais antiga, o Aurora solicita a atualização. Ele faz isso adicionando uma recomendação às tarefas de manutenção do cluster.

Os patches não são considerados atualizações e não são aplicados automaticamente. O Aurora PostgreSQL solicita que você aplique todos os patches adicionando uma recomendação às tarefas de manutenção do cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Como realizar atualizações de versão secundária e aplicar patches.

> **Note**
>
> Patches que resolvem problemas de segurança ou outros problemas críticos também são adicionados como tarefas de manutenção. No entanto, esses patches são necessários. Aplique patches de segurança ao cluster de banco de dados do Aurora PostgreSQL quando eles estiverem disponíveis em suas tarefas de manutenção pendentes.
> As atualizações automáticas de versões secundárias são realizadas para a versão secundária padrão.

O processo de atualização envolve a possibilidade de breves interrupções à medida que cada instância no cluster é atualizada para a nova versão. No entanto, após o Aurora PostgreSQL

14.3.3, 13.7.3, 12.11.3, 11.16.3, 10.21.3, bem como outras versões posteriores dessas versões secundárias e principais mais recentes, o processo de atualização usa o recurso ZDP (zero-downtime patch). Esse recurso minimiza as interrupções e, na maioria dos casos, as elimina completamente. Para obter mais informações, consulte Atualizações de versões secundárias e patches com tempo de inatividade zero. Consulte mais informações sobre os recursos compatíveis e as limitações do ZDP em Limitações da aplicação de patches com tempo de inatividade zero.

Atualizações de versão principal

Ao contrário de atualizações e patches de versões secundárias, o Aurora PostgreSQL não tem uma opção automática de atualização de versão principal. Novas versões principais do PostgreSQL podem conter alterações de banco de dados incompatíveis com versões anteriores das aplicações existentes. A nova funcionalidade pode fazer com que suas aplicações existentes parem de funcionar corretamente.

Para evitar problemas, é altamente recomendável seguir o processo descrito em Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal antes de atualizar as instâncias de banco de dados em seus clusters de banco de dados do Aurora PostgreSQL. Primeiro, verifique se suas aplicações podem ser executadas na nova versão seguindo esse procedimento. Depois, você pode atualizar manualmente o cluster de banco de dados do Aurora PostgreSQL para a nova versão.

O processo de upgrade envolve a possibilidade de uma breve interrupção quando todas as instâncias no cluster são atualizadas para a nova versão. O processo de planejamento preliminar também leva tempo. Recomendamos que você sempre execute tarefas de atualização durante a janela de manutenção do cluster ou quando as operações forem mínimas. Para ter mais informações, consulte Realizar uma atualização da versão principal.

> **Note**
>
> Tanto as atualizações de versões secundárias quanto as atualizações de versões principais podem envolver breves interrupções. Por esse motivo, recomendamos que você execute ou programe atualizações durante sua janela de manutenção ou durante outros períodos de baixa utilização.

Os clusters de banco de dados do Aurora PostgreSQL ocasionalmente exigem atualizações do sistema operacional. Essas atualizações às vezes incluem uma versão mais recente da biblioteca glibc. Durante essas atualizações, recomendamos que você siga as diretrizes descritas em Agrupamentos permitidos no Aurora PostgreSQL..

Obter uma lista de versões disponíveis em sua Região da AWS

Você pode obter uma lista de todas as versões do mecanismo disponíveis como destinos de upgrade para seu cluster de banco de dados do Aurora PostgreSQL consultando sua Região da AWS usando o comando describe-db-engine-version da AWS CLI conforme mostrado a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Por exemplo, para identificar os destinos de upgrade válidos para um cluster de banco de dados do Aurora PostgreSQL versão 12.10, execute o seguinte comando da AWS CLI:

Para Linux, macOS ou Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version 12.10 \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version 12.10 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Na tabela a seguir, você pode encontrar os destinos de atualização de versões principais e secundárias para diferentes versões de banco de dados do Aurora PostgreSQL. Para manter a compatibilidade, nem todas as versões são oferecidas como destinos de atualização. O Aurora PostgreSQL apresenta novos recursos e correções de erros a cada lançamento trimestral de uma versão secundária. Para ter informações sobre as versões secundárias do Aurora PostgreSQL, consulte as Release Notes for Aurora PostgreSQL.

Versão de origem atual	Destinos de atualização
17.7	Nenhum
17.6	17.7
17.5	17.7, 17.6
17.4	17.7, 17.6, 17.5
16.11	Nenhum
16.10	17.7 16.11
16.9	17.7 16.11, 17.6, 16.10, 17.5
16,8	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9
16.6	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9, 16.8

Versão de origem atual	Destinos de atualização
16.4	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9, 16.8, 16.6
16.3	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9, 16.8, 16.6, 16.4
16.2	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9, 16.8, 16.6, 16.4, 16.3
16.1	17.7 16.11, 17.6, 16.10, 17.5, 17.4, 16.9, 16.8, 16.6, 16.4, 16.3, 16.2
15.15	Nenhum
15.14	17.7 16.11 15.15
15.13	17.7 16.11, 15.15, 17.6, 16.10, 15.14, 17.5, 16.9
15.12	17.7 16.11, 15.15, 17.6, 16.10, 15.14, 15.13
15.10	17.7 16.11, 15.15, 17.6, 16.10, 15.14, 17.5, 17.4, 16.9, 16.8, 15.13
15.11	17.7 16.11, 15.15, 17.6, 16.10, 15.14, 17.5, 17.4, 16.9, 16.8, 16.6, 15.13, 15.12

Versão de origem atual	Destinos de atualização
14.20	Nenhum

Para qualquer versão que você esteja considerando, verifique sempre a disponibilidade da classe de instância de banco de dados do cluster. Por exemplo, o `db.r4` não é compatível com o Aurora PostgreSQL 13. Se seu cluster de banco de dados do Aurora PostgreSQL atualmente usa uma classe de instância db.r4, você precisará modificá-lo para usar uma classe de instância de banco de dados aceita para poder realizar a atualização para o Aurora PostgreSQL 13. Para ter mais informações sobre classes de instância de banco de dados do Aurora PostgreSQL, incluindo quais são baseadas em Graviton2 e quais são baseadas em Intel, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Realizar uma atualização da versão principal

As atualizações da versão principal podem conter alterações de banco de dados incompatíveis com as versões anteriores do banco de dados. A nova funcionalidade em uma nova versão pode fazer com que suas aplicações existentes parem de funcionar corretamente. Para evitar problemas, o Amazon Aurora não aplica atualizações da versão principal automaticamente. Em vez disso, recomendamos planejar cuidadosamente uma atualização de versão principal seguindo estas etapas:

1. Escolha na lista de destinos disponíveis a versão principal desejada das listadas para sua versão na tabela. Você pode obter uma lista precisa de versões disponíveis em suaRegião da AWS para sua versão atual usando a AWS CLI. Para obter detalhes, consulte Obter uma lista de versões disponíveis em sua Região da AWS.
2. Verifique se suas aplicações funcionam conforme o esperado em uma implantação de avaliação da nova versão. Para obter informações sobre o processo completo, consulte Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal.
3. Depois de verificar se suas aplicações funcionam conforme o esperado na implantação de avaliação, você pode atualizar seu cluster. Para obter detalhes, consulte Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal.

> **Note**
>
> Você pode realizar uma atualização da versão principal de versões 13 do Babelfish para Aurora PostgreSQL a partir da 13.6 para versões 14 do Aurora PostgreSQL a partir da 14.6. As versões 13.4 e 13.5 do Babelfish para Aurora PostgreSQL não oferecem suporte a atualizações da versão principal.

Você pode obter uma lista das versões do mecanismo disponíveis como destinos de upgrade da versão principal para seu cluster de banco de dados do Aurora PostgreSQL consultando sua Região da AWS usando o comando describe-db-engine-version da AWS CLI conforme mostrado a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}" ^
  --output text
```

Em alguns casos, a versão para a qual você deseja atualizar não é um destino para sua versão atual. Nesses casos, use as informações na versions table para realizar atualizações de versões secundárias até que o cluster esteja em uma versão que tenha o destino escolhido em sua linha de destinos.

Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal

Toda nova versão principal inclui melhorias no otimizador de consulta projetadas para aprimorar a performance. No entanto, sua workload pode incluir consultas que resultem em um plano de performance inferior na nova versão. É por isso que recomendamos que você teste e avalie a

performance antes de atualizar na produção. Você pode gerenciar a estabilidade do plano de consulta entre versões usando a extensão Query Plan Management (QPM), conforme detalhado em Garantir a estabilidade do plano após a atualização da versão principal.

Antes de atualizar os clusters de banco de dados do Aurora PostgreSQL de produção para uma nova versão principal, é altamente recomendável que você teste a atualização para verificar se todas as suas aplicações funcionam corretamente:

1. Tenha um grupo de parâmetros compatível com a versão pronto para uso.

   Se você estiver usando um grupo de parâmetros de instância de banco de dados ou de cluster de banco de dados personalizado, poderá escolher entre duas opções:

   a. Especifique a instância de banco de dados padrão, o grupo de parâmetros do cluster de banco de dados, ou ambos, para a nova versão do mecanismo do banco de dados.

   b. Crie seu próprio grupo de parâmetros personalizado para a nova versão do mecanismo de banco de dados.

2. Verifique se há bancos de dados inválidos e exclua os existentes.

   A coluna `datconnlimit` no catálogo `pg_database` inclui o valor `-2` para marcar como inválidos qualquer banco de dados que foi interrompido durante uma operação `DROP DATABASE`. Use a consulta a seguir para verificar se há bancos de dados inválidos.

```
SELECT
    datname
FROM
    pg_database
WHERE
    datconnlimit = - 2;
```

   Se a consulta retornar nomes de bancos de dados, esses bancos de dados são inválidos. Use a instrução `DROP DATABASE invalid_db_name` para excluir bancos de dados inválidos. Você pode usar a instrução dinâmica a seguir para excluir todos os bancos de dados inválidos.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

3. Verifique o uso sem suporte:

   - Confirme ou reverta todas as transações preparadas abertas antes de tentar uma atualização. Você pode usar a consulta a seguir para verificar se não há transações preparadas abertas na sua instância.

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

   - Remova todos os usos dos tipos de dados reg* antes de tentar fazer uma atualização. Exceto por `regtype` e `regclass`, você não pode atualizar os tipos de dados reg*. O utilitário pg_upgrade (usado pelo Amazon Aurora para fazer a atualização) não pode persistir esse tipo de dados. Para saber mais sobre esse utilitário, consulte pg_upgrade na documentação do PostgreSQL.

     Para verificar se não há nenhum uso de tipos de dados reg* sem suporte, use a consulta a seguir para cada banco de dados.

     ```
     SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
      pg_catalog.pg_attribute a
       WHERE c.oid = a.attrelid
           AND NOT a.attisdropped
           AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                              'pg_catalog.regprocedure'::pg_catalog.regtype,
                              'pg_catalog.regoper'::pg_catalog.regtype,
                              'pg_catalog.regoperator'::pg_catalog.regtype,
                              'pg_catalog.regconfig'::pg_catalog.regtype,
                              'pg_catalog.regdictionary'::pg_catalog.regtype)
           AND c.relnamespace = n.oid
           AND n.nspname NOT IN ('pg_catalog', 'information_schema');
     ```

   - Se você estiver atualizando do Aurora PostgreSQL versão 10.18 ou posterior e tiver a extensão `pgRouting` instalada, remova essa extensão antes de atualizar para a versão 12.4 ou posterior.

     Se você estiver fazendo upgrade do Aurora PostgreSQL versão 10.x com a extensão `pg_repack` versão 1.4.3 instalada, remova essa extensão antes de fazer upgrade para qualquer versão superior.

4. Verifique os bancos de dados do modelo 1 e do modelo 0.

   Para uma atualização bem-sucedida, os bancos de dados do modelo 1 e do modelo 0 devem existir e devem ser listados como um modelo. Para conferir isso, use o seguinte comando:

```
SELECT datname, datistemplate FROM pg_database;

datname    | datistemplate
-----------+---------------
template0  | t
rdsadmin   | f
template1  | t
postgres   | f
```

Na saída do comando, o valor `datistemplate` dos bancos de dados do modelo 1 e do modelo 0 deve ser `t`.

5. Descarte slots de replicação lógica.

   O processo de upgrade não poderá continuar se o cluster de banco de dados do Aurora PostgreSQL estiver usando qualquer slot de replicação lógica. Os slots de replicação lógica são normalmente usados para tarefas de migração de dados de curto prazo, como a migração de dados usando o AWS DMS, ou para replicar tabelas do banco de dados para data lakes, ferramentas de BI ou outros destinos. Antes de atualizar, certifique-se de saber a finalidade de qualquer slot de replicação lógica existente e confirme se não há problema em excluí-los. Você pode verificar os slots de replicação lógica usando a seguinte consulta:

   ```
   SELECT * FROM pg_replication_slots;
   ```

   Se os slots de replicação lógica ainda estiverem sendo usados, você não deve excluí-los e não poderá continuar com a atualização. No entanto, se os slots de replicação lógica não forem necessários, você poderá excluí-los usando o seguinte SQL:

   ```
   SELECT pg_drop_replication_slot(slot_name);
   ```

   Os cenários de replicação lógica que usam a extensão `pglogical` também precisam ter slots retirados do nó do editor para uma atualização bem-sucedida da versão principal nesse nó. No entanto, você pode reiniciar o processo de replicação a partir do nó do assinante após a atualização. Para ter mais informações, consulte Restabelecer a replicação lógica após uma atualização principal.

6. Execute um backup.

O processo de atualização cria um snapshot do cluster de banco de dados durante a atualização. Se você também quiser realizar um backup manual antes do processo de atualização, consulte Criar um snapshot de cluster de banco de dados para obter mais informações.

7. Atualize determinadas extensões para a versão mais recente disponível antes de executar a atualização da versão principal. As extensões a serem atualizadas incluem as seguintes:
   - `pgRouting`
   - `postgis_raster`
   - `postgis_tiger_geocoder`
   - `postgis_topology`
   - `address_standardizer`
   - `address_standardizer_data_us`

   Execute o comando a seguir para cada extensão instalada.

   ```
   ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
   ```

   Para ter mais informações, consulte Atualizar extensões do PostgreSQL. Para saber mais sobre como fazer upgrade do PostGIS, consulte Etapa 6: Atualize a extensão PostGIS.
8. Se você estiver atualizando para a versão 11.x, descarte as extensões não compatíveis antes de executar a atualização da versão principal. As extensões a serem descartadas incluem:
   - `chkpass`
   - `tsearch2`
9. Descarte os tipos de dados `unknown` de acordo com a versão de destino.

   O PostgreSQL versão 10 não dá suporte ao tipo de dados `unknown`. Se um banco de dados versão 9.6 utilizar o tipo de dados `unknown`, uma atualização para uma versão 10 exibirá uma mensagem de erro como a seguinte:

   ```
   Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
   The instance could not be upgraded because the 'unknown' data type is used in user
    tables.
   Please remove all usages of the 'unknown' data type and try again."
   ```

   Para localizar o tipo de dados `unknown` no banco de dados a fim de remover coluna incorreta ou alterar para um tipo de dados compatível, use o código SQL a seguir para cada banco de dados.

```
SELECT n.nspname, c.relname, a.attname
    FROM pg_catalog.pg_class c,
    pg_catalog.pg_namespace n,
    pg_catalog.pg_attribute a
    WHERE c.oid = a.attrelid AND NOT a.attisdropped AND
    a.atttypid = 'pg_catalog.unknown'::pg_catalog.regtype AND
    c.relkind IN ('r','m','c') AND
    c.relnamespace = n.oid AND
    n.nspname !~ '^pg_temp_' AND
    n.nspname !~ '^pg_toast_temp_' AND n.nspname NOT IN ('pg_catalog',
 'information_schema');
```

10 Execute uma simulação de atualização.

É altamente recomendável testar uma atualização da versão principal em uma cópia do seu banco de dados de produção antes de fazer a atualização no banco de dados de produção. Você pode monitorar os planos de execução na instância de teste duplicada para detectar possíveis regressões do plano de execução e avaliar sua performance. Para criar uma instância de teste duplicada, restaure o banco de dados a partir de um snapshot recente ou clone o banco de dados. Para ter mais informações, consulte Restauração a partir de um snapshot ou Clonar um volume para um cluster de banco de dados do Amazon Aurora.

Para ter mais informações, consulte Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal.

11 Atualize sua instância de produção.

Se a simulação da atualização da versão principal for bem-sucedida, você será capaz de atualizar seu banco de dados de produção com segurança. Para obter mais informações, consulte Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal.

> **Note**
>
> Durante o processo de atualização, o Aurora PostgreSQL cria um snapshot do cluster de banco de dados. Não é possível fazer uma restauração a um ponto anterior no tempo do cluster durante esse processo. Posteriormente, você pode fazer uma restauração a um ponto anterior no tempo (para momentos antes da atualização) e depois da conclusão do

snapshot automático da sua instância. No entanto, não é possível fazer uma restauração a um ponto anterior no tempo para uma versão secundária anterior.

Para obter informações sobre uma atualização em andamento, você pode usar o Amazon RDS para visualizar dois logs que são produzidos pelo utilitário pg_upgrade. Esses logs são `pg_upgrade_internal.log` e `pg_upgrade_server.log`. O Amazon Aurora acrescenta a data e a hora ao nome de arquivo desses logs. Você pode visualizar esses logs como visualiza qualquer outro log. Para ter mais informações, consulte Monitorar arquivos de log do Amazon Aurora.

12 Atualizar extensões do PostgreSQL. O processo de atualização do PostgreSQL não atualiza nenhuma extensão do PostgreSQL. Para obter mais informações, consulte Atualizar extensões do PostgreSQL.

Recomendações pós-atualização

Após a conclusão da atualização da versão principal, recomendamos o seguinte:

- Execute a operação `ANALYZE` para atualizar a tabela `pg_statistic`. Você deve fazer isso para cada banco de dados em todas as suas instâncias de banco de dados PostgreSQL. As estatísticas do otimizador não são transferidas durante uma atualização de versão principal, portanto, você precisa gerar novamente todas as estatísticas para evitar problemas de performance. Execute o comando sem nenhum parâmetro para gerar estatísticas para todas as tabelas regulares no banco de dados atual da seguinte forma:

```
ANALYZE VERBOSE;
```

O sinalizador `VERBOSE` é opcional, mas usá-lo mostra o progresso. Para obter mais informações, consulte ANALYZE na documentação do PostgreSQL.

Ao analisar tabelas específicas em vez de usar ANALYZE VERBOSE, execute o comando ANALYZE para cada tabela da seguinte forma:

```
ANALYZE table_name;
```

Para tabelas particionadas, sempre analise a tabela principal. Esse processo:

- Examina as linhas por amostragem automaticamente em todas as partições.

- Atualiza as estatísticas de cada partição recursivamente.
- Mantém estatísticas essenciais de planejamento no nível da tabela principal.

Embora as tabelas principais não armazenem dados reais, analisá-las é vital para otimizar as consultas. Executar ANALYZE somente em partições individuais pode levar a um desempenho de consulta insatisfatório, pois o otimizador não terá as estatísticas abrangentes necessárias para um planejamento eficiente entre partições.

> **Note**
>
> Execute ANALYZE em seu sistema após a atualização para evitar problemas de performance.

- Se você atualizou para o PostgreSQL versão 10, execute `REINDEX` em todos os índices de hash que tiver. Os índices de hash foram alterados na versão 10 e devem ser recriados. Para localizar índices de hash inválidos, execute o SQL a seguir para cada banco de dados que contém índices de hash.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

- Recomendamos testar a aplicação no banco de dados atualizado com uma workload semelhante para verificar se tudo funciona como esperado. Verificada a atualização, é possível excluir essa instância de teste.

Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal

Quando você inicia o processo de atualização para uma nova versão principal, o Aurora PostgreSQL tira um snapshot do cluster de banco de dados do Aurora antes de fazer qualquer alteração no cluster. Esse snapshot é criado apenas para atualizações de versão principal, não para atualizações de versão secundária. Quando o processo de atualização for concluído, você poderá encontrar esse snapshot entre os snapshots manuais listados em Snapshots no console do RDS. O nome do snapshot inclui `preupgrade` como prefixo, o nome do cluster de banco de dados do Aurora

PostgreSQL, a versão de origem, a versão de destino e a data e o carimbo de data e hora, conforme mostrado no exemplo a seguir.

```
preupgrade-docs-lab-apg-global-db-12-8-to-13-6-2022-05-19-00-19
```

Após a conclusão da atualização, você poderá usar o snapshot que o Aurora criou e armazenou na sua lista de snapshots manuais para restaurar o cluster de banco de dados para sua versão anterior, se necessário.

> **Tip**
>
> Em geral, os snapshots fornecem várias maneiras de restaurar seu cluster de banco de dados do Aurora para vários pontos no tempo. Para saber mais, consulte Restauração de um snapshot de um cluster de banco de dados e Restaurar um cluster de banco de dados para um horário especificado. No entanto, o Aurora PostgreSQL não oferece suporte ao uso de um snapshot para restaurar uma versão secundária anterior.

Durante o processo de atualização da versão principal, o Aurora aloca um volume e clona o cluster de banco de dados Aurora PostgreSQL de origem. Se a atualização falhar por qualquer motivo, o Aurora PostgreSQL usará o clone para reverter a atualização. Depois que mais de 15 clones de um volume de origem são alocados, os clones subsequentes se tornam cópias completas e levam mais tempo. Isso pode fazer com que o processo de atualização demore mais tempo. Se o Aurora PostgreSQL reverter a atualização, lembre-se de que:

- Você poderá ver entradas de faturamento e métricas do volume original e do volume clonado alocados durante a atualização. O Aurora PostgreSQL limpará o volume extra depois que a janela de retenção do backup do cluster for superior ao tempo da atualização.
- A próxima cópia do snapshot de região cruzada desse cluster será uma cópia completa em vez de uma cópia incremental.

Para atualizar com segurança as instâncias de banco de dados que compõem o cluster, o Aurora PostgreSQL usa o utilitário pg_upgrade. Após a conclusão da atualização do gravador, cada instância do leitor passa por uma breve interrupção durante a atualização para a nova versão principal. Para saber mais sobre esse utilitário do PostgreSQL, consulte pg_upgrade na documentação do PostgreSQL.

Você pode atualizar o cluster de banco de dados do Aurora PostgreSQL para uma nova versão usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como atualizar a versão do mecanismo de um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e o cluster de banco de dados que você deseja atualizar.
3. Selecione Modify. A página Modify DB cluster (Modificar cluster de banco de dados) é exibida.
4. Em Engine version (Versão do mecanismo), escolha a nova versão.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. Para aplicar as alterações imediatamente, escolha Apply immediately. Escolher essa opção pode causar uma interrupção em alguns casos. Para ter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify cluster (Modificar cluster) para salvar suas alterações.

   Ou selecione Back (Voltar) para editar as alterações ou Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para atualizar a versão de mecanismo de um cluster de banco de dados, use o comando modify-db-cluster da AWS CLI. Especifique os seguintes parâmetros:

- `--db-cluster-identifier`: o nome do cluster de banco de dados.
- `--engine-version`: o número da versão do mecanismo de banco de dados para a qual será feita a atualização. Para obter informações sobre versões de mecanismo válidas, use o comando AWS CLI describe-db-engine-versions.
- `--allow-major-version-upgrade`: um sinalizador obrigatório quando o parâmetro `--engine-version` for uma versão principal diferente da versão principal atual do cluster de banco de dados.
- `--no-apply-immediately`: aplica as alterações durante a próxima janela de manutenção. Para aplicar as alterações imediatamente, use `--apply-immediately`.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API do RDS

Para atualizar a versão do mecanismo de um cluster de banco de dados, use a operação ModifyDBCluster. Especifique os seguintes parâmetros:

- `DBClusterIdentifier` – o nome do cluster de banco de dados. Por exemplo *`mydbcluster`*.
- `EngineVersion`: o número da versão do mecanismo de banco de dados para a qual será feita a atualização. Para obter informações sobre versões de mecanismo válidas, use a operação DescribeDBEngineVersions.
- `AllowMajorVersionUpgrade`: um sinalizador obrigatório quando o parâmetro `EngineVersion` for uma versão principal diferente da versão principal atual do cluster de banco de dados.
- `ApplyImmediately`: se deseja ou não aplicar as alterações imediatamente ou durante a próxima janela de manutenção. Para aplicar as alterações imediatamente, defina o valor como `true`. Para aplicar alterações durante a próxima janela de manutenção, defina o valor como `false`.

Atualizações principais de bancos de dados globais

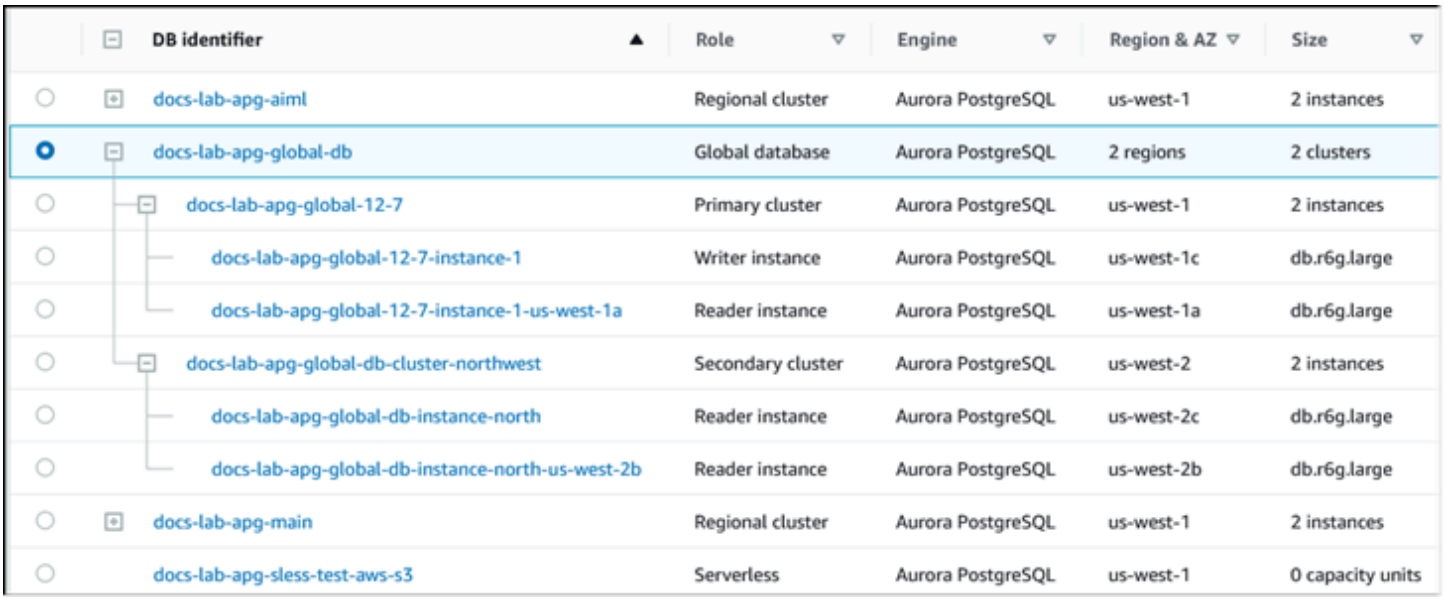
Para um cluster de banco de dados global do Aurora, o processo de atualização atualiza simultaneamente todos os clusters de banco de dados que compõem seu banco de dados global do Aurora. Ele faz isso para garantir que cada um execute a mesma versão do Aurora PostgreSQL. Isso

também garante que todas as alterações nas tabelas do sistema, em formatos de arquivo de dados, etc. sejam replicadas automaticamente para todos os clusters secundários.

Para atualizar um cluster de banco de dados global para uma nova versão principal do Aurora PostgreSQL, recomendamos que você teste suas aplicações na versão atualizada, conforme detalhado em Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal. Prepare as configurações do grupo de parâmetros de cluster de banco de dados e do grupo de parâmetros de banco de dados para cada Região da AWS em seu banco de dados global do Aurora antes da atualização, conforme detalhado em step 1. do Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal.

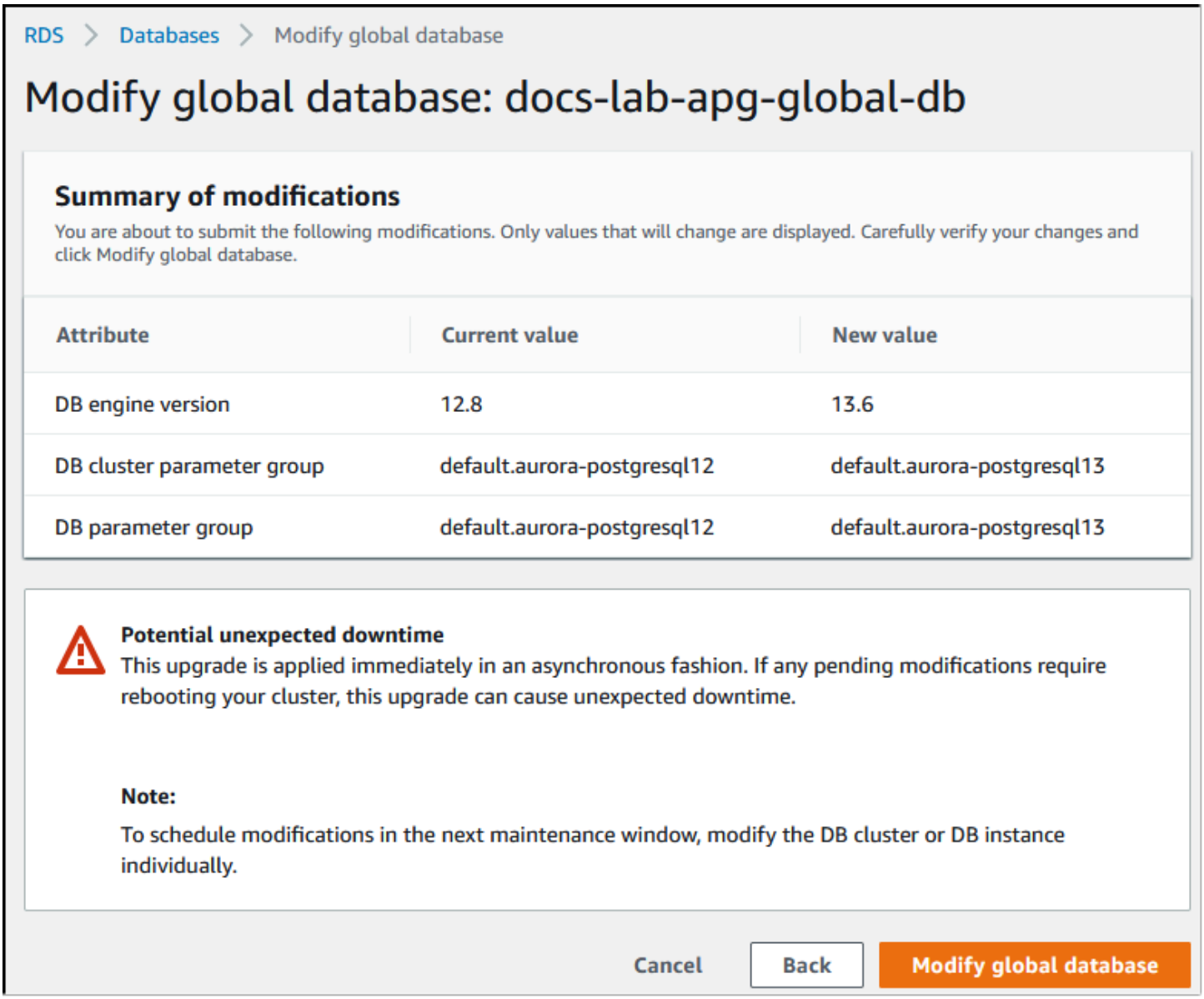
Se o cluster de banco de dados global do Aurora PostgreSQL tiver um objetivo de ponto de recuperação (RPO) definido para o parâmetro `rds.global_db_rpo`, redefina o parâmetro antes de atualizar. O processo de atualização da versão principal não funcionará se o RPO estiver ativado. Por padrão, esse parâmetro permanece desativado. Para ter mais informações sobre bancos de dados globais e RPO do Aurora PostgreSQL, consulte Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–.

Se você confirmar que suas aplicações podem ser executadas conforme o esperado na implantação de avaliação da nova versão, poderá iniciar o processo de atualização. Para fazer isto, consulte Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão principal. Escolha o item de nível superior na lista Databases (Bancos de dados) no console do RDS, Global database (Banco de dados global), conforme mostrado na imagem a seguir.



DB identifier	Role	Engine	Region & AZ	Size
docs-lab-apg-aiml	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-db	Global database	Aurora PostgreSQL	2 regions	2 clusters
docs-lab-apg-global-12-7	Primary cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-12-7-instance-1	Writer instance	Aurora PostgreSQL	us-west-1c	db.r6g.large
docs-lab-apg-global-12-7-instance-1-us-west-1a	Reader instance	Aurora PostgreSQL	us-west-1a	db.r6g.large
docs-lab-apg-global-db-cluster-northwest	Secondary cluster	Aurora PostgreSQL	us-west-2	2 instances
docs-lab-apg-global-db-instance-north	Reader instance	Aurora PostgreSQL	us-west-2c	db.r6g.large
docs-lab-apg-global-db-instance-north-us-west-2b	Reader instance	Aurora PostgreSQL	us-west-2b	db.r6g.large
docs-lab-apg-main	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-sless-test-aws-s3	Serverless	Aurora PostgreSQL	us-west-1	0 capacity units

Como em qualquer modificação, você pode confirmar que deseja que o processo prossiga quando solicitado.



Em vez de usar o console, você pode iniciar o processo de atualização usando a AWS CLI ou a API do RDS. Assim como no console, você trabalha no cluster de banco de dados global do Aurora, não em seus componentes, da seguinte forma:

- Use o comando modify-global-cluster AWS CLI para iniciar a atualização de seu banco de dados global do Aurora usando a AWS CLI.
- Use a API ModificyGlobalCluster para iniciar a atualização.

Realizar uma atualização da versão secundária

É possível usar os seguintes métodos para atualizar a versão secundária de um cluster de banco de dados ou para aplicar um patch em um cluster de banco de dados:

Tópicos

- Antes de realizar um upgrade da versão secundária
- Como realizar atualizações de versão secundária e aplicar patches
- Atualizações de versões secundárias e patches com tempo de inatividade zero
- Limitações da aplicação de patches com tempo de inatividade zero
- Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão secundária

Antes de realizar um upgrade da versão secundária

Recomendamos que você execute as seguintes ações para reduzir o tempo de inatividade durante um upgrade de versão secundária:

- A manutenção do cluster de banco de dados do Aurora deve ser realizada durante um período de baixo tráfego. Use o Insights de Performance para identificar esses períodos a fim de configurar as janelas de manutenção corretamente. Consulte mais informações sobre o Insights de Performance em Monitoring DB load with Performance Insights on Amazon RDS. Consulte mais informações sobre a janela de manutenção do cluster de banco de dados em Ajustar a janela de manutenção do cluster de banco de dados preferencial.
- Use SDKs da AWS compatíveis com oscilações e recuos exponenciais como uma prática recomendada. Consulte mais informações em Exponential Backoff And Jitter.

Como realizar atualizações de versão secundária e aplicar patches

Patches e atualizações de versões secundárias ficam disponíveis em Regiões da AWS somente após testes rigorosos. Antes de lançar atualizações e patches, o Aurora PostgreSQL testa para garantir que problemas de segurança conhecidos, erros e outros problemas que surgem após o lançamento da versão secundária da comunidade não interrompam a estabilidade da frota do Aurora PostgreSQL.

Se você ativar Habilitar o upgrade automático da versão secundária, o Aurora PostgreSQL atualizará periodicamente o cluster de banco de dados durante a janela de manutenção especificada. Verifique

se a opção Habilitar o upgrade automático da versão secundária está ativada para todas as instâncias de banco de em seu cluster de banco de dados do Aurora PostgreSQL. Para obter informações sobre como definir a configuração Upgrade automático de versões secundárias e como ela funciona quando aplicada nos níveis de cluster e instância, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.

Confira o valor da opção Habilitar o upgrade automático da versão secundária para todos os clusters de banco de dados do Aurora PostgreSQL usando o comando describe-db-instances da AWS CLI da forma a seguir.

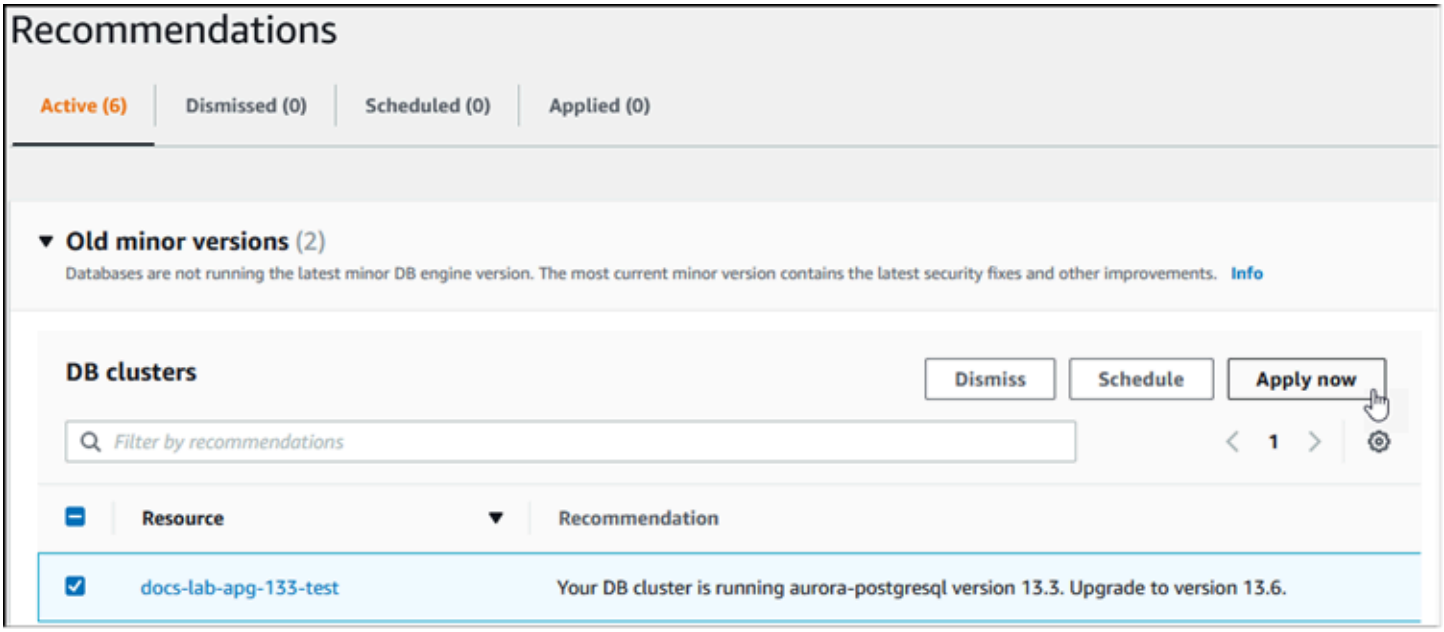
```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Essa consulta retorna uma lista de todos os clusters de banco de dados do Aurora e de suas instâncias com um valor `true` ou `false` para o status da configuração `AutoMinorVersionUpgrade`. O comando pressupõe que você tenha a AWS CLI configurada para a Região da AWS padrão pretendida.

Para obter mais informações sobre a opção AmVU e como modificar o cluster de banco de dados do Aurora para usá-la, consulte Atualizações da versão secundária automáticas para clusters de banco de dados do Aurora.

Você pode atualizar seus clusters de banco de dados do Aurora PostgreSQL para novas versões secundárias respondendo a tarefas de manutenção ou modificando o cluster para usar a nova versão.

Você pode identificar todas as atualizações ou patches disponíveis para seus clusters de banco de dados do Aurora PostgreSQL usando o console do RDS e abrindo o menu Recommendations (Recomendações). Nesse menu, é possível encontrar uma lista de vários problemas de manutenção, como Old minor versions (Versões secundárias antigas). Dependendo do ambiente de produção, você pode optar por programar (Schedule) a atualização ou tomar medidas imediatas, escolhendo Apply now (Aplicar agora), conforme mostrado a seguir.



Para saber mais sobre como manter um cluster de banco de dados do Aurora, inclusive como aplicar manualmente patches e atualizações de versões secundárias, consulte Manutenção de um cluster de banco de dados do Amazon Aurora.

Atualizações de versões secundárias e patches com tempo de inatividade zero

A atualização de um cluster de banco de dados do Aurora PostgreSQL envolve a possibilidade de uma interrupção. Durante o processo de atualização, o banco de dados é desligado à medida que está sendo atualizado. Se você iniciar a atualização enquanto o banco de dados estiver ocupado, perderá todas as conexões e transações que o cluster de banco de dados está processando. Se você esperar até que o banco de dados fique ocioso para realizar a atualização, talvez seja necessário esperar muito tempo.

O recurso ZDP (zero-time patch) melhora o processo de atualização. Com o ZDP, atualizações de versões menores e patches podem ser aplicados com impacto mínimo no cluster de banco de dados do Aurora PostgreSQL. O ZDP é usado ao aplicar patches ou atualizações de versões secundárias mais recentes para o Aurora PostgreSQL e outras versões posteriores dessas versões secundárias e principais mais recentes. Ou seja, a atualização para novas versões secundárias de qualquer uma dessas versões em diante usa o ZDP.

A tabela a seguir mostra as versões do Aurora PostgreSQL e as classes de instâncias de banco de dados em que o ZDP está disponível:

Versão	Classes de instância db.r*	Classes de instância db.t*	Classes de instância db.x*	Classe de instância db.serverless
10.21 e versões posteriores	Sim	Sim	Sim	N/D
11.16 e versões posteriores	Sim	Sim	Sim	N/D
11.17 e versões posteriores	Sim	Sim	Sim	N/D
12.11 e versões posteriores	Sim	Sim	Sim	N/D
12.12 e versões posteriores	Sim	Sim	Sim	N/D
13.7 e versões posteriores	Sim	Sim	Sim	N/D
13.8 e versões posteriores	Sim	Sim	Sim	Sim
14.3 e versões posteriores	Sim	Sim	Sim	N/D
14.4 e versões posteriores	Sim	Sim	Sim	N/D
14.5 e versões posteriores	Sim	Sim	Sim	Sim
15.3 e versões posteriores	Sim	Sim	Sim	Sim
16.1 e versões posteriores	Sim	Sim	Sim	Sim

Durante o processo de upgrade usando o recurso ZDP, o mecanismo de banco de dados procura um ponto tranquilo para pausar todas as novas transações. Essa ação protege o banco de dados durante patches e upgrades. Para garantir que suas aplicações funcionem sem problemas com transações pausadas, recomendamos integrar a lógica de novas tentativas em seu código. Essa abordagem garante que o sistema consiga gerenciar qualquer breve tempo de inatividade sem falhas e possa tentar realizar novamente as novas transações depois do upgrade.

Quando o recurso ZDP é concluído com êxito, as sessões da aplicação são preservadas, exceto aquelas com conexões descartadas, e o mecanismo de banco de dados é reiniciado enquanto o upgrade ainda está em andamento. Embora a reinicialização do mecanismo de banco de dados possa causar uma queda temporária no throughput, em geral, ela dura apenas alguns segundos ou, no máximo, cerca de um minuto.

Em alguns casos, a aplicação de patch de tempo de inatividade zero (ZDP) pode não ter êxito. Por exemplo, as alterações de parâmetros no estado `pending` no cluster de banco de dados do Aurora PostgreSQL ou em suas instâncias interferem no ZDP.

Você pode encontrar métricas e eventos para operações do ZDP na página Events (Eventos) no console. Os eventos incluem o início da atualização do ZDP e a conclusão da atualização. Nesse caso, é possível descobrir quanto tempo o processo demorou e o número de conexões preservadas e descartadas que ocorreram durante a reinicialização. É possível localizar detalhes no log de erros do banco de dados.

Limitações da aplicação de patches com tempo de inatividade zero

As seguintes limitações se referem à aplicação de patches com tempo de inatividade zero:

- O ZDP tenta preservar as conexões atuais do cliente com a instância de gravador do Aurora PostgreSQL durante todo o processo de upgrade do Aurora PostgreSQL. No entanto, nos seguintes casos, as conexões serão interrompidas para que o ZDP seja concluído:
  - Consultas ou transações de longa execução estão em andamento.
  - As instruções Data Definition Language (DDL) estão em execução.
  - As tabelas temporárias ou bloqueios de tabela estão em uso.
  - Todas as sessões estão sendo ouvidas nos canais de notificação.
  - Um cursor no status “'WITH HOLD” está em uso.
  - As conexões TLSv1.1 estão em uso. A partir das versões do Aurora PostgreSQL posteriores a 16.1, 15.3, 14.8, 13.11, 12.15 e 11.20, o ZDP é compatível com conexões TLSv1.3.
- O ZDP não é compatível nos seguintes casos:

- Quando os clusters de banco de dados do Aurora PostgreSQL são configurados como o Aurora Serverless v1.
- Durante o upgrade de qualquer instância de leitor do Aurora.
- Durante o upgrade de qualquer instância de leitor do Aurora que faça parte de um cluster do Aurora Global Database em uma região secundária.
- Durante os patches e atualizações do sistema operacional.

Atualizar o mecanismo do Aurora PostgreSQL para uma nova versão secundária

Você pode atualizar o cluster de banco de dados do Aurora PostgreSQL para uma nova versão secundária usando o console, a AWS CLI ou a API do RDS. Antes de realizar o upgrade, recomendamos seguir as mesmas práticas recomendadas para atualizações de versão principais. Assim como nas novas versões principais, as novas versões secundárias também podem ter melhorias no otimizador, como correções, que podem causar regressões no plano de consulta. Para garantir a estabilidade do plano, recomendamos usar a extensão Query Plan Management (QPM) conforme detalhado em Garantir a estabilidade do plano após a atualização da versão principal.

Console

Como atualizar o mecanismo de seu cluster de banco de dados do Aurora PostgreSQL

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e o cluster de banco de dados que você deseja atualizar.
3. Selecione Modify. A página Modify DB cluster (Modificar cluster de banco de dados) é exibida.
4. Em Engine version (Versão do mecanismo), escolha a nova versão.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. Para aplicar as alterações imediatamente, escolha Apply immediately. Escolher essa opção pode causar uma interrupção em alguns casos. Para ter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora.
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify cluster (Modificar cluster) para salvar suas alterações.

   Ou selecione Back (Voltar) para editar as alterações ou Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para atualizar a versão do mecanismo de um cluster de banco de dados, use o comando modify-db-cluster da AWS CLI com os seguintes parâmetros:

- `--db-cluster-identifier`: o nome de seu cluster de banco de dados do Aurora PostgreSQL.
- `--engine-version`: o número da versão do mecanismo de banco de dados para a qual será feita a atualização. Para obter informações sobre versões de mecanismo válidas, use o comando AWS CLI describe-db-engine-versions.
- `--no-apply-immediately`: aplica as alterações durante a próxima janela de manutenção. Para aplicar as alterações imediatamente, use `--apply-immediately`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --no-apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --no-apply-immediately
```

API do RDS

Para atualizar a versão do mecanismo de um cluster de banco de dados, use a operação ModifyDBCluster. Especifique os seguintes parâmetros:

- `DBClusterIdentifier` – o nome do cluster de banco de dados. Por exemplo *`mydbcluster`*.
- `EngineVersion`: o número da versão do mecanismo de banco de dados para a qual será feita a atualização. Para obter informações sobre versões de mecanismo válidas, use a operação DescribeDBEngineVersions.
- `ApplyImmediately`: se deseja ou não aplicar as alterações imediatamente ou durante a próxima janela de manutenção. Para aplicar as alterações imediatamente, defina o valor como `true`. Para aplicar alterações durante a próxima janela de manutenção, defina o valor como `false`.

Atualizar extensões do PostgreSQL

A atualização do cluster de banco de dados do Aurora PostgreSQL para uma nova versão principal ou secundária não atualiza as extensões do PostgreSQL simultaneamente. Para a maioria das extensões, você atualiza a extensão após a conclusão da atualização da versão principal ou secundária. No entanto, em alguns casos, você atualiza a extensão antes de atualizar o mecanismo de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte list of extensions to update em Testar um upgrade de cluster de banco de dados de produção para uma nova versão principal.

A instalação das extensões do PostgreSQL exigem privilégios de `rds_superuser`. Normalmente, um `rds_superuser` delega permissões por extensões específicas para usuários relevantes (perfis), para facilitar o gerenciamento de uma determinada extensão. Isso significa que a tarefa de atualizar todas as extensões do cluster de banco de dados do Aurora PostgreSQL pode envolver muitos usuários (perfis) diferentes. Tenha isso em mente principalmente se você quiser automatizar o processo de atualização usando scripts. Para ter mais informações sobre privilégios e funções do PostgreSQL, consulte Segurança com o Amazon Aurora PostgreSQL.

> **Note**
>
> Para obter informações sobre como atualizar a extensão PostGIS, consulte Gerenciar dados espaciais com a extensão PostGIS (Etapa 6: Atualize a extensão PostGIS).
> Para atualizar a extensão `pg_repack`, solte a extensão e crie a nova versão na instância de banco de dados atualizada. Para obter mais informações, consulte a instalação do pg_repack na documentação do `pg_repack`.

Para atualizar uma extensão após uma atualização de mecanismo, use o comando `ALTER EXTENSION UPDATE`.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

Para listar as extensões instaladas no momento, use o catálogo pg_extension do PostgreSQL no comando a seguir.

```
SELECT * FROM pg_extension;
```

Para visualizar uma lista das versões de extensão específicas disponíveis para a instalação, use a visualização pg_available_extension_versions do PostgreSQL no comando a seguir.

```
SELECT * FROM pg_available_extension_versions;
```

Técnica alternativa de atualização azul-verde

Em algumas situações, a prioridade é executar um switchover imediato do cluster antigo para um atualizado. Nessas situações, você pode seguir um processo de várias etapas que executa clusters antigos e novos lado a lado. Nesse caso, você replica dados do cluster antigo para o novo até que esteja pronto para que o novo cluster assuma o controle. Para obter mais detalhes, consulte Usar implantações azul/verde para atualizações de banco de dados.

Usar uma versão de suporte de longo prazo (LTS) do Aurora PostgreSQL

Cada nova versão do Aurora PostgreSQL permanece disponível por um período para que você use ao criar ou atualizar um cluster de banco de dados. Após esse período, é necessário atualizar os clusters que usam essa versão. Você pode atualizar manualmente o cluster antes do término do período de suporte, ou o Aurora pode atualizá-lo automaticamente quando a versão do Aurora PostgreSQL não for mais compatível.

O Aurora designa determinadas versões do Aurora PostgreSQL como releases de suporte de longo prazo (LTS). Os clusters de banco de dados que usam releases do LTS podem permanecer na mesma versão por mais tempo e passar por menos ciclos de atualização que os clusters que usam releases que não são do LTS. As versões secundárias do LTS incluem apenas correções de bugs (por meio de versões de patch) para problemas críticos de estabilidade e segurança; uma versão LTS não contém novos recursos lançados após sua apresentação.

Uma vez por ano, aplicam-se patches nos clusters de banco de dados executados em uma versão secundária de LTS para a versão de patch mais recente de release de LTS. Fazemos essa aplicação de patch para ajudar a garantir que você se beneficie de correções cumulativas para obter segurança e estabilidade. Podemos corrigir uma versão secundária de LTS com mais frequência, se houver correções críticas, como para segurança, que necessitem ser aplicadas.

> **Note**
>
> Se quiser permanecer em uma versão secundária de LTS durante o ciclo de vida, desative a atualização automática de versão secundária para as instâncias de banco de dados. Para

não atualizar automaticamente o cluster de banco de dados da versão secundária de LTS, desmarque a caixa de seleção Habilitar a atualização automática da versão secundária em todas as instâncias de banco de dados do cluster do Aurora.

Para a maioria dos clusters do Aurora PostgreSQL, recomendamos atualizar para o release mais recente, em vez de usar o release do LTS. Isso aproveita o Aurora como um serviço gerenciado e fornece acesso aos recursos e correções de erros mais recentes. Os releases do LTS destinam-se a clusters com as seguintes características:

- Não é possível permitir tempo de inatividade na aplicação Aurora PostgreSQL para atualizações fora de ocorrências raras para patches importantes.
- O ciclo de testes para o cluster e para as aplicações associadas exige muito tempo para cada atualização no mecanismo de banco de dados Aurora PostgreSQL.
- A versão do banco de dados para o cluster do Aurora PostgreSQL contém todos os recursos do mecanismo de banco de dados e correções de bug necessários a sua aplicação.

As versões atuais do LTS para Aurora PostgreSQL são as seguintes:

- PostgreSQL 16.8. Foi lançado em 07 de abril de 2025. Para ter mais informações, consulte PostgreSQL 16.8 em Notas de lançamento do Aurora PostgreSQL.
- PostgreSQL 15.10. Foi lançada em 27 de dezembro de 2024. Para ter mais informações, consulte PostgreSQL 15.10 em Release Notes for Aurora PostgreSQL.
- PostgreSQL 14.6. Foi lançado em 20 de janeiro de 2023. Para ter mais informações, consulte PostgreSQL 14.6 em Notas de lançamento do Aurora PostgreSQL.
- PostgreSQL 13.9. Foi lançado em 20 de janeiro de 2023. Para ter mais informações, consulte PostgreSQL 13.9 em Notas de lançamento do Aurora PostgreSQL.
- PostgreSQL 12.9. Foi lançado em 25 de fevereiro de 2022. Para ter mais informações, consulte PostgreSQL 12.9 em Notas de lançamento do Aurora PostgreSQL.
- PostgreSQL 11.9 (Aurora PostgreSQL versão 3.4 Foi lançada em 11 de dezembro de 2020. Para ter mais informações sobre essa versão, consulte PostgreSQL 11.9, Aurora PostgreSQL versão 3.4, em Notas de lançamento do Aurora PostgreSQL.

Consulte detalhes sobre os cronogramas de suporte e os ciclos de lançamento das versões LTS em Release calendars for Aurora PostgreSQL.

Para obter informações sobre como identificar versões do Aurora e do mecanismo de banco de dados, consulte Identificar as versões do Amazon Aurora PostgreSQL.

Uso do Amazon Aurora PostgreSQL Limitless Database

O Amazon Aurora PostgreSQL Limitless Database fornece escalabilidade horizontal automatizada para processar milhões de transações de gravação por segundo e gerenciar petabytes de dados, mantendo a simplicidade de operar em um único banco de dados. Com o Aurora PostgreSQL Limitless Database, você pode se concentrar na criação de aplicações de alta escala sem precisar criar e manter soluções complexas para escalar seus dados em várias instâncias de banco de dados para dar suporte a workloads.

Tópicos

- Arquitetura do Aurora PostgreSQL Limitless Database
- Introdução ao Aurora PostgreSQL Limitless Database
- Requisitos e considerações do Aurora PostgreSQL Limitless Database
- Pré-requisitos para usar o Aurora PostgreSQL Limitless Database
- Criação de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database
- Trabalho com grupos de fragmentos de banco de dados
- Criação de tabelas do Aurora PostgreSQL Limitless Database
- Carregar dados no Aurora PostgreSQL Limitless Database
- Consulta do Aurora PostgreSQL Limitless Database
- Gerenciamento do Aurora PostgreSQL Limitless Database
- Monitoramento do Aurora PostgreSQL Limitless Database
- Monitoramento de desempenho e recursos específicos da instância
- Fazer backup e restaurar o Aurora PostgreSQL Limitless Database
- Atualização do Amazon Aurora PostgreSQL Limitless Database
- Referência do Aurora PostgreSQL Limitless Database

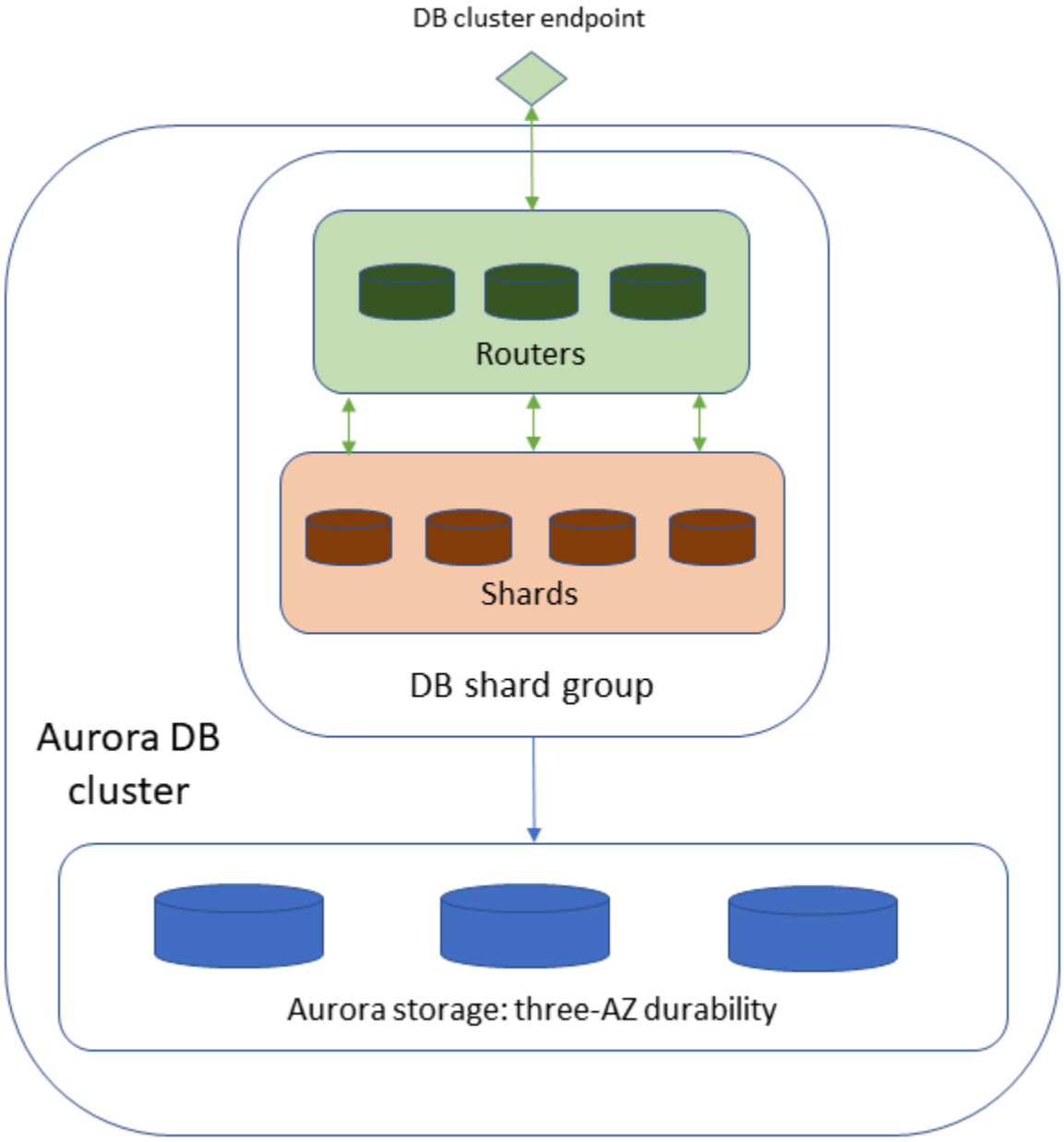
Arquitetura do Aurora PostgreSQL Limitless Database

O Limitless Database alcança escala com uma arquitetura de duas camadas que consiste em vários nós de banco de dados. Os nós são roteadores ou fragmentos.

- Os fragmentos são instâncias de banco de dados do Aurora PostgreSQL que armazenam cada uma um subconjunto dos dados do seu banco de dados, permitindo o processamento simultâneo para obter maior taxa de throughput de gravação.
- Os roteadores gerenciam a natureza distribuída do banco de dados e apresentam uma única imagem do banco de dados aos clientes do banco de dados. Os roteadores mantêm metadados sobre onde os dados são armazenados, analisam os comandos SQL recebidos e enviam esses comandos para os fragmentos. Em seguida, eles agregam dados dos fragmentos para retornar um único resultado ao cliente e gerenciam transações distribuídas para manter a consistência em todo o banco de dados distribuído.

O Aurora PostgreSQL Limitless Database difere de clusters de bancos de dados Aurora padrão por ter um grupo de fragmentos de banco de dados em vez de uma instância de banco de dados do gravador e instâncias de banco de dados do leitor. Todos os nós que compõem a arquitetura do Limitless Database estão contidos no grupo de fragmentos de banco de dados. Os fragmentos e roteadores individuais no grupo de fragmentos de banco de dados não estão visíveis na sua Conta da AWS. Você usa o endpoint do cluster de banco de dados para acessar o Limitless Database.

A figura a seguir mostra a arquitetura de alto nível do Aurora PostgreSQL Limitless Database.



Para obter mais informações sobre a arquitetura do Aurora PostgreSQL Limitless Database e como você pode usá-lo, veja Achieving scale with Amazon Aurora PostgreSQL Limitless Database no canal AWS Events no YouTube.

Para obter mais informações sobre a arquitetura de um cluster de banco de dados padrão do Aurora, consulte Clusters de banco de dados do Amazon Aurora.

Principais termos para o Aurora PostgreSQL Limitless Database

Grupo de fragmentos de banco de dados

Um contêiner para nós do Limitless Database (fragmentos e roteadores).

Roteador

Um nó que aceita conexões SQL de clientes, envia comandos SQL para fragmentos, mantém a consistência em todo o sistema e retorna os resultados aos clientes.

Fragmento

Um nó que armazena um subconjunto de tabelas fragmentadas, cópias completas de tabelas de referência e tabelas padrão. Aceita consultas de roteadores, mas não pode ser conectado diretamente pelos clientes.

Tabela fragmentada

Uma tabela com seus dados particionados em fragmentos.

Chave de fragmento

Uma coluna ou conjunto de colunas em uma tabela fragmentada que é usado para determinar o particionamento entre fragmentos.

Tabelas colocalizadas

Duas tabelas fragmentadas que compartilham a mesma chave de fragmento e são explicitamente declaradas como colocalizadas. Todos os dados do mesmo valor de chave de fragmento são enviados para o mesmo fragmento.

Tabela de referência

Uma tabela com seus dados copiados na íntegra em cada fragmento.

Tabela padrão

O tipo de tabela padrão no Limitless Database. É possível converter tabelas padrão em tabelas fragmentadas e de referência.

Todas as tabelas padrão são armazenadas no mesmo fragmento selecionado pelo sistema, permitindo que as junções entre tabelas padrão sejam realizadas em um único fragmento. No entanto, as tabelas padrão são limitadas pela capacidade máxima do fragmento (128 TiB). Esse fragmento também armazena dados de tabelas fragmentadas e de referência, portanto, o limite efetivo para tabelas padrão é menor que 128 TiB.

Tipos de tabela para o Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database é compatível com três tipos de tabela: fragmentada, de referência e padrão.

As tabelas fragmentadas têm seus dados distribuídos em todos os fragmentos no grupo de fragmentos de banco de dados. O Limitless Database faz isso automaticamente usando uma chave de fragmento, que é uma coluna ou conjunto de colunas que você especifica ao particionar a tabela. Todos os dados com o mesmo valor da chave de fragmento são enviados para o mesmo fragmento. A fragmentação é baseada em hash, não em intervalos ou listas.

Veja a seguir bons casos de uso para tabelas fragmentadas:

- A aplicação funciona com um subconjunto distinto de dados.
- A tabela é muito grande.
- A tabela potencialmente cresce mais rápido do que outras tabelas.

As tabelas fragmentadas podem ser colocalizadas, o que significa que elas compartilham a mesma chave de fragmento e que todos os dados de ambas as tabelas com o mesmo valor de chave de fragmento são enviados para o mesmo fragmento. Se você colocalizar tabelas e uni-las usando a chave de fragmento, a junção poderá ser realizada em um único fragmento porque todos os dados necessários estão presentes nesse fragmento.

As tabelas de referência têm uma cópia completa de todos os dados em cada fragmento no grupo de fragmentos de banco de dados. As tabelas de referência são normalmente usadas para tabelas menores com um volume de gravação menor, mas que ainda precisam ser unidas com frequência e não são adequadas para fragmentação. Exemplos de tabelas de referência incluem tabelas de datas e tabelas de dados geográficos, como estado, cidade e código postal.

As tabelas padrão são o tipo de tabela padrão no Aurora PostgreSQL Limitless Database. Elas não são tabelas distribuídas. O Aurora PostgreSQL Limitless Database é compatível com junções entre tabelas padrão e tabelas padrão, fragmentadas e de referência.

Faturamento do Aurora PostgreSQL Limitless Database

Para obter informações sobre como a cobrança do Aurora PostgreSQL Limitless Database é feita, consulte Faturamento da instância de banco de dados para Aurora.

Para obter informações sobre a definição de preço do Aurora, consulte a página de definição de preço do Aurora.

Introdução ao Aurora PostgreSQL Limitless Database

Realize as seguintes ações para começar a usar o Aurora PostgreSQL Limitless Database:

1. Crie um cluster de banco de dados Aurora PostgreSQL e um grupo de fragmentos de banco de dados para o Limitless Database. Para ter mais informações, consulte Criação de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database.
2. Crie tabelas fragmentadas e de referência no grupo de fragmentos de banco de dados. Para ter mais informações, consulte Criação de tabelas do Aurora PostgreSQL Limitless Database.
3. Altere a capacidade do seu grupo de fragmentos de banco de dados, divida fragmentos e adicione roteadores. Para ter mais informações, consulte Trabalho com grupos de fragmentos de banco de dados.
4. Carregue dados no grupo de fragmentos de banco de dados. Para ter mais informações, consulte Carregar dados no Aurora PostgreSQL Limitless Database.
5. Execute consultas e outras instruções SQL no grupo de fragmentos de banco de dados. Para ter mais informações, consulte Consulta do Aurora PostgreSQL Limitless Database.
6. Monitore o desempenho do Limitless Database. Para ter mais informações, consulte Monitoramento do Aurora PostgreSQL Limitless Database.

Requisitos e considerações do Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database tem os requisitos e considerações a seguir.

Tópicos

- Requisitos do Aurora PostgreSQL Limitless Database
- Considerações do Aurora PostgreSQL Limitless Database
- Recursos não compatíveis com o Aurora PostgreSQL Limitless Database

Requisitos do Aurora PostgreSQL Limitless Database

Siga estes requisitos do Aurora PostgreSQL Limitless Database.

- O Aurora PostgreSQL Limitless Database está disponível em todas as Regiões da AWS, exceto Ásia-Pacífico (Taipei).

  > **Note**
  >
  > Se você criar um cluster de banco de dados do Aurora PostgreSQL Limitless Database no Leste dos EUA (Norte da Virgínia), não inclua a zona de disponibilidade (AZ) `us-east-1e` no grupo de sub-redes de banco de dados. Devido a limitações de recursos, não é possível usar o Aurora Serverless v2 (e, portanto, o Aurora PostgreSQL Limitless Database) na AZ `us-east-1e`.

- O Aurora PostgreSQL Limitless Database é compatível somente com a configuração de armazenamento do cluster de banco de dados Aurora I/O-Optimized. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.
- O Aurora PostgreSQL Limitless Database usa versões especiais do mecanismo de banco de dados do Aurora PostgreSQL para o Aurora PostgreSQL Limitless Database: `16.X-limitless`. Consulte também as versões disponíveis do Aurora PostgreSQL Limitless Database.
- O cluster de banco de dados não pode ter nenhuma instância de banco de dados do gravador ou leitor.
- É necessário usar o Monitoramento aprimorado e o Insights de Performance. O tempo de retenção do Insights de Performance deve ser de pelo menos 1 mês (31 dias).

- É necessário exportar o log do PostgreSQL para o Amazon CloudWatch Logs.

> **Note**
>
> Alguns recursos necessários, como Monitoramento aprimorado, Insights de Performance e CloudWatch Logs, incorrem em custos adicionais. Para obter informações sobre a definição de preço do Aurora, consulte a página de definição de preço do Aurora.

Considerações do Aurora PostgreSQL Limitless Database

As considerações seguintes se aplicam a grupos de fragmentos de banco de dados no Aurora PostgreSQL Limitless Database:

- É possível ter somente um grupo de fragmentos de banco de dados por cluster de banco de dados.
- É possível ter até cinco grupos de fragmentos de banco de dados por Região da AWS.

  Portanto, é possível ter até cinco clusters de banco de dados do Aurora PostgreSQL Limitless Database por Região da AWS. Para obter mais informações, consulte Cotas no Amazon Aurora.
- É possível definir a capacidade máxima de um grupo de fragmentos de banco de dados de 16 a 6.144 ACUs. Para limites de capacidade superiores a 6.144 ACUs, entre em contato com a AWS.

  O número inicial de roteadores e fragmentos é determinado pela capacidade máxima que você define ao criar um grupo de fragmentos de banco de dados. Para obter mais informações, consulte Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados.
- O número de roteadores e fragmentos não muda quando você modifica a capacidade máxima de um grupo de fragmentos de banco de dados.
- A sub-rede de banco de dados em que você cria o grupo de fragmentos de banco de dados deve ter endereços IP livres suficientes para se conectar ao grupo de fragmentos de banco de dados. Você precisa de um endereço IP para cada roteador e até três endereços IP para cada fragmento no grupo de fragmentos de banco de dados.

  Para obter mais informações sobre o número de roteadores criados quando você cria um grupo de fragmentos de banco de dados, consulte Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados.

- Se você tornar o grupo de fragmentos de banco de dados acessível publicamente, configure um gateway da internet na VPC.
- Você usa funções SQL para dividir fragmentos e adicionar roteadores.
- A mesclagem de fragmentos não é compatível.
- Não é possível excluir fragmentos e roteadores individuais.
- Você não pode modificar (realizar operações `UPDATE`) chaves de fragmento de forma alguma, inclusive alterando seus valores nas linhas da tabela.

  Para alterar uma chave de fragmento, exclua-a e recrie-a.
- Os níveis de isolamento leitura repetível, leitura confirmada e leitura não confirmada são compatíveis. Não é possível definir o nível de isolamento como serializável.
- Alguns comandos SQL não são compatíveis. Para obter mais informações, consulte Referência do Aurora PostgreSQL Limitless Database.
- Nem todas as extensões do PostgreSQL são compatíveis. Para obter mais informações, consulte Extensões.
- Ao criar um grupo de fragmentos ou ao adicionar novos nós de grupo de fragmentos (fragmentos ou roteadores), esses nós são criados em uma das zonas de disponibilidade (AZs) disponíveis para o cluster de banco de dados. Não é possível escolher uma AZ específica para nós individuais.
- Se você usar uma redundância computacional de 2 (dois standbys computacionais para o grupo de fragmentos de banco de dados), o grupo de sub-redes de banco de dados deverá ter pelo menos três AZs.
- O Aurora PostgreSQL Limitless Database aceita nomes de tabela fragmentada com até 54 caracteres.

As considerações seguintes se aplicam ao cluster de banco de dados do Aurora PostgreSQL Limitless Database:

- Recomendamos que você use políticas gerenciadas pela AWS para limitar as permissões do banco de dados e aplicações àquelas que os clientes precisam para seus casos de uso. Para obter mais informações, consulte Práticas recomendadas de política.
- Ao criar um cluster de banco de dados do Aurora PostgreSQL Limitless Database, você só define parâmetros de ajuste de escala para o grupo de fragmentos de banco de dados.
- Se você precisar excluir seu cluster de banco de dados, primeiro deve excluir o grupo de fragmentos de banco de dados.

- O Aurora PostgreSQL Limitless Database não pode ser uma fonte de replicação.

Recursos não compatíveis com o Aurora PostgreSQL Limitless Database

Os seguintes recursos do Aurora PostgreSQL não são compatíveis com o Aurora PostgreSQL Limitless Database:

- Autenticação do Active Directory (Kerberos)
- Amazon DevOps Guru
- Amazon ElastiCache
- Implantações azul/verde do Amazon RDS
- Amazon RDS Proxy
- Auto Scaling do Aurora (adição automática de instâncias do leitor ao cluster de banco de dados)
- Banco de dados global do Aurora
- Machine learning do Aurora
- Recomendações do Aurora
- Aurora Serverless v1
- Integrações ETL zero do Aurora
- AWS Backup
- AWS LambdaIntegração do
- AWS Secrets Manager
- Babelfish for Aurora PostgreSQL
- Clonagem de clusters de banco de dados
- Endpoints personalizados
- Fluxos de atividades do banco de dados
- Réplicas de leitura
- API de dados do RDS

Pré-requisitos para usar o Aurora PostgreSQL Limitless Database

Para usar o Aurora PostgreSQL Limitless Database, você deve primeiro realizar as seguintes tarefas.

Tópicos

- Habilitação de operações de grupo de fragmentos de banco de dados

Habilitação de operações de grupo de fragmentos de banco de dados

Antes de criar um grupo de fragmentos de banco de dados, você deve habilitar as operações de grupo de fragmentos de banco de dados.

- Adicione a seguinte seção à política do IAM do perfil do IAM do usuário que acessa o Aurora PostgreSQL Limitless Database:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDBShardGroup",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBShardGroup",
                "rds:DescribeDBShardGroups",
                "rds:DeleteDBShardGroup",
                "rds:ModifyDBShardGroup",
                "rds:RebootDBShardGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:*:shard-group:*",
                "arn:aws:rds:*:*:cluster:*"
            ]
        }
    ]
}
```

Criação de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database

Você cria um novo cluster de banco de dados Aurora usando a versão Limitless Database do Aurora PostgreSQL e adiciona um grupo de fragmentos de banco de dados ao cluster. Ao adicionar um grupo de fragmentos de banco de dados, você especifica a capacidade computacional máxima para todo o grupo de fragmentos de banco de dados (soma da capacidade de todos os roteadores e fragmentos) em unidades de capacidade do Aurora (ACUs). Cada ACU é uma combinação de aproximadamente 2 gigabytes (GB) de memória, CPU correspondente e rede. O ajuste de escala aumenta ou diminui a capacidade do grupo de fragmentos de banco de dados, dependendo da workload da aplicação, de forma semelhante a como o Aurora Serverless v2 funciona.

Tópicos

- Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados
- Criar um cluster de banco de dados

Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados

O número inicial de roteadores e fragmentos em um grupo de fragmentos de banco de dados é determinado pela capacidade máxima que você define ao criar o grupo de fragmentos. Quanto maior a capacidade máxima, maior o número de roteadores e fragmentos criados no grupo de fragmentos de banco de dados.

Cada nó (fragmento ou roteador) tem seu próprio valor de capacidade atual, também medido em ACUs.

- O Limitless Database escala um nó para uma capacidade maior quando a capacidade atual é muito baixa para lidar com a carga. No entanto, os nós não terão a escala aumentada verticalmente quando a capacidade total estiver no máximo.
- O Limitless Database escala o nó para uma capacidade menor quando sua capacidade atual é maior do que a necessária. No entanto, os nós não terão a escala reduzida verticalmente quando a capacidade total estiver no mínimo.

A tabela a seguir mostra a correlação entre a capacidade máxima do grupo de fragmentos de banco de dados em unidades de capacidade do Aurora (ACUs) e o número de nós (roteadores e fragmentos) criados.

> **Note**
>
> Esses valores estão sujeitos a alterações.
> Se você definir a redundância computacional para um valor diferente de zero, o número total de fragmentos será dobrado ou triplicado. Isso incorrerá em custos adicionais.
> Os nós em standbys computacionais têm a escala aumentada e reduzida verticalmente de acordo com a mesma capacidade do gravador. Você não define a faixa de capacidade separadamente para os standbys.

Total de nós	Roteadores	Fragmentos	Capacidade mínima padrão (ACUs)	Intervalo de capacidade máxima (ACUs)
4	2	2	16	16-400
5	2	3	20	401-500
6	2	4	24	501-600
7	3	4	28	601-700
8	3	5	32	701-800
9	3	6	36	801-900
10	4	6	40	901-1.000
11	4	7	44	1.001-1.100
12	4	8	48	1.101-1.200
13	5	8	52	1.201-1.300
14	5	9	56	1.301-1.400

Total de nós	Roteadores	Fragmentos	Capacidade mínima padrão (ACUs)	Intervalo de capacidade máxima (ACUs)
15	5	10	60	1.401-1.500
16	6	10	64	1.501-1.600
17	6	11	68	1.601-1.700
18	6	12	72	1.701-1.800
19	7	12	76	1.801-1.900
20	7	13	80	1.901-2.000
21	7	14	84	2.001-2.100
22	8	14	88	2.101-2.200
23	8	15	92	2.201-2.300
24	8	16	96	2.301-6.144

A configuração dinâmica baseada na capacidade máxima do grupo de fragmentos de banco de dados está disponível somente durante a criação. O número de roteadores e fragmentos permanece o mesmo quando a capacidade máxima é modificada. Para ter mais informações, consulte Alteração da capacidade de um grupo de fragmentos de banco de dados.

Você pode usar comandos SQL para adicionar fragmentos e roteadores a um grupo de fragmentos de banco de dados. Para obter mais informações, consulte:

- Dividir um fragmento em um grupo de fragmentos de banco de dados
- Adicionar um roteador a um grupo de fragmentos de banco de dados

> **Note**
>
> Não é possível excluir fragmentos ou roteadores.

Criar um cluster de banco de dados

Use os procedimentos a seguir para criar um cluster de banco de dados do Aurora PostgreSQL que use o Aurora PostgreSQL Limitless Database.

Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para criar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database. Você cria o cluster de banco de dados primário e o grupo de fragmentos de banco de dados.

Console

Quando você usa o Console de gerenciamento da AWS para criar o cluster de banco de dados primário, o grupo de fragmentos de banco de dados também é criado no mesmo procedimento.

Como criar um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Create database (Criar banco de dados).

   A página Criar banco de dados é exibida.
3. Em Tipo de mecanismo, escolha Aurora (compatível com PostgreSQL).
4. Em Versão, escolha uma das seguintes opções:

   - Aurora PostgreSQL com Limitless Database (compatível com PostgreSQL 16.4)
   - Aurora PostgreSQL com Limitless Database (compatível com PostgreSQL 16.6)
5. Em Aurora PostgreSQL Limitless Database:

a. Insira um Identificador do grupo de fragmentos de banco de dados.

> **Important**
>
> Depois de criar o grupo de fragmentos de banco de dados, não é possível alterar o identificador do cluster de banco de dados ou o identificador do grupo de fragmentos de banco de dados.

b. Em Faixa de capacidade do grupo de fragmentos de banco de dados:

   i. Insira a Capacidade mínima (ACUs). Use um valor de pelo menos 16 ACUs.

      Para um ambiente de desenvolvimento, o valor padrão é 16 ACUs. Para um ambiente de produção, o valor padrão é 24 ACUs.

   ii. Insira a Capacidade máxima (ACUs). Use um valor de pelo menos 16 ACUs ou no máximo 6.144 ACUs.

      Para um ambiente de desenvolvimento, o valor padrão é 64 ACUs. Para um ambiente de produção, o valor padrão é 384 ACUs.

   Para obter mais informações, consulte Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados.

c. Em Implantação de grupos de fragmentos de banco de dados, escolha se deseja criar standbys para o grupo de fragmentos de banco de dados:

- Sem redundância computacional: cria um grupo de fragmentos de banco de dados sem standbys para cada fragmento. Este é o valor padrão.
- Redundância computacional com um único destino de failover: cria um grupo de fragmentos de banco de dados com um standby computacional em uma zona de disponibilidade (AZ) diferente.
- Redundância computacional com dois destinos de failover: cria um grupo de fragmentos de banco de dados com dois standbys computacionais em duas AZs diferentes.

> **Note**
>
> Se você definir a redundância de computação com um valor diferente de zero, o número total de fragmentos será dobrado ou triplicado. Essas instâncias extras de banco de dados são instâncias de computação em espera que têm a escala aumentada e reduzida verticalmente de acordo com a mesma capacidade do gravador. Você não define a faixa de capacidade separadamente para os standbys. Portanto, o uso e o faturamento de ACU dobram e triplicam correspondentemente. Para saber qual é o uso exato de ACU resultante da redundância de computação, consulte a métrica `DBShardGroupComputeRedundancyCapacity` em Métricas do DBShardGroup.

d. Escolha se deseja tornar o grupo de fragmentos de banco de dados acessível publicamente.

> **Note**
>
> Não é possível modificar essa configuração após criar o grupo de fragmentos de banco de dados.

6. Em Conectividade:

a. (Opcional) Selecione Conectar-se a um recurso computacional do EC2 e, em seguida, escolha uma instância existente do EC2 ou crie uma.

> **Note**
>
> Se você se conectar a uma instância do EC2, não poderá tornar o grupo de fragmentos de banco de dados acessível publicamente.

b. Em Tipo de rede, escolha IPv4 ou Modo de pilha dupla.

c. Escolha a Nuvem privada virtual (VPC) e o Grupo de sub-redes de banco de dados, ou use as configurações padrão.

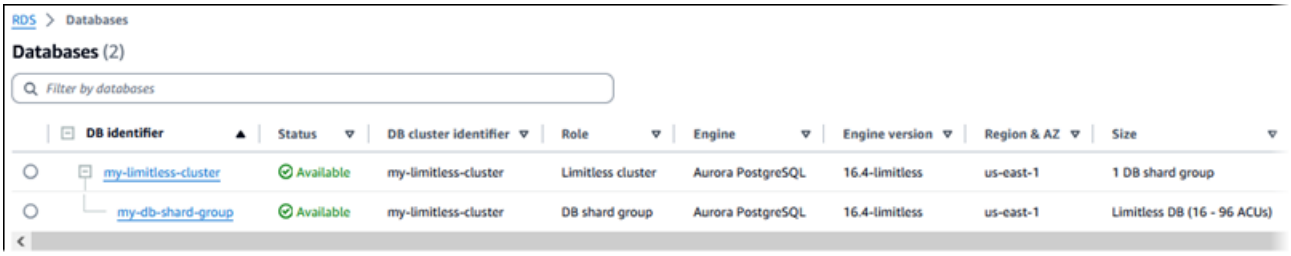
> **Note**
>
> Se você criar um cluster de banco de dados do Limitless Database na região Leste dos EUA (Norte da Virgínia), não inclua a zona de disponibilidade (AZ) `us-east-1e` no grupo de sub-redes de banco de dados. Devido às limitações de recursos, o Aurora Serverless v2 (e, portanto, o Limitless Database) não é compatível na AZ `us-east-1e`.

d. Escolha o Grupo de segurança da VPC (firewall) ou use a configuração padrão.

7. Em Autenticação de banco de dados, escolha Autenticação de senha ou Autenticação de senha e do banco de dados do IAM.
8. Em Monitoramento, certifique-se de que as caixas de seleção Ativar o Performance Insights e Habilitar monitoramento avançado estejam marcadas.

   Para Insights de Performance, escolha um tempo de retenção de pelo menos 1 mês.
9. Expanda Configuração adicional na parte inferior da página.
10. Em Exportações de log, certifique-se de que a caixa de seleção Log do PostgreSQL esteja marcada.
11. Especifique outras configurações conforme necessário. Para obter mais informações, consulte Configurações de clusters de bancos de dados do Aurora.
12. Selecione Criar banco de dados.

Após a criação do cluster de banco de dados primário e do grupo de fragmentos de banco de dados, eles são exibidos na página Bancos de dados.


RDS > Databases
Databases (2)
Filter by databases
DB identifier
Status
DB cluster identifier
Role
Engine
Engine version
Region & AZ
Size
my-limitless-cluster
Available
my-limitless-cluster
Limitless cluster
Aurora PostgreSQL
16.4-limitless
us-east-1
1 DB shard group
my-db-shard-group
Available
my-limitless-cluster
DB shard group
Aurora PostgreSQL
16.4-limitless
us-east-1
Limitless DB (16 - 96 ACUs)

CLI

Ao usar a AWS CLI para criar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database, você executa as seguintes tarefas:

1. Criar o cluster de banco de dados primário.
2. Criar o grupo de fragmentos de banco de dados.

Criar o cluster de banco de dados primário

Os seguintes parâmetros são necessários para criar o cluster de banco de dados:

- `--db-cluster-identifier`: o nome de seu cluster de banco de dados.
- `--engine`: o cluster de banco de dados deve usar o mecanismo de banco de dados `aurora-postgresql`.
- `--engine-version`: o cluster de banco de dados deve usar uma das versões do mecanismo de banco de dados:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: o cluster de banco de dados deve usar a configuração de armazenamento `aurora-iopt1` do cluster de banco de dados.
- `--cluster-scalability-type`: especifica o modo de escalabilidade do cluster de banco de dados do Aurora. Quando definido como `limitless`, o cluster opera como um Aurora PostgreSQL Limitless Database. Quando definido como `standard` (o padrão), o cluster usa a criação normal de instância de banco de dados.

  > **Note**
  >
  > Não é possível modificar essa configuração após a criação do cluster de banco de dados.

- `--master-username`: o nome do usuário principal do cluster de banco de dados.
- `--master-user-password`: a senha do usuário principal.
- `--enable-performance-insights`: é necessário habilitar o Insights de Performance.
- `--performance-insights-retention-period`: o período de retenção do Insights de Performance deve ser de pelo menos 31 dias.

- `--monitoring-interval`: o intervalo, em segundos, entre os pontos quando as métricas do Monitoramento aprimorado são coletadas para o cluster de banco de dados. Esse valor não pode ser `0`.
- `--monitoring-role-arn`: o nome do recurso da Amazon (ARN) do perfil do IAM que permite que o RDS envie métricas do - para o Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: você deve exportar logs `postgresql` para o CloudWatch Logs.

Os seguintes parâmetros são opcionais:

- `--db-subnet-group-name`: o grupo de sub-redes de banco de dados a ser associado ao cluster de banco de dados. Isso também determina a VPC associada ao cluster de banco de dados.

  > **Note**
  >
  > Se você criar um cluster de banco de dados do Limitless Database na região Leste dos EUA (Norte da Virgínia), não inclua a zona de disponibilidade (AZ) `us-east-1e` no grupo de sub-redes de banco de dados. Devido às limitações de recursos, o Aurora Serverless v2 (e, portanto, o Limitless Database) não é compatível na AZ `us-east-1e`.

- `--vpc-security-group-ids`: uma lista de grupos de segurança da VPC para associar ao cluster de banco de dados.
- `--performance-insights-kms-key-id`: o identificador da AWS KMS key para criptografia de dados do Insights de Performance. Se você não especificar uma chave do KMS, a chave padrão da sua Conta da AWS será usada.
- `--region`: a Região da AWS onde você cria o cluster de banco de dados. Ela deve ser compatível com o Aurora PostgreSQL Limitless Database.

Para usar a VPC padrão e o grupo de segurança da VPC, omita as opções `--db-subnet-group-name` e `--vpc-security-group-ids`.

Como criar o cluster de banco de dados primário

- ```
  aws rds create-db-cluster \
      --db-cluster-identifier my-limitless-cluster \
      --engine aurora-postgresql \
      --engine-version 16.6-limitless \
  ```
  ```

```
    --storage-type aurora-iopt1 \
    --cluster-scalability-type limitless \
    --master-username myuser \
    --master-user-password mypassword \
    --db-subnet-group-name mysubnetgroup \
    --vpc-security-group-ids sg-c7e5b0d2 \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
    --enable-cloudwatch-logs-exports postgresql
```

Para obter mais informações, consulte create-db-cluster.

Criar o grupo de fragmentos de banco de dados

Em seguida, você cria o grupo de fragmentos de banco de dados no cluster de banco de dados que acabou de criar. Os seguintes parâmetros são obrigatórios:

- `--db-shard-group-identifier`: o nome do grupo de fragmentos de banco de dados.

  O identificador do grupo de fragmentos de banco de dados tem as seguintes restrições:

  - Ele deve ser exclusivo na Conta da AWS e Região da AWS em que você o criou.
  - Deve conter de 1 a 63 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hífens consecutivos.
  -

    Important

    Depois de criar o grupo de fragmentos de banco de dados, não é possível alterar o identificador do cluster de banco de dados ou o identificador do grupo de fragmentos de banco de dados.

- `--db-cluster-identifier`: o nome do cluster de banco de dados no qual você está criando o grupo de fragmentos de banco de dados.
- `--max-acu`: a capacidade máxima do grupo de fragmentos de banco de dados. Deve ser de 16 a 6.144 ACUs. Para limites de capacidade superiores a 6.144 ACUs, entre em contato com a AWS.

  O número inicial de roteadores e fragmentos é determinado pela capacidade máxima que você define ao criar o grupo de fragmentos de banco de dados. Quanto maior a capacidade máxima, maior o número de roteadores e fragmentos criados no grupo de fragmentos de banco de dados. Para obter mais informações, consulte Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados.

Os seguintes parâmetros são opcionais:

- `--compute-redundancy`: se deve criar standbys para o grupo de fragmentos de banco de dados. Esse parâmetro pode ter um dos seguintes valores:
  - `0`: cria um grupo de fragmentos de banco de dados sem standbys para cada fragmento. Este é o valor padrão.

- 1: cria um grupo de fragmentos de banco de dados com um standby computacional em uma zona de disponibilidade (AZ) diferente.
- 2: cria um grupo de fragmentos de banco de dados com dois standbys computacionais em duas AZs diferentes.

> **Note**
>
> Se você definir a redundância computacional para um valor diferente de zero, o número total de fragmentos será dobrado ou triplicado. Isso incorrerá em custos adicionais.
> Os nós em standbys computacionais têm a escala aumentada e reduzida verticalmente de acordo com a mesma capacidade do gravador. Você não define a faixa de capacidade separadamente para os standbys.

- `--min-acu`: a capacidade mínima do grupo de fragmentos de banco de dados. Deve ser de pelo menos 16 ACUs, que é o valor padrão.
- `--publicly-accessible|--no-publicly-accessible`: se devem ser atribuídos endereços IP acessíveis publicamente ao grupo de fragmentos de banco de dados. O acesso ao grupo de fragmentos de banco de dados é controlado pelos grupos de segurança usados pelo cluster.

  O padrão é `--no-publicly-accessible`.

> **Note**
>
> Não é possível modificar essa configuração após criar o grupo de fragmentos de banco de dados.

Como criar o grupo de fragmentos de banco de dados

- ```
  aws rds create-db-shard-group \
      --db-shard-group-identifier my-db-shard-group \
      --db-cluster-identifier my-limitless-cluster \
      --max-acu 1000
  ```
  ```

Trabalho com grupos de fragmentos de banco de dados

Realize as seguintes tarefas para adicionar e gerenciar um grupo de fragmentos de banco de dados para o Aurora PostgreSQL Limitless Database.

Tópicos

- Conexão ao cluster de banco de dados do Aurora PostgreSQL Limitless Database
- Localização do número de roteadores e fragmentos em um grupo de fragmentos de banco de dados
- Descrição de grupos de fragmentos de banco de dados
- Reinicialização de um grupo de fragmentos de banco de dados
- Alteração da capacidade de um grupo de fragmentos de banco de dados
- Dividir um fragmento em um grupo de fragmentos de banco de dados
- Adicionar um roteador a um grupo de fragmentos de banco de dados
- Exclusão de um grupo de fragmentos de banco de dados
- Adição de um grupo de fragmentos de banco de dados a um cluster de banco de dados do Aurora PostgreSQL Limitless Database existente

Conexão ao cluster de banco de dados do Aurora PostgreSQL Limitless Database

Para trabalhar com o Aurora PostgreSQL Limitless Database, você se conecta ao endpoint e gravador ou leitor do cluster. Você pode usar `psql` ou qualquer outro utilitário de conexão que funcione com o PostgreSQL:

```
$ psql -h DB_cluster_endpoint -p port_number -U database_username -d postgres_limitless
```

O exemplo a seguir usa o endpoint para o cluster de banco de dados que você criou em CLI.

```
$ psql -h my-limitless-cluster.cluster-ckifpdyyyxxx.us-east-1.rds.amazonaws.com -p 5432
 -U postgres -d postgres_limitless
```

> **Note**
>
> O banco de dados padrão para o grupo de fragmentos de banco de dados no Aurora PostgreSQL Limitless Database é `postgres_limitless`.

Uso do Limitless Connection Plugin

Ao se conectar ao Aurora PostgreSQL Limitless Database, os clientes se conectam usando o endpoint do cluster e são roteados para um roteador de transações pelo Amazon Route 53. No entanto, o Route 53 é limitado na capacidade de balancear a carga e pode permitir workloads desiguais nos roteadores de transações. O Limitless Connection Plugin no Driver JDBC da AWS resolve isso realizando o balanceamento de carga do lado do cliente com reconhecimento de carga. Para obter mais informações sobre o Driver JDBC da AWS, consulte Conectar-se ao Aurora PostgreSQL com o driver JDBC da Amazon Web Services (AWS).

Localização do número de roteadores e fragmentos em um grupo de fragmentos de banco de dados

Você pode usar a consulta a seguir para encontrar o número de roteadores e fragmentos:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Descrição de grupos de fragmentos de banco de dados

Use o comando `describe-db-shard-groups` da AWS CLI para descrever seus grupos de fragmentos de banco de dados. O seguinte parâmetro é opcional:

- `--db-shard-group-identifier`: o nome de um grupo de fragmentos de banco de dados.

O exemplo a seguir descreve um grupo de fragmentos de banco de dados específico.

```
aws rds describe-db-shard-groups --db-shard-group-identifier my-db-shard-group
```

A saída será semelhante ao seguinte exemplo.

```
{
    "DBShardGroups": [
        {
            "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
            "DBShardGroupIdentifier": "my-shard-group",
            "DBClusterIdentifier": "my-limitless-cluster",
            "MaxACU": 1000.0,
            "ComputeRedundancy": 0,
            "Status": "available",
            "PubliclyAccessible": false,
```

```
            "Endpoint": "my-limitless-cluster.limitless-ccetp2abcdef.us-
east-1.rds.amazonaws.com"
        }
    ]
}
```

Reinicialização de um grupo de fragmentos de banco de dados

Às vezes, você precisa reinicializar o grupo de fragmentos de banco de dados, por exemplo, quando o parâmetro `max_connections` muda devido a uma alteração na capacidade máxima.

Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para alterar a capacidade de um grupo de fragmentos de banco de dados.

Console

Use o procedimento a seguir.

Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

1. Navegue até a página Databases (Bancos de dados).
2. Selecione o grupo de fragmentos de banco de dados que você deseja reiniciar.
3. Em Actions (Ações), escolha Reboot (Reiniciar).
4. Escolha Confirmar.

CLI

Para reiniciar um grupo de fragmentos de banco de dados, use o comando AWS CLI da `reboot-db-shard-group` com o parâmetro a seguir:

- `--db-shard-group-identifier`: o nome de um grupo de fragmentos de banco de dados.

O exemplo a seguir reinicializa um grupo de fragmentos de banco de dados.

```
aws rds reboot-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Alteração da capacidade de um grupo de fragmentos de banco de dados

Você pode usar o Console de gerenciamento da AWS ou a AWS CLI para alterar a capacidade de um grupo de fragmentos de banco de dados.

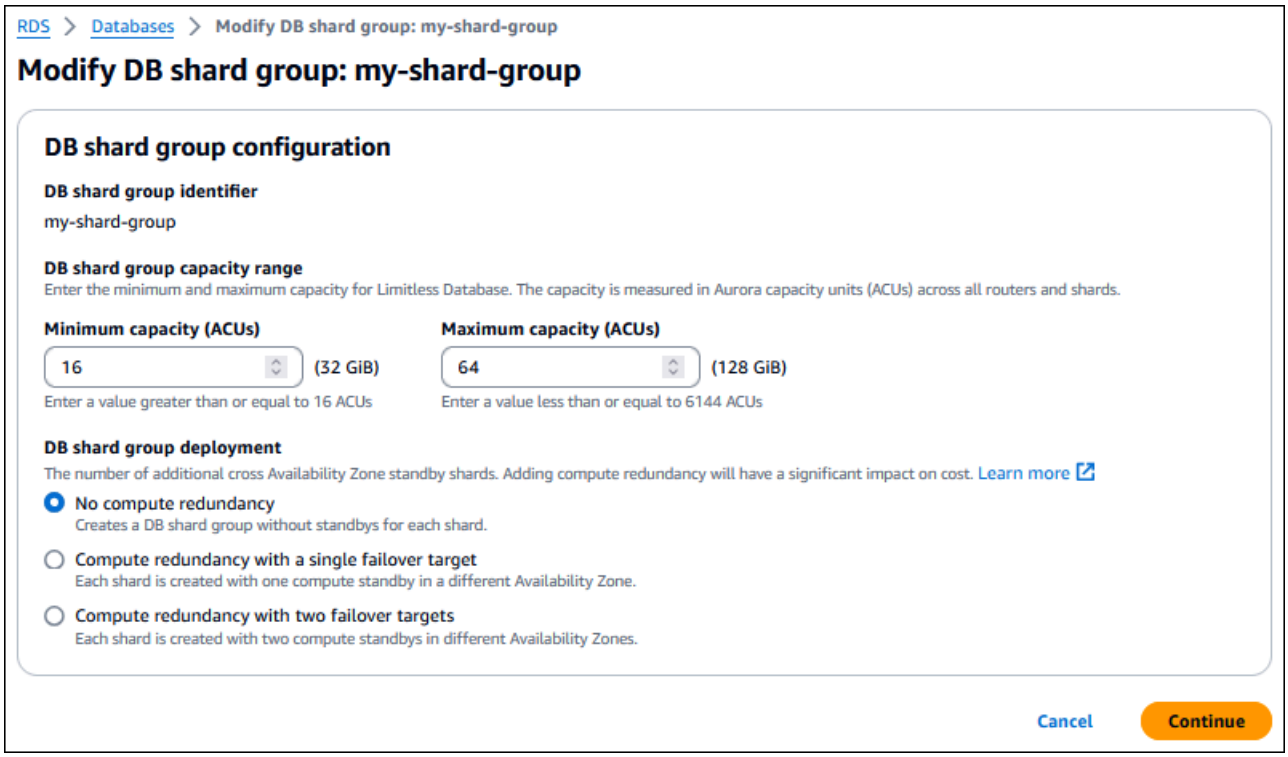
Console

Use o procedimento a seguir.

Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

1. Navegue até a página Databases (Bancos de dados).
2. Selecione o grupo de fragmentos de banco de dados que você deseja modificar.
3. Para Actions (Ações), escolha Modify (Modificar).

   A página Modificar grupo de fragmentos de banco de dados é exibida.



4. Insira um novo valor de Capacidade mínima (ACUs), por exemplo, **`100`**.
5. Insira um novo valor de Capacidade máxima (ACUs), por exemplo, **`1000`**.
6. Escolha Continuar.

   A página de confirmação é exibida, com um resumo das alterações.

7. Revise as alterações e selecione Modificar grupo de fragmentos de banco de dados.

CLI

Use o comando `modify-db-shard-group` da AWS CLI com os seguintes parâmetros:

- `--db-shard-group-identifier`: o nome do grupo de fragmentos de banco de dados.
- `--max-acu`: a nova capacidade máxima do grupo de fragmentos de banco de dados. É possível definir a capacidade máxima do grupo de fragmentos de banco de dados de 16 a 6.144 ACUs. Para limites de capacidade superiores a 6.144 ACUs, entre em contato com a AWS.

  O número de roteadores e fragmentos não muda.
- `--min-acu`: a nova capacidade mínima do grupo de fragmentos de banco de dados. Deve ser de pelo menos 16 ACUs, que é o valor padrão.

O exemplo de CLI a seguir altera a faixa de capacidade de um grupo de fragmentos de banco de dados para 100 a 1.000 ACUs.

```
aws rds modify-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --min-acu 100 \
    --max-acu 1000
```

Dividir um fragmento em um grupo de fragmentos de banco de dados

Você pode dividir manualmente um fragmento em um grupo de fragmentos de banco de dados em dois fragmentos menores. Isso é chamado de divisão de fragmentos iniciada pelo usuário.

O Aurora PostgreSQL Limitless Database também pode dividir fragmentos quando eles têm grandes quantidades de dados ou um uso muito alto. Isso é chamado de divisão de fragmentos iniciada pelo sistema.

Tópicos

- Pré-requisitos
- Dividir um fragmento
- Rastrear divisões de fragmentos
- Finalizar divisões de fragmentos
- Cancelar uma divisão de fragmentos

Pré-requisitos

As divisões de fragmento iniciadas pelo usuário têm os seguintes pré-requisitos:

- É necessário ter um grupo de fragmentos de banco de dados.
- O grupo de fragmentos de banco de dados não pode estar vazio: ele deve conter pelo menos uma tabela fragmentada.
- Um usuário deve ter o privilégio `rds_aurora_limitless_cluster_admin`. O `rds_superuser` tem esse privilégio; portanto, o usuário principal também o tem. O `rds_superuser` pode conceder o privilégio a outros usuários:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

- É necessário saber o ID do subcluster (nó) do fragmento que deseja dividir. É possível obter o ID usando a seguinte consulta:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
```

```
1               | router
2               | router
3               | shard
4               | shard
5               | shard
6               | shard
```

Para habilitar divisões de fragmento iniciadas pelo sistema, defina os seguintes parâmetros de cluster de banco de dados em um grupo de parâmetros de cluster de banco de dados personalizado associado ao seu cluster de banco de dados:

Parameter	Valor
`rds_aurora.limitless_enable_auto_scale`	on
`rds_aurora.limitless_auto_scale_options`	`split_shard` ou `add_router,split_shard`
`rds_aurora.limitless_finalize_split_shard_mode`	Este parâmetro determina como as divisões de fragmento iniciadas pelo sistema são finalizadas. O valor pode ser um dos seguintes: • `user_initiated` : você decide quando finalizar a divisão do fragmento. Este é o valor padrão. • `immediate` : as divisões de fragmentos são finalizadas imediatamente. Para obter mais informações, consulte Finalizar divisões de fragmentos. **Note** Este parâmetro se aplica apenas a divisões de fragmento iniciadas pelo sistema.

Para obter mais informações, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

Dividir um fragmento

Para dividir um fragmento em um grupo de fragmentos de banco de dados, use a função `rds_aurora.limitless_split_shard`. Essa função inicia um trabalho de divisão de fragmentos que é executado de forma assíncrona.

```
SELECT rds_aurora.limitless_split_shard('subcluster_id');
```

Aguarde o retorno de um ID de trabalho após o envio bem-sucedido do trabalho, por exemplo:

```
SELECT rds_aurora.limitless_split_shard('3');

    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Operações simultâneas de divisão de fragmentos não são aceitas. Execute todas as operações sequencialmente e conclua cada uma antes de iniciar outra operação de adição.

Rastrear divisões de fragmentos

É possível usar o ID do trabalho para rastrear um trabalho de divisão de fragmentos. Para descrever um trabalho específico e obter mais detalhes sobre ele, execute a seguinte consulta:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(job_id);
```

Por exemplo:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(1691300000000);

    job_id     |    action    |      job_details      | status  |    submission_time
 |                 message
```

```
--------------+------------+----------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | SPLIT_SHARD | Split Shard 3 by User | SUCCESS | 2023-08-06 05:33:20+00
 | Scaling job succeeded.                +
               |            |                       |         |
 | New shard instance with ID 7 was created.
(1 row)
```

A consulta retorna um erro quando você passa um trabalho inexistente como entrada.

```
SELECT * from rds_aurora.limitless_list_shard_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

É possível acompanhar o status de todos os trabalhos de divisão de fragmentos usando a mesma consulta sem um ID de trabalho, por exemplo:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs();

    job_id     |   action    |  job_details          |   status    |    submission_time
     |                 message
--------------+------------+----------------------+-------------
+------------------------
+-------------------------------------------------------------
 1691200000000 | SPLIT_SHARD | Split Shard 3 by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | SPLIT_SHARD | Split Shard 4 by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded. +
               |            |                       |             |
     | New shard instance with ID 7 was created.
 1691400000000 | SPLIT_SHARD | Split Shard 5 by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add shard job 1691400000000.
               |            |                       |             |
     | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | SPLIT_SHARD | Split Shard 5 by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

O status de trabalho pode ser um dos seguintes:

- `IN_PROGRESS`: o trabalho de divisão de fragmentos foi enviado e está em andamento. É possível ter apenas um trabalho em andamento por vez.

- `PENDING`: o trabalho de divisão de fragmentos está aguardando sua finalização. Para obter mais informações, consulte Finalizar divisões de fragmentos.
- `CANCELLATION_IN_PROGRESS`: o trabalho de divisão de fragmentos está sendo cancelado pelo usuário.
- `CANCELED`: o trabalho de divisão de fragmentos foi cancelado com sucesso pelo usuário ou pelo sistema.
- `SUCCESS`: o trabalho de divisão de fragmentos foi concluído com sucesso. O campo `message` contém o ID da instância do novo fragmento.
- `FAILED`: o trabalho de divisão de fragmentos falhou. O campo `message` contém os detalhes da falha e todas as ações que podem ser tomadas como acompanhamento do trabalho que falhou.

Finalizar divisões de fragmentos

A finalização é a última etapa do processo de divisão de fragmentos. Isso causa algum tempo de inatividade. Se você iniciar um trabalho de divisão de fragmentos, a finalização ocorrerá imediatamente após a conclusão bem-sucedida do trabalho.

Às vezes, o sistema divide fragmentos com base na workload, quando você habilita as divisões de fragmentos iniciadas pelo sistema usando o parâmetro `rds_aurora.limitless_enable_auto_scale`.

Nesse caso, é possível escolher se a finalização acontece imediatamente ou em um horário que você escolher. Você usa o parâmetro de cluster de banco de dados `rds_aurora.limitless_finalize_split_shard_mode` para escolher quando isso acontece:

- Se você definir o valor como `immediate`, isso acontecerá imediatamente.
- Se você definir o valor como `user_initiated`, você precisará finalizar o trabalho de divisão de fragmentos manualmente.

  Um evento do RDS é enviado para você e o status do trabalho de divisão de fragmentos é definido como `PENDING`.

Quando definido como `user_initiated`, você usa a função `rds_aurora.limitless_finalize_split_shard` para finalizar o trabalho de divisão de fragmentos:

```
SELECT * FROM rds_aurora.limitless_finalize_split_shard(job_id);
```

> **Note**
>
> Essa função se aplica somente às divisões de fragmentos iniciadas pelo sistema, não por você.

Cancelar uma divisão de fragmentos

Você pode cancelar uma divisão de fragmentos iniciada pelo usuário ou pelo sistema que tenha o status `IN_PROGRESS` ou `PENDING`. Você precisa do ID do trabalho para cancelá-lo.

```
SELECT * from rds_aurora.limitless_cancel_shard_scale_jobs(job_id);
```

Nenhuma saída é retornada, a menos que haja um erro. É possível acompanhar o cancelamento usando uma consulta de rastreamento de trabalho.

Adicionar um roteador a um grupo de fragmentos de banco de dados

Você pode adicionar um roteador a um grupo de fragmentos de banco de dados.

Tópicos

- Pré-requisitos
- Adicionar um roteador
- Rastrear adições de roteador
- Cancelar a adição de um roteador

Pré-requisitos

A adição de um roteador tem os seguintes pré-requisitos:

- É necessário ter um grupo de fragmentos de banco de dados.
- Um usuário deve ter o privilégio `rds_aurora_limitless_cluster_admin`. O `rds_superuser` tem esse privilégio; portanto, o usuário principal também o tem. O `rds_superuser` pode conceder o privilégio a outros usuários:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

> **Note**
>
> Se você alterar o certificado CA padrão da sua Conta da AWS após a criação do grupo de fragmentos de banco de dados, o novo roteador usará o novo certificado CA, que é diferente do certificado CA do roteador existente. Dependendo do seu armazenamento confiável, algumas conexões podem falhar.

- Para habilitar a adição de roteador iniciada pelo sistema, defina os seguintes parâmetros de cluster de banco de dados em um grupo de parâmetros de cluster de banco de dados personalizado associado ao seu cluster de banco de dados:

Parameter	Valor
`rds_aurora.limitless_enable_auto_scale`	`on`
`rds_aurora.limitless_auto_scale_options`	`add_router` ou `add_router,split_shard`

Para obter mais informações, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

Adicionar um roteador

Para adicionar um roteador, use a função `rds_aurora.limitless_add_router`. Essa função inicia um trabalho de adição de roteador que é executado de forma assíncrona.

```
SELECT rds_aurora.limitless_add_router();
```

Aguarde o retorno de um ID de trabalho após o envio bem-sucedido do trabalho, por exemplo:

```
    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Operações simultâneas de adição de roteador não são aceitas. Execute as operações sequencialmente e conclua cada uma antes de iniciar outra operação de adição.

Rastrear adições de roteador

É possível usar o ID do trabalho para rastrear um trabalho de adição de roteador. Para descrever um trabalho específico e obter mais detalhes sobre ele, execute a seguinte consulta:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(job_id);
```

Por exemplo:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(1691300000000);

    job_id     |   action   |        job_details       | status  |    submission_time
   |                  message
---------------+------------+--------------------------+---------
+------------------------+--------------------------------------------
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                  +
               |            |                          |         |
   | New router instance with ID 7 was created.
(1 row)
```

A consulta retorna um erro quando você passa um trabalho inexistente como entrada.

```
SELECT * from rds_aurora.limitless_list_router_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

É possível rastrear o status de todos os trabalhos de adição de roteador usando a mesma consulta sem um ID de trabalho, por exemplo:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs();

    job_id     |   action   |        job_details       |   status    |
 submission_time     |                  message
---------------+------------+--------------------------+-------------
+------------------------+--------------------------------------------
 1691200000000 | ADD_ROUTER | Add 1 new Router by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                 +
               |            |                          |             |
       | New router instance with ID 7 was created.
 1691400000000 | ADD_ROUTER | Add 1 new Router by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add router job 1691400000000.
               |            |                          |             |
       | Retry the command. If the issue persists, contact AWS Support.
```

```
 1691500000000 | ADD_ROUTER | Add 1 new Router by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

O status de trabalho pode ser um dos seguintes:

- `IN_PROGRESS`: o trabalho de adição do roteador foi enviado e está em andamento. É possível ter apenas um trabalho em andamento por vez.
- `CANCELLATION_IN_PROGRESS`: o trabalho de adição do roteador está sendo cancelado pelo usuário.
- `CANCELED`: o trabalho de adição do roteador foi cancelado com sucesso pelo usuário ou pelo sistema.
- `SUCCESS`: o trabalho de adição do roteador foi concluído com sucesso. O campo `message` contém o ID da instância do novo roteador.
- `FAILED`: o trabalho de adição do roteador falhou. O campo `message` contém os detalhes da falha e todas as ações que podem ser tomadas como acompanhamento do trabalho que falhou.

> **Note**
>
> Não há status `PENDING` porque as adições de roteador não precisam ser finalizadas. Elas não incorrem em tempo de inatividade.

Cancelar a adição de um roteador

É possível cancelar a adição de um roteador com o status `IN_PROGRESS`. Você precisa do ID do trabalho para cancelá-lo.

```
SELECT * from rds_aurora.limitless_cancel_router_scale_jobs(job_id);
```

Nenhuma saída é retornada, a menos que haja um erro. É possível acompanhar o cancelamento usando uma consulta de rastreamento de trabalho.

Exclusão de um grupo de fragmentos de banco de dados

Você pode excluir um grupo de fragmentos de banco de dados, se necessário. A exclusão do grupo de fragmentos de banco de dados exclui os nós de computação (fragmentos e roteadores), mas não o armazenamento.

Console

Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

1. Navegue até a página Databases (Bancos de dados).
2. Selecione o grupo de fragmentos de banco de dados que você deseja excluir.
3. Em Ações, escolha Excluir.
4. Insira **`delete me`** na caixa e escolha Excluir.

O grupo de fragmentos de banco de dados é excluído.

AWS CLI

Para excluir um grupo de fragmentos de banco de dados, use o comando AWS CLI da `delete-db-shard-group` com o parâmetro a seguir:

- `--db-shard-group-identifier`: o nome do grupo de fragmentos de banco de dados.

O exemplo a seguir exclui um grupo de fragmentos de banco de dados no cluster de banco de dados do Aurora PostgreSQL criado anteriormente.

```
aws rds delete-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Adição de um grupo de fragmentos de banco de dados a um cluster de banco de dados do Aurora PostgreSQL Limitless Database existente

Você pode criar um grupo de fragmentos de banco de dados em um cluster de banco de dados existente, por exemplo, se estiver restaurando um cluster de banco de dados ou se tiver excluído o grupo de fragmentos de banco de dados.

Para obter mais informações sobre os requisitos do cluster de banco de dados primário e do grupo de fragmentos de banco de dados, consulte Requisitos e considerações do Aurora PostgreSQL Limitless Database.

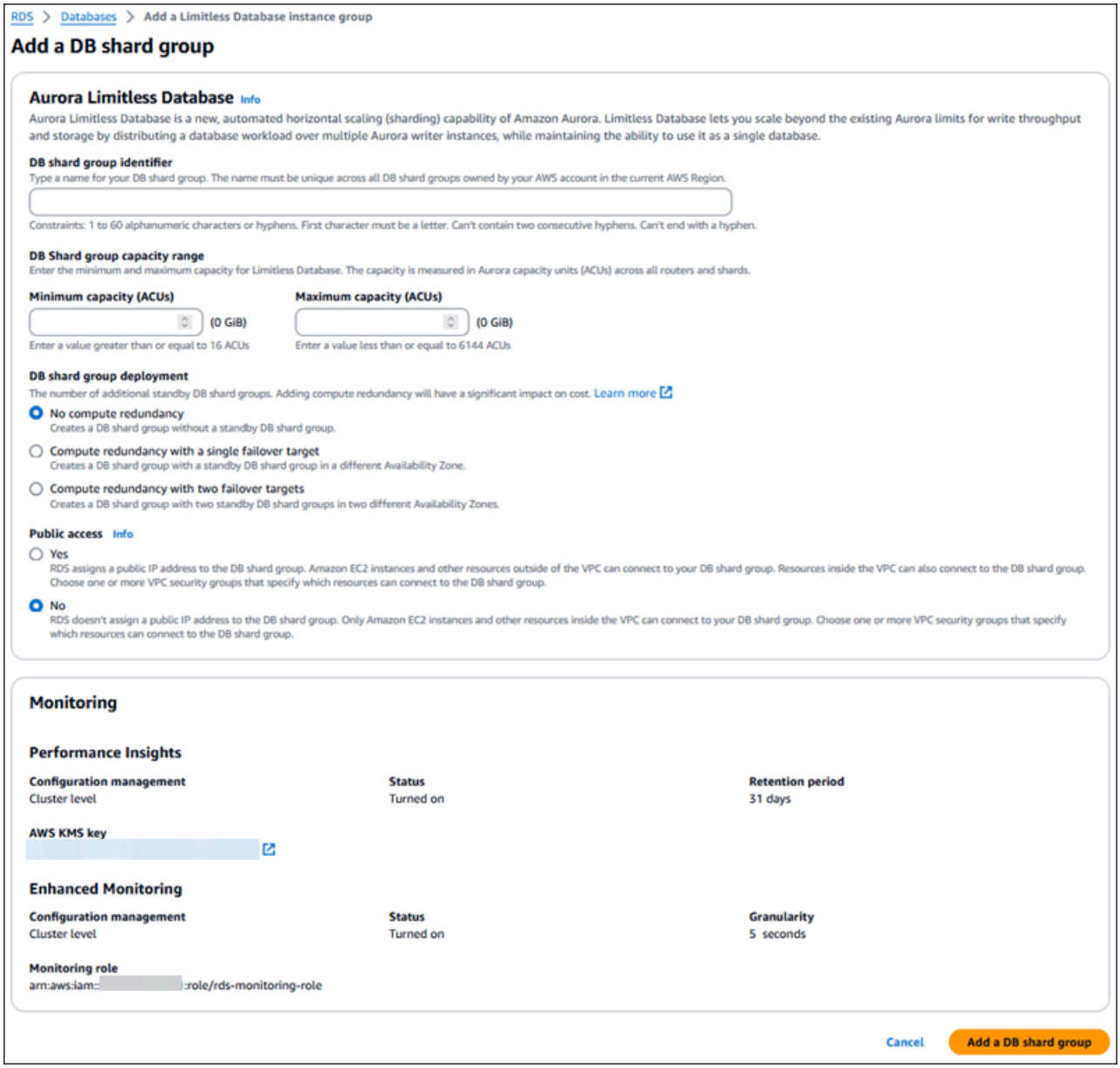
> **Note**
>
> É possível ter somente um grupo de fragmentos de banco de dados por cluster.
> O cluster de banco de dados do Limitless Database deve estar no estado `available` antes que você possa criar um grupo de fragmentos de banco de dados.

Console

Você pode usar o Console de gerenciamento da AWS para adicionar um grupo de fragmentos de banco de dados a um cluster de banco de dados existente.

Como adicionar um grupo de fragmentos de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Navegue até a página Databases (Bancos de dados).
3. Selecione o cluster de banco de dados do Limitless Database ao qual você deseja adicionar um grupo de fragmentos de banco de dados.
4. Em Ações, escolha Adicionar um grupo de fragmentos de banco de dados.



5. Insira um Identificador do grupo de fragmentos de banco de dados.

> **Important**
>
> Depois de criar o grupo de fragmentos de banco de dados, não é possível alterar o identificador do cluster de banco de dados ou o identificador do grupo de fragmentos de banco de dados.

6. Insira a Capacidade mínima (ACUs). Use um valor de pelo menos 16 ACUs.
7. Insira a Capacidade máxima (ACUs). Use um valor de 16 a 6144 ACUs.

Para ter mais informações, consulte Correlação da capacidade máxima do grupo de fragmentos de banco de dados com o número de roteadores e fragmentos criados.

8. Em Implantação de grupos de fragmentos de banco de dados, escolha se deseja criar standbys para o grupo de fragmentos de banco de dados:

   - Sem redundância computacional: cria um grupo de fragmentos de banco de dados sem standbys para cada fragmento. Este é o valor padrão.
   - Redundância computacional com um único destino de failover: cria um grupo de fragmentos de banco de dados com um standby computacional em uma zona de disponibilidade (AZ) diferente.
   - Redundância computacional com dois destinos de failover: cria um grupo de fragmentos de banco de dados com dois standbys computacionais em duas AZs diferentes.

9. Escolha se deseja tornar o grupo de fragmentos de banco de dados acessível publicamente.

   > **Note**
   >
   > Não é possível modificar essa configuração após criar o grupo de fragmentos de banco de dados.

10. Escolha Adicionar um grupo de fragmentos de banco de dados.

AWS CLI

Use o comando `create-db-shard-group` da AWS CLI para criar um grupo de fragmentos de banco de dados.

Os seguintes parâmetros são obrigatórios:

- `--db-cluster-identifier`: o cluster de banco de dados ao qual o grupo de fragmentos de banco de dados pertence.
- `--db-shard-group-identifier`: o nome do grupo de fragmentos de banco de dados.

  O identificador do grupo de fragmentos de banco de dados tem as seguintes restrições:

  - Ele deve ser exclusivo na Conta da AWS e Região da AWS em que você o criou.
  - Deve conter de 1 a 63 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hífens consecutivos.

  > **Important**
  >
  > Depois de criar o grupo de fragmentos de banco de dados, não é possível alterar o identificador do cluster de banco de dados ou o identificador do grupo de fragmentos de banco de dados.

- `--max-acu`: a capacidade máxima do grupo de fragmentos de banco de dados. Use um valor de 16 a 6144 ACUs.

Os seguintes parâmetros são opcionais:

- `--compute-redundancy`: se deve criar standbys para o grupo de fragmentos de banco de dados. Esse parâmetro pode ter um dos seguintes valores:
  - `0`: cria um grupo de fragmentos de banco de dados sem standbys para cada fragmento. Este é o valor padrão.
  - `1`: cria um grupo de fragmentos de banco de dados com um standby computacional em uma zona de disponibilidade (AZ) diferente.
  - `2`: cria um grupo de fragmentos de banco de dados com dois standbys computacionais em duas AZs diferentes.

> **Note**
>
> Se você definir a redundância computacional para um valor diferente de zero, o número total de nós será dobrado ou triplicado. Isso incorrerá em custos adicionais.

- `--min-acu`: a capacidade mínima do grupo de fragmentos de banco de dados. Deve ser de pelo menos 16 ACUs, que é o valor padrão.
- `--publicly-accessible|--no-publicly-accessible`: se devem ser atribuídos endereços IP acessíveis publicamente ao grupo de fragmentos de banco de dados. O acesso ao grupo de fragmentos de banco de dados é controlado pelos grupos de segurança usados pelo cluster.

  O padrão é `--no-publicly-accessible`.

> **Note**
>
> Não é possível modificar essa configuração após criar o grupo de fragmentos de banco de dados.

O exemplo a seguir cria um grupo de fragmentos de banco de dados no cluster de banco de dados do Aurora PostgreSQL.

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-db-cluster \
    --db-shard-group-identifier my-new-shard-group \
    --max-acu 1000
```

A saída será semelhante ao seguinte exemplo.

```
{
    "Status": "CREATING",
    "Endpoint": "my-db-cluster.limitless-ckifpdyyyxxx.us-east-1.rds.amazonaws.com",
    "PubliclyAccessible": false,
    "DBClusterIdentifier": "my-db-cluster",
    "MaxACU": 1000.0,
    "DBShardGroupIdentifier": "my-new-shard-group",
    "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
    "ComputeRedundancy": 0
```

```
}
```

Criação de tabelas do Aurora PostgreSQL Limitless Database

Há três tipos de tabelas que contêm seus dados no Aurora PostgreSQL Limitless Database:

- Padrão: é o tipo de tabela padrão no Aurora PostgreSQL Limitless Database. Você cria tabelas padrão usando o comando CREATE TABLE e pode executar operações de Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML) nelas.

  As tabelas padrão não são tabelas distribuídas. Elas são armazenadas em um dos fragmentos escolhidos internamente pelo sistema.
- Fragmentadas: essas tabelas são distribuídas em vários fragmentos. Os dados são divididos entre os fragmentos com base nos valores das colunas designadas na tabela. Esse conjunto de colunas é chamado de chave de fragmento.
- Referência: essas tabelas são replicadas em todos os fragmentos. Elas são usadas para dados de referência pouco modificados, como catálogos de produtos e códigos postais.

  As consultas de junção entre tabelas de referência e tabelas fragmentadas podem ser executadas em fragmentos, eliminando a movimentação desnecessária de dados entre fragmentos e roteadores.

Existem duas maneiras de criar tabelas ilimitadas:

- Criação de tabelas ilimitadas usando variáveis: use este método quando quiser criar novas tabelas fragmentadas e de referência.
- Converter tabelas padrão em tabelas ilimitadas: use este método quando quiser converter tabelas padrão existentes em tabelas fragmentadas e de referência.

Também fornecemos esquemas de exemplo para o Aurora PostgreSQL Limitless Database.

Criação de tabelas ilimitadas usando variáveis

Você pode usar variáveis para criar tabelas fragmentadas e de referência definindo o modo de criação de tabela. As tabelas criadas usarão esse modo até que você defina um modo diferente.

Use as seguintes variáveis para criar tabelas fragmentadas e de referência:

- `rds_aurora.limitless_create_table_mode`: defina essa variável de sessão como `sharded` ou `reference`. O valor padrão da variável é `standard`.
- `rds_aurora.limitless_create_table_shard_key`: defina essa variável de sessão como uma matriz de nomes de colunas para usar como chaves de fragmento. Essa variável é ignorada quando `rds_aurora.limitless_create_table_mode` não é `sharded`.

  Formate o valor como `untyped array literal`, semelhante a quando você insere literais em uma coluna de matriz. Para obter mais informações, consulte Matrizes na documentação do PostgreSQL.
- `rds_aurora.limitless_create_table_collocate_with`: defina essa variável de sessão com um nome de tabela específico para colocar tabelas recém-criadas com essa tabela.

  Se duas ou mais tabelas forem fragmentadas usando a mesma chave de fragmento, você poderá alinhar (colocalizar) explicitamente essas tabelas. Quando duas ou mais tabelas são colocalizadas, as linhas dessas tabelas com os mesmos valores de chave de fragmento são colocadas no mesmo fragmento. A colocalização ajuda a restringir algumas operações a um único fragmento, o que resulta em melhor desempenho.

> **Note**
>
> Todas as chaves primárias e únicas devem incluir a chave de fragmento. Isso significa que a chave de fragmento é um subconjunto da chave primária ou única.
> As tabelas ilimitadas têm algumas limitações. Para ter mais informações, consulte Limitações de DDL e outras informações do Aurora PostgreSQL Limitless Database.

Tópicos

- Exemplos usando variáveis para criar tabelas ilimitadas
- Visualizações de tabelas de banco de dados Aurora PostgreSQL Limitless

Exemplos usando variáveis para criar tabelas ilimitadas

Os exemplos a seguir mostram como usar essas variáveis para criar tabelas fragmentadas e de referência.

Crie uma tabela fragmentada chamada `items`, com o `id` da a chave de fragmento.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items(id int, val int, item text);
COMMIT;
```

Crie uma tabela fragmentada chamada `items`, com uma chave de fragmento composta pelas colunas `item_id` e `item_cat`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
CREATE TABLE items(item_id int, item_cat varchar, val int, item text);
COMMIT;
```

Crie uma tabela fragmentada chamada `item_description`, com uma chave de fragmento composta pelas colunas `item_id` e `item_cat`, e colocalize-a com a tabela `items` do exemplo anterior.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
SET LOCAL rds_aurora.limitless_create_table_collocate_with='items';
CREATE TABLE item_description(item_id int, item_cat varchar, color_id int);
COMMIT;
```

Crie uma tabela de referência chamada `colors`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE colors(color_id int primary key, color varchar);
COMMIT;
```

Para redefinir a variável de sessão `rds_aurora.limitless_create_table_mode` como `standard`, use a seguinte declaração:

```
RESET rds_aurora.limitless_create_table_mode;
```

Depois de redefinir essa variável, as tabelas são criadas como tabelas padrão, que é o padrão. Para obter mais informações sobre tabelas padrão, consulte Converter tabelas padrão em tabelas ilimitadas.

Visualizações de tabelas de banco de dados Aurora PostgreSQL Limitless

Para obter informações sobre as tabelas do Limitless Database, use as visualizações a seguir.

rds_aurora.limitless_tables

A visualização `rds_aurora.limitless_tables` contém informações sobre tabelas ilimitadas e seus tipos.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | standard    | active       | standard    |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     |
 HASH (order_id)
         2 |     18579 | public      | customer    | active       | sharded     |
 HASH (cust_id)
(4 rows)
```

rds_aurora.limitless_table_collocations

A visualização `rds_aurora.limitless_table_collocations` contém informações sobre tabelas fragmentadas colocalizadas. Por exemplo, as tabelas `orders` e `customers` são colocalizadas e têm o mesmo `collocation_id`. As tabelas `users` e `followers` são colocalizadas e têm o mesmo `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
          16002 | public      | orders
          16002 | public      | customers
          16005 | public      | users
          16005 | public      | followers
(4 rows)
```

rds_aurora.limitless_table_collocation_distributions

A visualização `rds_aurora.limitless_table_collocation_distributions` mostra a distribuição de chaves para cada colocalização.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
          16002 |             6 | -9223372036854775808 | -4611686018427387904
          16002 |             5 | -4611686018427387904 |                    0
          16002 |             4 |                    0 |  4611686018427387904
          16002 |             3 |  4611686018427387904 |  9223372036854775807
          16005 |             6 | -9223372036854775808 | -4611686018427387904
          16005 |             5 | -4611686018427387904 |                    0
          16005 |             4 |                    0 |  4611686018427387904
          16005 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Converter tabelas padrão em tabelas ilimitadas

É possível converter tabelas padrão em tabelas fragmentadas ou de referência. Durante a conversão, os dados são movidos da tabela padrão para a tabela distribuída e, em seguida, a tabela padrão de origem é excluída. Os dados são movidos usando o comando `INSERT INTO SELECT FROM`.

Sumário

- Criar tabelas fragmentadas
- Criar tabelas colocalizadas
- Criar tabelas de referência

Criar tabelas fragmentadas

Você cria tabelas fragmentadas executando o procedimento `rds_aurora.limitless_alter_table_type_sharded` em tabelas padrão. Esse procedimento recebe uma tabela padrão e uma lista de colunas e, em seguida, distribui a tabela fornecida usando a lista de colunas como a chave de fragmento. O procedimento é executado de forma síncrona e adquire um bloqueio `ACCESS EXCLUSIVE` na tabela.

Depois que o procedimento for concluído com sucesso, a tabela padrão de origem será excluída e uma tabela fragmentada com o mesmo nome ficará disponível.

O procedimento `rds_aurora.limitless_alter_table_type_sharded` usa a seguinte sintaxe:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('schema.table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn']);
```

O procedimento requer os seguintes parâmetros:

- `schema`: o esquema do banco de dados que contém a tabela a ser fragmentada. Se o esquema não for especificado, o procedimento usa o `search_path`.
- `table`: a tabela a ser fragmentada.
- `shard_key`*n*: uma matriz de colunas da tabela para usar como chave de fragmento.

  Os valores da chave de fragmento são literais de string e, portanto, diferenciam maiúsculas de minúsculas. Se uma chave de fragmento contiver aspas simples ('), use outra aspa simples de

escape. Por exemplo, se uma coluna da tabela tiver o nome `customer's id`, use `customer''s id` como chave de fragmento. Barras invertidas (\) e aspas duplas (") não precisam de escape.

> **Note**
>
> Todas as chaves primárias e únicas devem incluir a chave de fragmento. Isso significa que a chave de fragmento é um subconjunto da chave primária ou única.
> Em tabelas fragmentadas, a restrição `CHECK` não é compatível com expressões.
> Para ter mais informações, consulte Restrições.

Como criar uma tabela fragmentada

O exemplo a seguir mostra como criar a tabela fragmentada `customer` com a chave de fragmento `customer_id`.

1. Crie a tabela padrão.

```
CREATE TABLE customer (customer_id INT PRIMARY KEY NOT NULL, zipcode INT, email
 VARCHAR);
```

2. Converta a tabela padrão em uma tabela fragmentada.

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.customer',
 ARRAY['customer_id']);

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
(6 rows)
```

Criar tabelas colocalizadas

Se duas ou mais tabelas forem fragmentadas usando a mesma chave de fragmento, você poderá alinhar (colocalizar) explicitamente essas tabelas. Quando duas ou mais tabelas são colocalizadas, as linhas dessas tabelas com os mesmos valores de chave de fragmento são colocadas no mesmo fragmento. A colocalização ajuda a restringir algumas operações a um único fragmento, o que resulta em melhor desempenho.

Você usa o procedimento `rds_aurora.limitless_alter_table_type_sharded` com a seguinte sintaxe:

```
postgres=> CALL
 rds_aurora.limitless_alter_table_type_sharded('schema.collocated_table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn'], 'schema.sharded_table');
```

O procedimento requer os seguintes parâmetros:

- `schema`: o esquema do banco de dados que contém as tabelas a serem colocalizadas. Se o esquema não for especificado, o procedimento usa o `search_path`.
- `collocated_table`: a tabela a ser colocalizada.
- `shard_key`*n*: uma matriz de colunas da tabela para usar como chave de fragmento.

  É necessário usar a mesma chave de fragmento da tabela fragmentada original, incluindo os mesmos nomes e tipos de coluna.
- `sharded_table`: a tabela fragmentada com a qual você está colocalizando o `collocated_table`.

Para criar uma tabela colocalizada

1. Crie a primeira tabela fragmentada seguindo o procedimento em Criar tabelas fragmentadas.
2. Crie a tabela padrão para a tabela colocalizada.

   ```
   CREATE TABLE mytable2 (customer_id INT PRIMARY KEY NOT NULL, column1 INT, column2
    VARCHAR);
   ```
3. Converta a tabela padrão em uma tabela colocalizada.

   ```
   postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.mytable2',
   ```

```
ARRAY['customer_id'], 'public.customer');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | mytable2     | partitioned table | postgres_limitless
 public | mytable2_fs1 | foreign table     | postgres_limitless
 public | mytable2_fs2 | foreign table     | postgres_limitless
 public | mytable2_fs3 | foreign table     | postgres_limitless
 public | mytable2_fs4 | foreign table     | postgres_limitless
 public | mytable2_fs5 | foreign table     | postgres_limitless
(12 rows)
```

Criar tabelas de referência

Você cria tabelas de referência executando o procedimento `rds_aurora.limitless_alter_table_type_reference` em tabelas padrão. Esse procedimento replica uma determinada tabela para todos os fragmentos no grupo de fragmentos de banco de dados e altera o tipo de tabela para referência. O procedimento é executado de forma síncrona e adquire um bloqueio `ACCESS EXCLUSIVE` na tabela.

Depois que o procedimento for concluído com sucesso, a tabela padrão de origem será excluída e uma tabela de referência com o mesmo nome ficará disponível.

O procedimento `rds_aurora.limitless_alter_table_type_reference` usa a seguinte sintaxe:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_reference('schema.table');
```

O procedimento armazenado requer os seguintes parâmetros:

- `schema`: o esquema do banco de dados que contém a tabela a ser replicada. Se o esquema não for especificado, o procedimento usa o `search_path`.

- `table`: a tabela a ser replicada.

> **Note**
>
> A tabela padrão a partir da qual você cria a tabela de referência deve ter uma chave primária.
> Nas tabelas de referência, a restrição CHECK não é compatível com expressões.
> A função anterior, `limitless_table_alter_type_reference`, está obsoleta.

Como criar uma tabela de referência

Os exemplos a seguir mostram como criar a tabela de referência `zipcodes`.

1. Crie a tabela padrão.

```
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);
```

2. Converta a tabela padrão em uma tabela de referência.

```
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcodes');

postgres=> \d

                    List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | zipcodes     | foreign table     | postgres_limitless
(7 rows)
```

A saída mostra a tabela fragmentada `customer` e a tabela de referência `zipcodes`.

Exemplos de esquemas do Aurora PostgreSQL Limitless Database

Fornecemos os seguintes exemplos de esquemas do Aurora PostgreSQL Limitless Database:

- Exemplo de esquema de comércio eletrônico Limitless
- Limitless pgbench

Você pode usar esses esquemas para criar rapidamente um banco de dados de amostra e carregar dados nas tabelas do Aurora PostgreSQL Limitless Database. Para obter mais informações, consulte o repositório GitHub.

Carregar dados no Aurora PostgreSQL Limitless Database

Você pode carregar dados nas tabelas do Aurora PostgreSQL Limitless Database usando o comando `COPY` ou o utilitário de carregamento de dados.

> **Note**
>
> É possível carregar dados em tabelas fragmentadas, de referência e padrão.

Sumário

- Usar o comando COPY com o Aurora PostgreSQL Limitless Database
  - Usar o comando COPY para carregar dados no Aurora PostgreSQL Limitless Database
    - Dividir dados em vários arquivos.
  - Usar o comando COPY para copiar dados do Limitless Database para um arquivo
- Usar o utilitário de carregamento de dados do Aurora PostgreSQL Limitless Database
  - Limitações
  - Pré-requisitos
  - Preparar o banco de dados de origem
  - Preparar o banco de dados de destino
  - Criar as credenciais do banco de dados
    - Criar as credenciais do banco de dados de origem
    - Criar as credenciais do banco de dados de destino
  - Configurar a autenticação do banco de dados e o acesso a recursos usando um script
    - Script de configuração para o utilitário de carregamento de dados
    - Saída do script de configuração do utilitário de carregamento de dados
    - Limpar recursos com falha
  - Configurar a autenticação do banco de dados e o acesso a recursos manualmente
    - Criar a AWS KMS key gerenciada pelo cliente
    - Criar os segredos do banco de dados
    - Criar o perfil do IAM
    - Atualizar a AWS KMS key gerenciada pelo cliente
    - Adicionar as políticas de permissão do perfil do IAM

- Carregar dados de um cluster de banco de dados do Aurora PostgreSQL ou de uma instância de banco de dados do RDS para PostgreSQL
- Monitorar o carregamento de dados
  - Listar trabalhos de carregamento de dados
  - Visualizar detalhes dos trabalhos de carregamento de dados usando o ID do trabalho
  - Monitorar o grupo de logs do Amazon CloudWatch
  - Monitorar eventos do RDS
- Cancelamento do carregamento de dados

Usar o comando COPY com o Aurora PostgreSQL Limitless Database

É possível usar a funcionalidade \copy no utilitário `psql` para importar e exportar dados do Aurora PostgreSQL Limitless Database

Usar o comando COPY para carregar dados no Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database é compatível com a funcionalidade \copy no utilitário `psql` para importar dados.

No Limitless Database, assim como no Aurora PostgreSQL, não há compatibilidade com o seguinte:

- Acesso SSH direto às instâncias de banco de dados: não é possível copiar um arquivo de dados (como no formato .csv) para o host da instância de banco de dados e executar `COPY` a partir do arquivo.
- Usar arquivos locais na instância de banco de dados: use `COPY ... FROM STDIN` e `COPY ... TO STDOUT`.

O comando `COPY` no PostgreSQL tem opções para trabalhar com arquivos locais (`FROM/TO`) e transmitir dados usando uma conexão entre o cliente e o servidor (`STDIN/STDOUT`). Para obter mais informações, consulte COPY na documentação do PostgreSQL.

O comando `\copy` no utilitário `psql` do PostgreSQL funciona com arquivos locais no computador em que você executa o cliente `psql`. Ele invoca o respectivo comando `COPY ... FROM STDIN` ou `COPY ... FROM STDOUT` no servidor remoto (por exemplo, Limitless Database) ao qual você se conecta. Ele lê os dados do arquivo local para `STDIN` ou grava nele a partir de `STDOUT`.

Dividir dados em vários arquivos.

Os dados são armazenados em vários fragmentos no Aurora PostgreSQL Limitless Database. Para acelerar o carregamento de dados usando `\copy`, é possível dividir os dados em vários arquivos. Em seguida, importe de forma independente para cada arquivo de dados executando comandos `\copy` separados em paralelo.

Por exemplo, você tem um arquivo de dados de entrada no formato CSV com 3 milhões de linhas para importar. Você pode dividir o arquivo em partes, cada uma contendo 200.000 linhas (15 partes):

```
split -l200000 data.csv data_ --additional-suffix=.csv -d
```

Isso resulta em arquivos `data_00.csv` a `data_14.csv`. Em seguida, é possível importar dados usando 15 comandos `\copy` paralelos, por exemplo:

```
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table from '/tmp/data_00.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_01.csv';" postgres_limitless &
...
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_13.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_14.csv';" postgres_limitless
```

Usando essa técnica, a mesma quantidade de dados é importada aproximadamente 10 vezes mais rápido do que usando um único comando `\copy`.

Usar o comando COPY para copiar dados do Limitless Database para um arquivo

É possível usar o comando \copy para copiar dados de uma tabela ilimitada para um arquivo, conforme mostrado no seguinte exemplo:

```
postgres_limitless=> \copy test_table TO '/tmp/test_table.csv' DELIMITER ',' CSV
 HEADER;
```

Usar o utilitário de carregamento de dados do Aurora PostgreSQL Limitless Database

O Aurora fornece um utilitário para carregar dados diretamente no Limitless Database a partir de um cluster de banco de dados do Aurora PostgreSQL ou de uma instância de banco de dados do RDS para PostgreSQL.

Execute as seguintes etapas para usar o utilitário de carregamento de dados:

1. Pré-requisitos
2. Preparar o banco de dados de origem
3. Preparar o banco de dados de destino
4. Criar as credenciais do banco de dados
5. Um dos seguintes:
   - Configurar a autenticação do banco de dados e o acesso a recursos usando um script (recomendado)
   - Configurar a autenticação do banco de dados e o acesso a recursos manualmente
6. Carregar dados de um cluster de banco de dados do Aurora PostgreSQL ou de uma instância de banco de dados do RDS para PostgreSQL

Limitações

O utilitário de carregamento de dados tem as seguintes limitações:

- Os seguintes tipos de dados não são compatíveis: `enum`, `ARRAY`, `BOX`, `CIRCLE`, `LINE`, `LSEG`, `PATH`, `PG_LSN`, `PG_SNAPSHOT`, `POLYGON`, `TSQUERY`, `TSVECTOR` e `TXID_SNAPSHOT`.
- Os zeros à esquerda (`0`) são removidos do tipo de dados `VARBIT` durante o carregamento.
- A migração de dados falha quando há chaves estrangeiras nas tabelas de destino.
- O utilitário Limitless Data permite as seguintes configurações de origem para clusters de banco de dados multi-AZ do Amazon RDS para PostgreSQL:
  - Instância principal
    - Modos permitidos: snapshot, snapshot_then_cdc.
  - Instância de réplica
    - Modo permitido: somente snapshot.

    - Requisito: hot_standby_feedback deve estar habilitado.
  - Não permitido: snapshot_then_cdc.

Pré-requisitos

O utilitário de carregamento de dados tem os seguintes pré-requisitos:

- O banco de dados de origem usa o Aurora PostgreSQL ou o RDS para PostgreSQL versão 11.x e superior.
- O banco de dados de origem está na mesma Conta da AWS e Região da AWS que o grupo de fragmentos de banco de dados de destino.
- O cluster de banco de dados ou a instância de banco de dados de origem está no estado `available`.
- As tabelas no banco de dados de origem e no banco de dados ilimitado têm os mesmos nomes de tabela, nomes de colunas e tipos de dados de coluna.
- As tabelas de origem e destino têm chaves primárias que usam as mesmas colunas e ordens de colunas.
- Você deve ter um ambiente para se conectar a um banco de dados ilimitado para executar comandos de carregamento de dados. Os comandos disponíveis são os seguintes:
  - `rds_aurora.limitless_data_load_start`
  - `rds_aurora.limitless_data_load_cancel`
- Para CDC:
  - Tanto o banco de dados de origem quanto o grupo de fragmentos de banco de dados de destino devem usar o mesmo grupo de sub-rede de banco de dados, grupo de segurança de VPC e porta de banco de dados. Essas configurações são para conexões de rede com o banco de dados de origem e os roteadores no grupo de fragmentos de banco de dados.
  - Você deve habilitar a replicação lógica no banco de dados de origem. O usuário do banco de dados de origem deve ter privilégios para ler a replicação lógica.

Preparar o banco de dados de origem

Para acessar o banco de dados de origem para carregamento de dados, você deve permitir o tráfego de rede de entrada para ele. Siga as etapas a seguir.

Como permitir o tráfego de rede para o banco de dados de origem

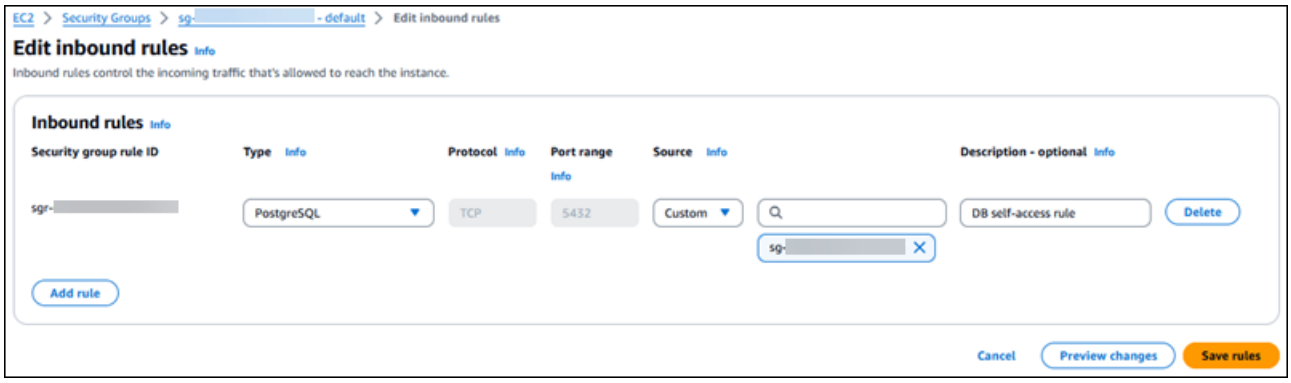
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Navegue até a página Grupos de segurança.
3. Escolha o ID do grupo de segurança usado pelo cluster ou instância de banco de dados de origem.

   Por exemplo, o ID do grupo de segurança é `sg-056a84f1712b77926`.
4. Na guia Regras de entrada:

   a. Escolha Editar regras de entrada.

   b. Adicione uma nova regra de entrada para o cluster ou instância de banco de dados de origem:

      - Intervalo de portas: porta de banco de dados para o banco de dados de origem, geralmente `5432`
      - ID do grupo de segurança: `sg-056a84f1712b77926` neste exemplo

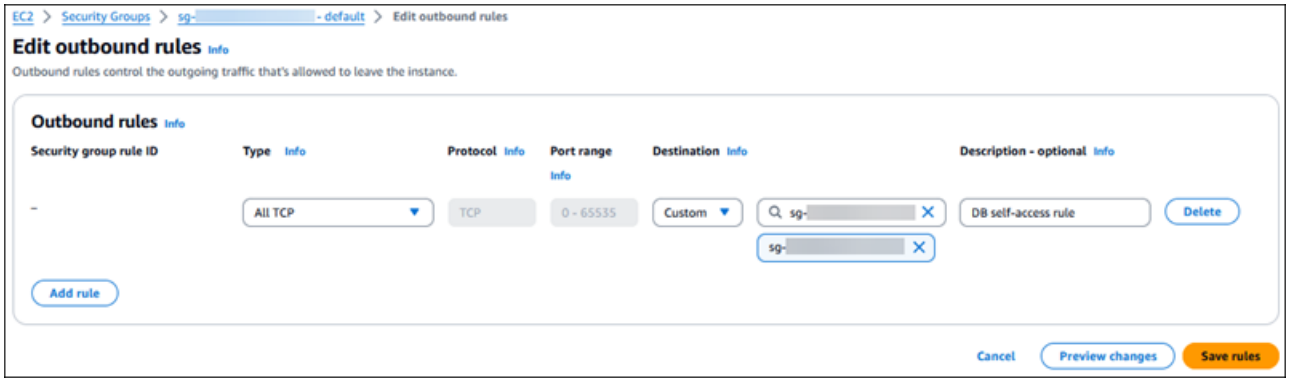


5. Na guia Regras de saída:

   a. Escolha Editar regras de saída.

   b. Adicione uma nova regra de saída para o cluster ou instância de banco de dados de origem:

      - Porta do banco de dados: `All traffic` (inclui portas `0-65535`)
      - ID do grupo de segurança: `sg-056a84f1712b77926` neste exemplo



6. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
7. Navegue até a página ACLs de rede.
8. Adicione a configuração padrão da ACL de rede conforme descrito em ACL de rede padrão.

Preparar o banco de dados de destino

Siga os procedimentos em Criação de tabelas do Aurora PostgreSQL Limitless Database para criar as tabelas de destino no grupo de fragmentos de banco de dados.

Suas tabelas de destino devem ter os mesmos esquemas, nomes de tabela e chaves primárias das tabelas de origem.

Criar as credenciais do banco de dados

É necessário criar usuários de banco de dados nos bancos de dados de origem e destino e conceder os privilégios necessários aos usuários. Para obter mais informações, consulte CREATE USER e GRANT na documentação do PostgreSQL.

Criar as credenciais do banco de dados de origem

O usuário do banco de dados de origem é passado no comando para iniciar o carregamento. Esse usuário deve ter privilégios para executar a replicação do banco de dados de origem.

1. Use o usuário principal do banco de dados (ou outro usuário com a função `rds_superuser`) para criar um usuário do banco de dados de origem com privilégios `LOGIN`.

```
CREATE USER source_db_username WITH PASSWORD 'source_db_user_password';
```

2. Conceda a função `rds_superuser` ao usuário do banco de dados de origem.

```
GRANT rds_superuser to source_db_username;
```

3. Se você estiver usando o modo `full_load_and_cdc`, conceda a função `rds_replication` ao usuário do banco de dados de origem. O perfil `rds_replication` concede permissões para gerenciar slots lógicos e transmitir dados utilizando slots lógicos.

```
GRANT rds_replication to source_db_username;
```

Criar as credenciais do banco de dados de destino

O usuário do banco de dados de destino deve ter permissão para gravar nas tabelas de destino no grupo de fragmentos de banco de dados.

1. Use o usuário principal do banco de dados (ou outro usuário com a função `rds_superuser`) para criar um usuário do banco de dados de destino com privilégios `LOGIN`.

```
CREATE USER destination_db_username WITH PASSWORD 'destination_db_user_password';
```

2. Conceda a função `rds_superuser` ao usuário do banco de dados de destino.

```
GRANT rds_superuser to destination_db_username;
```

Configurar a autenticação do banco de dados e o acesso a recursos usando um script

O script de configuração cria uma AWS KMS key gerenciada pelo cliente, um perfil do AWS Identity and Access Management (IAM) e dois segredos do AWS Secrets Manager.

Execute as etapas a seguir para usar o script de configuração:

1. A AWS CLI deve estar instalada e configurada com as credenciais da sua Conta da AWS.
2. Instale o processador JSON de linha de comando `jq`. Para obter mais informações, consulte jqlang/jq.
3. Copie o arquivo data_loading_script.zip para o seu computador e extraia o arquivo `data_load_aws_setup_script.sh` dele.
4. Edite o script para substituir as variáveis de espaço reservado pelos valores apropriados para o seguinte:
   - Sua Conta da AWS
   - O Região da AWS
   - Credenciais do banco de dados de origem
   - Credenciais do banco de dados de destino
5. Abra um novo terminal no computador e execute o seguinte comando:

```
bash ./data_load_aws_setup_script.sh
```

Script de configuração para o utilitário de carregamento de dados

Fornecemos o texto do arquivo `data_load_aws_setup_script.sh` aqui para referência.

```
#!/bin/bash
# Aurora Limitless data loading - AWS resources setup script #
# Set up the account credentials in advance. #
# Update the following script variables. #

###################################
#### Start of variable section ####

ACCOUNT_ID="12-digit_AWS_account_ID"
REGION="AWS_Region"
DATE=$(date +'%m%d%H%M%S')
RANDOM_SUFFIX="${DATE}"
```

```
SOURCE_SECRET_NAME="secret-source-${DATE}"
SOURCE_USERNAME="source_db_username"
SOURCE_PASSWORD="source_db_password"
DESTINATION_SECRET_NAME="secret-destination-${DATE}"
DESTINATION_USERNAME="destination_db_username"
DESTINATION_PASSWORD="destination_db_password"
DATA_LOAD_IAM_ROLE_NAME="aurora-data-loader-${RANDOM_SUFFIX}"
TMP_WORK_DIR="./tmp_data_load_aws_resource_setup/"

#### End of variable section ####
################################

# Main logic start
echo "DATE - [${DATE}]"
echo "RANDOM_SUFFIX - [${RANDOM_SUFFIX}]"
echo 'START!'

mkdir -p $TMP_WORK_DIR

# Create the symmetric KMS key for encryption and decryption.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_key_response.txt"
aws kms create-key --region $REGION | tee $TMP_FILE_PATH
KMS_KEY_ARN=$(cat $TMP_FILE_PATH | jq -r '.KeyMetadata.Arn')
aws kms create-alias \
    --alias-name alias/"${DATA_LOAD_IAM_ROLE_NAME}-key" \
    --target-key-id $KMS_KEY_ARN \
    --region $REGION

# Create the source secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_source_secret_response.txt"
aws secretsmanager create-secret \
    --name $SOURCE_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$SOURCE_USERNAME\",\"password\":\"$SOURCE_PASSWORD
\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
SOURCE_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the destination secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_destination_secret_response.txt"
aws secretsmanager create-secret \
    --name $DESTINATION_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
```

```
    --secret-string "{\"username\":\"$DESTINATION_USERNAME\",\"password\":
\"$DESTINATION_PASSWORD\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
DESTINATION_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the RDS trust policy JSON file.
# Use only rds.amazonaws.com for RDS PROD use cases.
TRUST_POLICY_PATH="${TMP_WORK_DIR}rds_trust_policy.json"
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}' > $TRUST_POLICY_PATH

# Create the IAM role.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_iam_role_response.txt"
aws iam create-role \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --assume-role-policy-document "file://${TRUST_POLICY_PATH}" \
    --tags Key=assumer,Value=aurora_limitless_table_data_load \
    --region $REGION \
    | tee $TMP_FILE_PATH
IAM_ROLE_ARN=$(cat $TMP_FILE_PATH | jq -r '.Role.Arn')

# Create the permission policy JSON file.
PERMISSION_POLICY_PATH="${TMP_WORK_DIR}data_load_permission_policy.json"
permission_json_policy=$(cat &lt;&lt;EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "$SOURCE_SECRET_ARN",
                "$DESTINATION_SECRET_ARN"
            ]
        },
        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "$KMS_KEY_ARN"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
```

```
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $permission_json_policy > $PERMISSION_POLICY_PATH

# Add the inline policy.
aws iam put-role-policy \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --policy-name aurora-limitless-data-load-policy \
    --policy-document "file://${PERMISSION_POLICY_PATH}" \
    --region $REGION

# Create the key policy JSON file.
KEY_POLICY_PATH="${TMP_WORK_DIR}data_load_key_policy.json"
key_json_policy=$(cat &lt;&lt;EOF
{
    "Id": "key-aurora-limitless-data-load-$RANDOM_SUFFIX",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "$IAM_ROLE_ARN"
            },
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "*"
        }
    ]
}
EOF
)
echo $key_json_policy > $KEY_POLICY_PATH

# Add the key policy.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_put_key_policy_response.txt"
sleep 10 # sleep 10 sec for IAM role ready
aws kms put-key-policy \
    --key-id $KMS_KEY_ARN \
    --policy-name default \
    --policy "file://${KEY_POLICY_PATH}" \
    --region $REGION \
    | tee $TMP_FILE_PATH

echo 'DONE!'

echo "ACCOUNT_ID : [${ACCOUNT_ID}]"
echo "REGION : [${REGION}]"
echo "RANDOM_SUFFIX : [${RANDOM_SUFFIX}]"
echo "IAM_ROLE_ARN : [${IAM_ROLE_ARN}]"
echo "SOURCE_SECRET_ARN : [${SOURCE_SECRET_ARN}]"
echo "DESTINATION_SECRET_ARN : [${DESTINATION_SECRET_ARN}]"

# Example of a successful run:
# ACCOUNT_ID : [012345678912]
# REGION : [ap-northeast-1]
# RANDOM_SUFFIX : [0305000703]
# IAM_ROLE_ARN : [arn:aws:iam::012345678912:role/aurora-data-loader-0305000703]
# SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-source-0305000703-yQDtow]
# DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-destination-0305000703-5d5Jy8]

# If you want to manually clean up failed resource,
# please remove them in the following order:
# 1. IAM role.
 # aws iam delete-role-policy --role-name Test-Role --policy-name ExamplePolicy --
region us-east-1
# aws iam delete-role --role-name Test-Role --region us-east-1
# 2. Source and destination secrets.
```

```
# aws secretsmanager delete-secret --secret-id MyTestSecret --force-delete-without-
recovery --region us-east-1
# 3. KDM key.
# aws kms schedule-key-deletion --key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab --pending-window-in-days 7
 --region us-east-1
```

Saída do script de configuração do utilitário de carregamento de dados

O exemplo a seguir mostra a saída de uma execução bem-sucedida do script.

```
% bash ./data_load_aws_setup_script.sh
DATE - [0305000703]
RANDOM_SUFFIX - [0305000703]
START!
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:ap-
northeast-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-03-05T00:07:49.852000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow",
    "Name": "secret-source-0305000703",
    "VersionId": "a017bebe-a71b-4220-b923-6850c2599c26"
}
{
```

```
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
destination-0305000703-5d5Jy8",
    "Name": "secret-destination-0305000703",
    "VersionId": "32a1f989-6391-46b1-9182-f65d242f5eb6"
}
{
    "Role": {
        "Path": "/",
        "RoleName": "aurora-data-loader-0305000703",
        "RoleId": "AROAYPX63ITQOYORQSC6U",
        "Arn": "arn:aws:iam::123456789012:role/aurora-data-loader-0305000703",
        "CreateDate": "2024-03-05T00:07:54+00:00",
        "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "rds.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "assumer",
                "Value": "aurora_limitless_table_data_load"
            }
        ]
    }
}
DONE!
ACCOUNT_ID : [123456789012]
REGION : [ap-northeast-1]
RANDOM_SUFFIX : [0305000703]
IAM_ROLE_ARN : [arn:aws:iam::123456789012:role/aurora-data-loader-0305000703]
SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow]
DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:123456789012:secret:secret-destination-0305000703-5d5Jy8]
```

Limpar recursos com falha

Se você quiser limpar recursos com falha manualmente, remova-os na seguinte ordem:

1. Perfil do IAM, por exemplo:

```
aws iam delete-role-policy \
--role-name Test-Role \
--policy-name ExamplePolicy

aws iam delete-role \
--role-name Test-Role
```

2. Segredos de origem e destino, por exemplo:

```
aws secretsmanager delete-secret \
--secret-id MyTestSecret \
--force-delete-without-recovery
```

3. Chave do KMS, por exemplo:

```
aws kms schedule-key-deletion \
--key-id arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
 \
--pending-window-in-days 7
```

Em seguida, você pode tentar executar o script novamente.

Configurar a autenticação do banco de dados e o acesso a recursos manualmente

O processo manual para configurar a autenticação do banco de dados e o acesso aos recursos tem as seguintes etapas:

1. Criar a AWS KMS key gerenciada pelo cliente
2. Adicionar as políticas de permissão do perfil do IAM
3. Criar os segredos do banco de dados
4. Criar o perfil do IAM
5. Atualizar a AWS KMS key gerenciada pelo cliente

Esse processo é opcional e executa as mesmas tarefas que em Configurar a autenticação do banco de dados e o acesso a recursos usando um script. Recomendamos usar o script.

Criar a AWS KMS key gerenciada pelo cliente

Siga os procedimentos em Creating symmetric encryption keys para criar uma chave do KMS gerenciada pelo cliente. Também é possível usar uma chave existente se ela atender a estes requisitos.

Como criar uma chave do KMS gerenciada pelo cliente

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS KMS em https://console.aws.amazon.com/kms.
2. Navegue até a página Chaves gerenciadas pelo cliente.
3. Escolha Criar chave.
4. Na página Configurar chave:
    a. Em Tipo de chave, selecione Simétrico.
    b. Em Uso da chave, selecione Criptografar e descriptografar.
    c. Escolha Próximo.
5. Na página Adicionar rótulos, insira um Alias, como **`limitless`**, e escolha Próximo.
6. Na página Definir permissões administrativas da chave, verifique se a caixa de seleção Permitir que administradores de chaves excluam esta chave está marcada e escolha Próximo.
7. Na página Definir permissões de uso da chave, escolha Próximo.

8. Na página Review (Rever), escolha Finish (Concluir).

   Você atualizará a política de chaves posteriormente.

Registre os nomes do recurso da Amazon (ARN) da chave do KMS a ser utilizada em Adicionar as políticas de permissão do perfil do IAM.

Para obter informações sobre como usar a AWS CLI para criar a chave do KMS gerenciada pelo cliente, consulte create-key e create-alias.

Criar os segredos do banco de dados

Para permitir que o utilitário de carregamento de dados acesse as tabelas do banco de dados de origem e destino, crie dois segredos no AWS Secrets Manager: um para o banco de dados de origem e outro para o banco de dados de destino. Esses segredos armazenam os nomes de usuário e senhas para acessar os bancos de dados de origem e destino.

Siga os procedimentos em Create an AWS Secrets Manager secret para criar os segredos de par de chave/valor.

Como criar os segredos do banco de dados

1. Abra o console Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Armazenar Novo Segredo.
3. Na página Escolher tipo de segredo:

   a. Em Tipo de segredo, selecione Outro tipo de segredo.

   b. Em Pares de chave/valor, escolha a guia Texto simples.

   c. Insira o código JSON a seguir, onde *sourcedbreader* e *sourcedbpassword* são as credenciais do usuário do banco de dados de origem de Criar as credenciais do banco de dados de origem.

   ```
   {
       "username":"sourcedbreader",
       "password":"sourcedbpassword"
   }
   ```

   d. Em Chave de criptografia, escolha a chave do KMS que você criou em Criar a AWS KMS key gerenciada pelo cliente, por exemplo `limitless`.

e. Escolha Próximo.

4. Na página Configurar segredo, insira um Nome do segredo, como **`source_DB_secret`**, e escolha Próximo.
5. Na página Configurar alternância - opcional, escolha Próximo.
6. Na página Review (Análise), escolha Store (Armazenar).
7. Repita o procedimento para o segredo do banco de dados de destino:

   a. Insira o código JSON a seguir, onde *`destinationdbwriter`* e *`destinationdbpassword`* são as credenciais do usuário do banco de dados de destino de Criar as credenciais do banco de dados de destino.

   ```
   {
       "username":"destinationdbwriter",
       "password":"destinationdbpassword"
   }
   ```

   b. Insira um Nome do segredo, como **`destination_DB_secret`**.

Registre os ARNs dos segredos a serem usados em Adicionar as políticas de permissão do perfil do IAM.

Criar o perfil do IAM

O carregamento de dados exige que você forneça acesso aos recursos da AWS. Para fornecer acesso, você cria o perfil do IAM `aurora-data-loader` seguindo os procedimentos descritos em Criar um perfil para delegar permissões a um usuário do IAM.

Para criar perfil do IAM

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. Navegue até o página Perfis.
3. Selecione Criar perfil.
4. Na página Selecionar entidade confiável:

   a. Em Tipo de entidade confiável, selecione Política de confiança personalizada.

   b. Insira o seguinte código JSON para a política de confiança personalizada:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

c. Escolha Próximo.

5. Na página Adicionar permissões, escolha Próximo.
6. Na página Nomear, revisar e criar:

   a. Em Nome do perfil, insira **`aurora-data-loader`** ou outro nome que preferir.
   b. Selecione Adicionar etiqueta e insira a seguinte etiqueta:

      - Chave: **`assumer`**
      - Valor: **`aurora_limitless_table_data_load`**

> **Important**
>
> O Aurora PostgreSQL Limitless Database só pode assumir um perfil do IAM que tenha essa etiqueta.

   c. Selecione Criar perfil.

Atualizar a AWS KMS key gerenciada pelo cliente

Siga os procedimentos em Changing a key policy para adicionar o perfil do IAM `aurora-data-loader` à política de chave padrão.

Como adicionar o perfil do IAM à política de chave

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS KMS em https://console.aws.amazon.com/kms.
2. Navegue até a página Chaves gerenciadas pelo cliente.
3. Escolha a chave do KMS que você criou em Criar a AWS KMS key gerenciada pelo cliente, por exemplo `limitless`.
4. Na guia Política de chaves, em Usuários de chaves, escolha Adicionar.
5. Na janela Adicionar usuários de chaves, selecione o nome do perfil do IAM que você criou em Criar o perfil do IAM, por exemplo aurora-data-loader.
6. Escolha Adicionar.

Adicionar as políticas de permissão do perfil do IAM

É necessário adicionar políticas de permissão ao perfil do IAM que você criou. Isso permite que o utilitário de carregamento de dados do Aurora PostgreSQL Limitless Database acesse os recursos relacionados da AWS para criar conexões de rede e recuperar os segredos das credenciais dos bancos de dados de origem e destino.

Para obter mais informações, consulte Modificar uma função.

Como adicionar as políticas de permissão

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. Navegue até o página Perfis.
3. Escolha o perfil do IAM que você criou em Criar o perfil do IAM, por exemplo aurora-data-loader.
4. Na guia Permissões, em Políticas de permissões, escolha Adicionar permissões e, em seguida, Criar política em linha.
5. Na página Especifique permissões, escolha o editor JSON.
6. Copie e cole o modelo a seguir no editor JSON, substituindo os espaços reservados pelos ARNs dos segredos do banco de dados e da chave do KMS.

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:source_DB_secret-ABC123",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:destination_DB_secret-456DEF"
            ]
        },        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/aa11bb22-
####-####-####-fedcba123456"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Verifique se há erros e corrija-os.
8. Escolha Próximo.
9. Na página Revisar e criar, insira um Nome da política, como **`data_loading_policy`**, e escolha Criar política.

Carregar dados de um cluster de banco de dados do Aurora PostgreSQL ou de uma instância de banco de dados do RDS para PostgreSQL

Depois de concluir a configuração do recurso e da autenticação, conecte-se ao endpoint do cluster e chame o procedimento armazenado `rds_aurora.limitless_data_load_start` de um banco de dados ilimitado, como `postgres_limitless`. O banco de dados ilimitado é um banco de dados no grupo de fragmentos de banco de dados para o qual você deseja migrar dados.

Essa função se conecta de forma assíncrona em segundo plano ao banco de dados de origem especificado no comando, lê os dados da fonte e carrega os dados nos fragmentos. Para um melhor desempenho, os dados são carregados usando threads paralelas. A função recupera um snapshot da tabela em um ponto no tempo executando um comando `SELECT` para ler os dados das tabelas fornecidas no comando.

É possível carregar dados em tabelas fragmentadas, de referência e padrão.

É possível carregar dados no nível do banco de dados, esquema ou tabela nas chamadas `rds_aurora.limitless_data_load_start`.

- Banco de dados: é possível carregar um banco de dados por vez em cada chamada, sem limite no número de esquemas ou tabelas dentro do banco de dados.
- Esquema: é possível carregar no máximo 15 esquemas em cada chamada, sem limite no número de tabelas em cada esquema.
- Tabela: você pode carregar no máximo 15 tabelas em cada chamada.

> **Note**
>
> Este recurso não usa snapshots do Amazon RDS ou isolamento em um ponto no tempo do banco de dados. Para manter a consistência entre as tabelas, recomendamos clonar o banco de dados de origem e apontar para esse banco de dados clonado como a fonte.

O procedimento armazenado usa a seguinte sintaxe:

```
CALL rds_aurora.limitless_data_load_start('source_type',
    'source_DB_cluster_or_instance_ID',
    'source_database_name',
    'streaming_mode',
    'data_loading_IAM_role_arn',
    'source_DB_secret_arn',
    'destination_DB_secret_arn',
    'ignore_primary_key_conflict_boolean_flag',
    'is_dry_run',
    (optional parameter) schemas/tables => ARRAY['name1', 'name2', ...]);
```

Os parâmetros de entrada são os seguintes:

- `source_type`: o tipo de fonte: `aurora_postgresql` ou `rds_postgresql`
- `source_DB_cluster_or_instance_ID`: o identificador do cluster de banco de dados do Aurora PostgreSQL de origem ou o identificador da instância de banco de dados do RDS para PostgreSQL
- `source_database_name`: o nome do banco de dados de origem, como *`postgres`*
- `streaming_mode`: se deve incluir a captura de dados de alteração (CDC): `full_load` ou `full_load_and_cdc`
- `data_loading_IAM_role_arn`: o nome do recurso da Amazon (ARN) do perfil do IAM para `aurora-data-loader`
- `source_DB_secret_arn`: o ARN do segredo do banco de dados de origem
- `destination_DB_secret_arn`: o ARN do segredo do banco de dados de destino
- `ignore_primary_key_conflict_boolean_flag`: se deve continuar se ocorrer um conflito de chave primária:
  - Se definido como `true`, o carregamento de dados ignora novas alterações nas linhas com um conflito de chave primária.
  - Se definido como `false`, o carregamento de dados substitui as linhas existentes nas tabelas de destino quando encontra um conflito de chave primária.
- `is_dry_run`: se deve testar se o trabalho de carregamento de dados pode se conectar aos bancos de dados de origem e destino:
  - Se definido como `true`, testa as conexões sem carregar dados
  - Se definido como `false`, carrega os dados

- (opcional) `schemas` ou `tables`: uma matriz de esquemas ou tabelas para carregar. Você pode especificar qualquer um destes valores:
  - Uma lista de tabelas no formato `tables => ARRAY['schema1.table1', 'schema1.table2', 'schema2.table1', ...]`
  - Uma lista de esquemas no formato `schemas => ARRAY['schema1', 'schema2', ...]`

  Se você não incluir esse parâmetro, todo o banco de dados de origem especificado será migrado.

O parâmetro de saída é o ID do trabalho com uma mensagem.

O exemplo a seguir mostra como usar o procedimento armazenado `rds_aurora.limitless_data_load_start` para carregar dados de um cluster de banco de dados do Aurora PostgreSQL.

```
CALL rds_aurora.limitless_data_load_start('aurora_postgresql',
    'my-db-cluster',
    'postgres',
    'full_load_and_cdc',
    'arn:aws:iam::123456789012:role/aurora-data-loader-8f2c66',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-source-8f2c66-EWrr0V',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-destination-8f2c66-
d04fbD',
    'true',
    'false',
    tables => ARRAY['public.customer', 'public.order', 'public.orderdetails']);

INFO: limitless data load job id 1688761223647 is starting.
```

Monitorar o carregamento de dados

O Aurora PostgreSQL Limitless Database fornece várias maneiras de monitorar trabalhos de carregamento de dados:

- Listar trabalhos de carregamento de dados
- Visualizar detalhes dos trabalhos de carregamento de dados usando o ID do trabalho
- Monitorar o grupo de logs do Amazon CloudWatch
- Monitorar eventos do RDS

Listar trabalhos de carregamento de dados

Você pode se conectar ao endpoint do cluster e usar a visualização `rds_aurora.limitless_data_load_jobs` para listar os trabalhos de carregamento de dados.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_jobs LIMIT 6;

    job_id     |  status   | message |     source_db_identifier      | source_db_name |
 full_load_complete_time |
   progress_details                                                                 |
     start_time       |   last_updated_time    |  streaming_mode   | source_engine_type
 | ignore_primary_key_conflict | is_dryrun
--------------+----------+---------+-------------------------------+---------------
+-------------------------
+-------------------------------------------------------------------------------------------------------------------------------
+-----------------------+-----------------------+------------------
+--------------------+-----------------------------+-----------
 1725697520693 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:48:15+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "9
 of 9 tables loaded", "COMPLETED_AT": "2024/09/07 08:48:15+00", "RECORDS_MIGRATED":
 600003}} | 2024-09-07 08:47:13+00 | 2024-09-07 08:48:15+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696114225 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:24:20+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "3
 of 3 tables loaded", "COMPLETED_AT": "2024/09/07 08:24:20+00", "RECORDS_MIGRATED":
 200001}} | 2024-09-07 08:23:56+00 | 2024-09-07 08:24:20+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696067630 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:23:45+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "6
 of 6 tables loaded", "COMPLETED_AT": "2024/09/07 08:23:45+00", "RECORDS_MIGRATED":
```

```
 400002}} | 2024-09-07 08:23:10+00 | 2024-09-07 08:23:45+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725694221753 | CANCELED  |         | persistent-kdm-auto-source-01 | postgres
    |                         | {}

           | 2024-09-07 07:31:18+00 | 2024-09-07 07:51:49+00 | full_load_and_cdc |
 aurora_postgresql  | t                           | f
 1725691698210 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:51+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:51+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:42+00 | 2024-09-07 07:10:52+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725691695049 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:48+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:48+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:41+00 | 2024-09-07 07:10:48+00 | full_load         |
 aurora_postgresql  | t                           | f
(6 rows)
```

Os registros de trabalho são excluídos após 90 dias.

Visualizar detalhes dos trabalhos de carregamento de dados usando o ID do trabalho

Se você souber um ID de trabalho, poderá se conectar ao endpoint do cluster e usar a visualização `rds_aurora.limitless_data_load_job_details` para ver os detalhes desse trabalho de carregamento de dados, incluindo o nome da tabela, o status do trabalho e o número de linhas carregadas. Você pode obter o ID do trabalho nas respostas às funções de início do carregamento de dados ou na visualização `rds_aurora.limitless_data_load_jobs`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_job_details WHERE
 job_id='1725696114225';

job_id        | destination_table_name | destination_schema_name | start_time
      | status    | full_load_rows | full_load_total_rows | full_load_complete_time |
 cdc_insert | cdc_update | cdc_delete
--------------+------------------------+-------------------------
+-------------------------+-----------+----------------+---------------------
+-------------------------+------------+------------+------------
1725696114225 | standard_1             | public                  | 2024-09-07
 08:23:57+00 | COMPLETED | 100000         | 100000              | 2024-09-07
 08:24:08+00  | 0          | 0          | 0
```

```
1725696114225 | standard_2             | public                  | 2024-09-07
 08:24:08+00 | COMPLETED | 100000        | 100000                | 2024-09-07
 08:24:17+00  | 0          | 0          | 0
1725696114225 | standard_3             | public                  | 2024-09-07
 08:24:18+00 | COMPLETED | 1             | 1                     | 2024-09-07
 08:24:20+00  | 0          | 0          | 0
1725696114225 | standard_4             | public                  | 2024-09-07
 08:23:58+00 | PENDING   | 0             | 0                     |
    | 0          | 0          | 0
(4 rows)
```

Os registros de trabalho são excluídos após 90 dias.

Monitorar o grupo de logs do Amazon CloudWatch

Depois que o status do trabalho de carregamento de dados mudar para `RUNNING`, você poderá verificar o progresso do runtime usando o Amazon CloudWatch Logs.

Como monitorar os fluxos de logs do CloudWatch

Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.

1. Navegue até Logs e, em seguida, Grupos de logs.
2. Escolha o grupo de logs /aws/rds/aurora-limitless-database.
3. Pesquise o fluxo de logs do seu trabalho de carregamento de dados por job_id.

   O fluxo de logs tem o padrão Data-Load-Job-***job_id***.
4. Escolha o fluxo de logs para ver os eventos de logs.

Cada fluxo de logs mostra eventos contendo o status do trabalho e o número de linhas carregadas nas tabelas de destino do Aurora PostgreSQL Limitless Database. Se um trabalho de carregamento de dados falhar, também será criado um log de erros que mostra o status da falha e o motivo.

Os registros de trabalho são excluídos após 90 dias.

Monitorar eventos do RDS

O trabalho de carregamento de dados também publica eventos do RDS, por exemplo, quando um trabalho é bem-sucedido, falha ou é cancelado. É possível visualizar os eventos a partir do banco de dados de destino.

Para ter mais informações, consulte Eventos de grupo de fragmentos de banco de dados.

Cancelamento do carregamento de dados

Para cancelar um trabalho de carregamento de dados, chame o procedimento armazenado `rds_aurora.limitless_data_load_cancel`, com o ID do trabalho como parâmetro de entrada. Você chama esse procedimento armazenado do mesmo banco de dados no grupo de fragmentos em que o trabalho específico de carregamento de dados foi iniciado. Por exemplo:

```
CALL rds_aurora.limitless_data_load_cancel(12345);

INFO: limitless data load job with id 12345 is canceling without rollback.
```

Não é possível cancelar um trabalho de carregamento de dados que não existe ou não está sendo executado no mesmo grupo de fragmentos de banco de dados.

O utilitário de carregamento de dados do Aurora PostgreSQL Limitless Database deixa os dados carregados nas tabelas de destino sem reversão, como mostra a resposta. Se você não quiser manter os dados carregados, recomendamos truncar as tabelas de destino.

Consulta do Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database é compatível com a sintaxe do PostgreSQL para consultas. Você pode consultar o Limitless Database usando `psql` ou qualquer outro utilitário de conexão que funcione com o PostgreSQL. Para executar consultas, você se conecta ao endpoint sem limites, conforme mostrado em Conexão ao cluster de banco de dados do Aurora PostgreSQL Limitless Database.

Todas as consultas `SELECT` do PostgreSQL são compatíveis com o Aurora PostgreSQL Limitless Database. No entanto, as consultas são realizadas em duas camadas:

1. Roteador para o qual o cliente envia a consulta
2. Fragmentos onde os dados estão localizados

O desempenho depende da consulta ao banco de dados de uma forma que permita alcançar um alto grau de processamento simultâneo de consultas diferentes em fragmentos diferentes. As consultas são analisadas primeiro na camada de transação distribuída (roteador). Antes de planejar a execução da consulta, há uma fase de análise para identificar a localização de todas as relações que participam da consulta. Se todas as relações forem tabelas fragmentadas com uma chave de fragmento filtrada no mesmo fragmento ou tabelas de referência, o planejamento da consulta será ignorado na camada do roteador e completamente transferido para o fragmento para planejamento e execução. Esse processo reduz o número de viagens de ida e volta entre diferentes nós (roteador e fragmento) e resulta em melhor desempenho na maioria dos casos. Para ter mais informações, consulte Consultas de fragmento único no Aurora PostgreSQL Limitless Database.

> **Note**
>
> Pode haver casos específicos, como um produto cartesiano (junção cruzada), em que a consulta tem um desempenho melhor ao recuperar dados separadamente do fragmento.

Para obter mais informações, consulte EXPLAIN na Referência do Aurora PostgreSQL Limitless Database. Para obter informações gerais sobre consultas, consulte Consultas na documentação do PostgreSQL.

Tópicos

- Consultas de fragmento único no Aurora PostgreSQL Limitless Database

- Consultas distribuídas no Aurora PostgreSQL Limitless Database
- Rastreamento de consultas distribuído em logs do PostgreSQL no Aurora PostgreSQL Limitless Database
- Deadlocks distribuídos no Aurora PostgreSQL Limitless Database

Consultas de fragmento único no Aurora PostgreSQL Limitless Database

Uma consulta de fragmento único é uma consulta que pode ser executada diretamente em um fragmento, mantendo a semântica ACID do SQL. Quando essa consulta é encontrada pelo planejador de consultas no roteador, o planejador a detecta e procede enviando toda a consulta SQL para o fragmento correspondente.

Essa otimização reduz o número de viagens de ida e volta na rede do roteador para o fragmento, melhorando o desempenho. Atualmente, essa otimização é executada para consultas `INSERT`, `SELECT`, `UPDATE` e `DELETE`.

Tópicos

- Exemplos de consulta de fragmento único
- Restrições para consultas de fragmento único
- Junções totalmente qualificadas (explícitas)
- Configurar uma chave de fragmento ativa

Exemplos de consulta de fragmento único

Nos exemplos a seguir, temos a tabela fragmentada `customers`, com a chave de fragmento `customer_id`, e a tabela de referência `zipcodes`.

SELECT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM customers WHERE
 customer_id = 100;

                       QUERY PLAN
-------------------------------------------------------
 Foreign Scan
   Output: customer_id, other_id, customer_name, balance
   Remote SQL:  SELECT customer_id,
     other_id,
```

```
     customer_name,
     balance
    FROM public.customers
   WHERE (customer_id = 100)
 Single Shard Optimized
(9 rows)
```

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM orders
    LEFT JOIN zipcodes ON orders.zipcode_id = zipcodes.zipcode_id
    WHERE customer_id = 11;

                                              QUERY PLAN

-----------------------------------------------------------------------------------------------
 Foreign Scan
   Output: customer_id, order_id, zipcode_id, customer_name, balance,
 zipcodes.zipcode_id, zipcodes.city
   Remote SQL:  SELECT orders.customer_id,
     orders.order_id,
     orders.zipcode_id,
     orders.customer_name,
     orders.balance,
     zipcodes.zipcode_id,
     zipcodes.city
    FROM (public.orders
      LEFT JOIN public.zipcodes ON ((orders.zipcode_id = zipcodes.zipcode_id)))
   WHERE (orders.customer_id = 11)
 Single Shard Optimized
(13 rows)
```

INSERT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) INSERT INTO customers
    (customer_id, other_id, customer_name, balance)
    VALUES (1, 10, 'saikiran', 1000);

                      QUERY PLAN
------------------------------------------------------
 Insert on public.customers
   ->  Result
         Output: 1, 10, 'saikiran'::text, '1000'::real
 Single Shard Optimized
```

```
(4 rows)
```

UPDATE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) UPDATE orders SET balance = balance +
 100
    WHERE customer_id = 100;

                                        QUERY PLAN

-----------------------------------------------------------------------------------------------
 Update on public.orders
   Foreign Update on public.orders_fs00002 orders_1
   ->  Foreign Update
         Remote SQL:  UPDATE public.orders SET balance = (balance + (100)::double
 precision)
   WHERE (customer_id = 100)
 Single Shard Optimized
(6 rows)
```

DELETE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) DELETE FROM orders
    WHERE customer_id = 100 and balance = 0;

                             QUERY PLAN
---------------------------------------------------------------------
 Delete on public.orders
   Foreign Delete on public.orders_fs00002 orders_1
   ->  Foreign Delete
         Remote SQL:  DELETE FROM public.orders
   WHERE ((customer_id = 100) AND (balance = (0)::double precision))
 Single Shard Optimized
(6 rows)
```

Restrições para consultas de fragmento único

As consultas de fragmento único têm as seguintes restrições:

Funções

Se uma consulta de fragmento único contiver uma função, a consulta se qualificará para otimização de fragmento único somente se uma das seguintes condições se aplicar:

- A função é imutável. Para obter mais informações, consulte Volatilidade da função.
- A função é mutável, mas está registrada na visualização `rds_aurora.limitless_distributed_functions`. Para obter mais informações, consulte Distribuição de funções.

Visões

Se uma consulta contiver uma ou mais visualizações, a otimização de fragmento único será desabilitada para a consulta se ela tiver uma das seguintes condições:

- Toda visualização tem o atributo `security_barrier`.
- Os objetos usados na consulta exigem vários privilégios de usuário. Por exemplo, uma consulta contém duas visualizações, e as visualizações são executadas sob dois usuários diferentes.

```
CREATE VIEW v1 AS SELECT customer_name FROM customers c WHERE c.customer_id =  1;
CREATE VIEW v2 WITH (security_barrier) AS SELECT customer_name FROM customers c
 WHERE c.customer_id =  1;

postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v1;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Output: customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 c  (cost=0.00..24.12 rows=6 width=32)
           Output: c.customer_name
           Filter: (c.customer_id = 1)
         Query Identifier: -6005737533846718506
   Remote SQL:  SELECT customer_name
    FROM ( SELECT c.customer_name
             FROM public.customers c
            WHERE (c.customer_id = 1)) v1
 Query Identifier: -5754424854414896228
(12 rows)


postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v2;
                                        QUERY PLAN
```

```
-------------------------------------------------------------------------------------------------
 Foreign Scan on public.customers_fs00001 c  (cost=100.00..128.41 rows=7 width=32)
   Output: c.customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 customers  (cost=0.00..24.12 rows=6
 width=32)
           Output: customers.customer_name
           Filter: (customers.customer_id = 1)
         Query Identifier: 4136563775490008117
   Remote SQL: SELECT customer_name FROM public.customers WHERE ((customer_id = 1))
 Query Identifier: 5056054318010163757
(9 rows)
```

Instruções PREPARE e EXECUTE

O Aurora PostgreSQL Limitless Database comporta a otimização de fragmento único para declarações `SELECT`, `UPDATE` e `DELETE` preparadas.

No entanto, se você usar declarações preparadas para `PREPARE` e `EXECUTE` com `plan_cache_mode` definido como `'force_generic_plan'`, o planejador de consultas rejeitará a otimização de fragmento único para essa consulta.

PL/pgSQL

As consultas com variáveis PL/pgSQL são executadas como instruções preparadas implicitamente. Se uma consulta contiver qualquer variável PL/pgSQL, o planejador de consultas rejeitará a otimização de fragmento único.

A otimização é compatível no bloco PL/pgSQL se a instrução não contiver nenhuma variável PL/pgSQL.

Junções totalmente qualificadas (explícitas)

A otimização de fragmento único é baseada em eliminações de partições. O otimizador do PostgreSQL elimina partições com base em condições constantes. Se o Aurora PostgreSQL Limitless Database descobrir que todas as partições e tabelas restantes estão no mesmo fragmento, ele marca a consulta como elegível para otimização de fragmento único. Todas as condições de filtro devem ser explícitas para que a eliminação de partições funcione. O Aurora PostgreSQL Limitless Database não pode eliminar partições sem um ou mais predicados de junção ou predicados de filtro nas chaves de fragmento de cada tabela fragmentada na instrução.

Suponha que tenhamos particionado as tabelas `customers`, `orders` e `order_details` com base na coluna `customer_id`. Nesse esquema, a aplicação tenta manter todos os dados de um cliente em um único fragmento.

Considere a seguinte consulta:

```
SELECT * FROM
    customers c, orders o, order_details od
WHERE c.customer_id = o.customer_id
    AND od.order_id = o.order_id
    AND c.customer_id = 1;
```

Essa consulta recupera todos os dados de um cliente (`c.customer_id = 1`). Os dados desse cliente estão em um único fragmento, mas o Aurora PostgreSQL Limitless Database não qualifica essa consulta como uma consulta de fragmento único. O processo de otimização da consulta é o seguinte:

1. O otimizador pode eliminar partições para `customers` e `orders` com base na seguinte condição:

   ```
   c.customer_id = 1
   c.customer_id = o.customer_id
   o.customer_id =  1 (transitive implicit condition)
   ```

2. O otimizador não consegue eliminar nenhuma partição para `order_details`, pois não há nenhuma condição constante na tabela.
3. O otimizador conclui que leu todas as partições de `order_details`. Portanto, a consulta não pode ser qualificada para otimização de fragmento único.

Para fazer disso uma consulta de fragmento único, adicionamos a seguinte condição de junção explícita:

```
o.customer_id = od.customer_id
```

A consulta alterada fica assim:

```
SELECT * FROM
    customers c, orders o,  order_details od
WHERE c.customer_id = o.customer_id
     AND o.customer_id = od.customer_id
```

```
    AND od. order_id = o. order_id
AND c.customer_id =  1;
```

Agora, o otimizador pode eliminar partições para `order_details`. A nova consulta se torna uma consulta de fragmento único e se qualifica para otimização.

Configurar uma chave de fragmento ativa

Esse recurso permite que você defina uma única chave de fragmento ao consultar o banco de dados, fazendo com que todas as consultas `SELECT` e DML sejam anexadas à chave de fragmento como um predicado constante. Esse recurso é útil se você migrou para o Aurora PostgreSQL Limitless Database e desnormalizou o esquema adicionando chaves de fragmento às tabelas.

É possível anexar um predicado de chave de fragmento automaticamente à lógica SQL existente, sem alterar a semântica das consultas. A anexação de um predicado de chave de fragmento ativa é feita somente para tabelas compatíveis.

O recurso de chave de fragmento ativa usa a variável `rds_aurora.limitless_active_shard_key`, que tem a seguinte sintaxe:

```
SET [session | local] rds_aurora.limitless_active_shard_key = '{"col1_value",
 "col2_value", ...}';
```

Algumas considerações sobre chaves de fragmento ativas e chaves estrangeiras:

- Uma tabela fragmentada pode ter uma restrição de chave externa se as tabelas pai e filha tiverem localização compartilhada e a chave externa for um superconjunto da chave de fragmento.
- Uma tabela fragmentada pode ter uma restrição de chave externa para uma tabela de referência.
- Uma tabela de referência pode ter uma restrição de chave externa para outra tabela de referência.

Suponha que temos uma tabela `customers` que é fragmentada na coluna `customer_id`.

```
BEGIN;
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE customers(customer_id int PRIMARY KEY, name text , email text);
COMMIT;
```

Com um conjunto de chaves de fragmento ativo, as consultas têm as transformações a seguir.

SELECT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
SELECT * FROM customers;

-- This statement is changed to:
SELECT * FROM customers WHERE customer_id = '123'::int;
```

INSERT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
INSERT INTO customers(name, email) VALUES('Alex', 'alex@example.com');

-- This statement is changed to:
INSERT INTO customers(customer_id, name, email) VALUES('123'::int, 'Alex',
 'alex@example.com');
```

UPDATE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
UPDATE customers SET email = 'alex_new_email@example.com';

-- This statement is changed to:
UPDATE customers SET email = 'alex_new_email@example.com' WHERE customer_id =
 '123'::int;
```

DELETE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
DELETE FROM customers;

-- This statement is changed to:
DELETE FROM customers WHERE customer_id = '123'::int;
```

Junções

Ao realizar operações de junção em tabelas com uma chave de fragmento ativa, o predicado da chave de fragmento é automaticamente adicionado a todas as tabelas envolvidas na junção. Essa adição automática do predicado da chave de fragmento ocorre somente quando todas as tabelas na consulta pertencem ao mesmo grupo de colocalização. Se a consulta envolver tabelas de diferentes grupos de colocalização, um erro será gerado em vez disso.

Suponha que também temos tabelas `orders` e `order_details` que estão colocalizadas com a tabela `customers`.

```
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_collocate_with='customers';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE orders (id int , customer_id int, total_amount int, date date);
CREATE TABLE order_details (id int , order_id int, customer_id int, product_name
 VARCHAR(100), price int);
COMMIT;
```

Recupere as últimas 10 faturas de pedidos de um cliente cujo ID de cliente é 10.

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    order_details.order_id = orders.order_id AND
    customers.customer_id = 10
    order by order_date limit 10;
```

Essa consulta é transformada na seguinte:

```
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    orders.order_id = order_details.order_id AND
    customers.customer_id = 10 AND
    order_details.customer_id = 10 AND
    orders.customer_id = 10 AND
    ORDER BY "order_date" LIMIT 10;
```

Tabelas compatíveis com chave de fragmento ativa

O predicado da chave de fragmento é adicionado somente às tabelas compatíveis com a chave de fragmento ativa. Uma tabela é considerada compatível se tiver o mesmo número de colunas em sua chave de fragmento, conforme especificado na variável `rds_aurora.limitless_active_shard_key`. Se a consulta envolver tabelas incompatíveis com a chave de fragmento ativa, o sistema gerará um erro em vez de prosseguir com a consulta.

Por exemplo:

```
-- Compatible table
```

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';

-- The following query works because the customers table is sharded on one column.
SELECT * FROM customers;

-- Incompatible table
SET rds_aurora.limitless_active_shard_key = '{"10","20"}';

-- The following query raises a error because the customers table isn't sharded on
 two columns.
 SELECT * FROM customers;
```

Consultas distribuídas no Aurora PostgreSQL Limitless Database

As consultas distribuídas são executadas em um roteador e em mais de um fragmento. A consulta é recebida por um dos roteadores. O roteador cria e gerencia a transação distribuída, que é enviada aos fragmentos participantes. Os fragmentos criam uma transação local com o contexto fornecido pelo roteador e a consulta é executada.

Quando a transação é confirmada, o roteador usa um protocolo de confirmação em duas fases otimizado, se necessário, e um controle de simultaneidade de várias versões (MVCC) baseado em tempo para fornecer semântica ACID em um sistema de banco de dados distribuído.

O MVCC baseado em tempo registra o horário de confirmação de cada transação e usa o horário de início da transação para gerar o horário do snapshot dos dados. Para identificar se uma transação está confirmada (visível) dado o snapshot do leitor, o banco de dados compara o horário de confirmação com o horário do snapshot. Se o horário de confirmação for menor que o horário do snapshot do leitor, a transação estará visível; caso contrário, invisível. Sob esse protocolo, você sempre verá dados fortemente consistentes no Aurora PostgreSQL Limitless Database.

Rastreamento de consultas distribuído em logs do PostgreSQL no Aurora PostgreSQL Limitless Database

O rastreamento de consultas distribuído é uma ferramenta para rastrear e correlacionar consultas em logs do PostgreSQL no Aurora PostgreSQL Limitless Database. No Aurora PostgreSQL, é possível usar o ID da transação para identificar uma transação. Mas no Aurora PostgreSQL Limitless Database, o ID da transação pode se repetir em vários roteadores. Portanto, recomendamos usar o ID de rastreamento no Limitless Database.

Os principais casos de uso são:

- Usar a função `rds_aurora.limitless_get_last_trace_id()` para encontrar o ID de rastreamento exclusivo da última consulta executada na sessão atual. Em seguida, pesquisar o grupo de logs do cluster de banco de dados no Amazon CloudWatch Logs usando esse ID de rastreamento para encontrar todos os logs relacionados.

  Você pode usar os parâmetros `log_min_messages` e `log_min_error_statement` em conjunto para controlar o volume de logs impressos e imprimir uma declaração que contenha o ID de rastreamento.
- Use o parâmetro `log_min_duration_statement` para determinar o tempo de execução acima do qual todas as consultas imprimem sua duração de execução e ID de rastreamento. Esse tempo de execução pode então ser pesquisado no grupo de logs do cluster de banco de dados no CloudWatch Logs para ajudar a determinar os nós de gargalo e os esforços de otimização do planejador.

  O parâmetro `log_min_duration_statement` habilita o ID de rastreamento para todos os nós, independentemente dos valores dos parâmetros `log_min_messages` e `log_min_error_statement`.

Tópicos

- ID de rastreamento
- Usar o rastreamento de consultas
- Exemplos de log

ID de rastreamento

O ponto central deste recurso é um identificador exclusivo conhecido como ID de rastreamento. O ID de rastreamento é uma string de 31 dígitos anexada às linhas de log STATEMENT dos logs do PostgreSQL, que atua como um identificador preciso para correlacionar logs relacionados à execução de uma consulta específica. Os exemplos são `1126253375719408504000000000011` e `1126253375719408495000000000090`.

O ID de rastreamento é composto pelo seguinte:

- Identificador da transação: os primeiros 20 dígitos, que identificam a transação de forma exclusiva.
- Identificador de comando: os primeiros 30 dígitos, que indicam uma consulta individual em uma transação.

  Se mais de 4.294.967.294 consultas forem executadas dentro de um bloco de transação explícito, o identificador de comando retorna para 1. Quando isso acontece, você recebe uma notificação pela seguinte mensagem de `LOG` no log do PostgreSQL:

  ```
  wrapping around the tracing ID back to 1 after running 4294967294 (4.2 billion or
   2^32-2) queries inside of an explicit transaction block
  ```

- Identificador do tipo de nó: o último dígito, que indica se o nó está funcionando como um roteador coordenador (`1`) ou um nó participante (`0`).

Os seguintes exemplos ilustram os componentes do ID de rastreamento:

- `1126253375719408504000000000011`:
  - Identificador da transação: `1126253375719408504`
  - Identificador de comando: `1126253375719408504000000000001` 1 indica o primeiro comando no bloco de transação
  - Identificador do tipo de nó: 1 indica um roteador coordenador
- `1126253375719408495000000000090`:
  - Identificador da transação: `1126253375719408495`
  - Identificador de comando: `1126253375719408495000000000009` 9 indica o nono comando no bloco de transação
  - Identificador do tipo de nó: `0` indica um nó participante

Usar o rastreamento de consultas

Execute as seguintes tarefas para usar o rastreamento de consultas:

1. Verifique se o rastreamento está habilitado.

   Você pode verificar o seguinte comando:

   ```
   SHOW rds_aurora.limitless_log_distributed_trace_id;
   ```

   Por padrão, ele é habilitado (on). Se não estiver habilitado, configure-o usando este comando:

   ```
   SET rds_aurora.limitless_log_distributed_trace_id = on;
   ```

2. Controle o volume dos logs impressos configurando o nível de severidade do log.

   O volume de log é controlado pelo parâmetro `log_min_messages`. O parâmetro `log_min_error_statement` é usado para imprimir a linha `STATEMENT` com o ID de rastreamento. Por padrão, ambos estão definidos como `ERROR`. Você pode verificar usando os seguintes comandos:

   ```
   SHOW log_min_messages;
   SHOW log_min_error_statement;
   ```

   Para atualizar o nível de gravidade e imprimir a linha `STATEMENT` da sessão atual, use os seguintes comandos com um desses níveis de gravidade:

   ```
   SET log_min_messages = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO | NOTICE
    | WARNING | ERROR | LOG | FATAL | PANIC';
   SET log_min_error_statement = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO
    | NOTICE | WARNING | ERROR | LOG | FATAL | PANIC';
   ```

   Por exemplo:

   ```
   SET log_min_messages = 'WARNING';
   SET log_min_error_statement = 'WARNING';
   ```

3. Habilite a impressão do ID de rastreamento nos logs acima de um tempo de execução específico.

O parâmetro `log_min_duration_statement` pode ser alterado para o tempo mínimo de execução da consulta acima do qual a consulta imprime uma linha de log com a duração da execução, junto com os IDs de rastreamento em todo o cluster de banco de dados. Por padrão, esse parâmetro é definido como `-1`, o que significa que está desabilitado. Você pode verificar o seguinte comando:

```
SHOW log_min_duration_statement;
```

Alterá-lo para `0` imprime a duração e o ID de rastreamento nos logs de cada consulta no cluster de banco de dados. É possível defini-lo como `0` para a sessão atual usando o seguinte comando:

```
SET log_min_duration_statement = 0;
```

4. Obtenha o ID de rastreamento.

   Depois de executar uma consulta (mesmo dentro de um bloco de transação explícito), chame a função `rds_aurora.limitless_get_last_trace_id` para obter o ID de rastreamento da última consulta executada:

   ```
   SELECT * FROM rds_aurora.limitless_get_last_trace_id();
   ```

   Essa função retorna o identificador da transação e o identificador do comando. Ela não retorna o identificador do tipo de nó.

   ```
   => SELECT * FROM customers;
    customer_id | fname | lname
   -------------+-------+-------
   (0 rows)

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier |       command_identifier
   ------------------------+--------------------------------
    10104661421959001813   | 1010466142195900181300000000001
   (1 row)
   ```

   A função retorna uma linha em branco para consultas não distribuídas, pois elas não têm um ID de rastreamento.

```
=> SET search_path = public;
SET

=> SELECT * FROM rds_aurora.limitless_get_last_trace_id();
 transaction_identifier | command_identifier
------------------------+--------------------
                        |
(1 row)
```

> **Note**
>
> Para consultas VACUUM e ANALYZE, a instrução de duração não é registrada com o ID de rastreamento. Portanto, `limitless_get_last_trace_id()` não retorna o ID de rastreamento. Se VACUUM ou ANALYZE for uma operação de longa duração, você pode usar a seguinte consulta para obter o ID de rastreamento dessa operação:
>
> ```
> SELECT * FROM rds_aurora.limitless_stat_activity
> WHERE distributed_tracing_id IS NOT NULL;
> ```
>
> Se o servidor for interrompido antes de você encontrar o último ID de rastreamento, será necessário pesquisar os logs do PostgreSQL manualmente para encontrar os IDs de rastreamento nos logs logo antes da falha.

5. Pesquise o ID de rastreamento nos logs do cluster de banco de dados usando o CloudWatch.

   Use o CloudWatch Insights para consultar o grupo de logs do cluster de banco de dados, conforme mostrado nos exemplos a seguir.

   - Consultar um identificador de transação específico e todos os comandos executados dentro dele:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813/
     | sort @timestamp desc
     ```

   - Consultar um identificador de comando específico:

     ```
     fields @timestamp, @message
     | filter @message like /1010466142195900181300000000001/
     ```

```
| sort @timestamp desc
```

6. Examine todos os logs em todo o cluster de banco de dados produzidos pela consulta distribuída.

Exemplos de log

Os exemplos a seguir mostram o uso do rastreamento de consultas.

Correlacionar logs para consultas propensas a erros

Neste exemplo, o comando `TRUNCATE` é executado na tabela `customers` quando essa tabela não existe.

Sem rastreamento de consultas

Arquivo de log do PostgreSQL no roteador coordenador:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:CONTEXT: remote
 SQL command: truncate public.customers;
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:STATEMENT:
 truncate customers;
```

Arquivo de log do PostgreSQL em um fragmento participante:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:STATEMENT:
 truncate customers;
```

Esses logs são comuns. Eles não têm os identificadores precisos necessários para correlacionar facilmente as consultas em todo o cluster de banco de dados.

Com rastreamento de consultas

Arquivo de log do PostgreSQL no roteador coordenador:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:CONTEXT: remote SQL
 command: truncate public.customers;
```

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT: /* tid =
 11262533757194085027000000000011 */ truncate customers;
```

Arquivo de log do PostgreSQL em um fragmento participante:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR:  failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT:  /* tid
 = 11262533757194085027000000000010 */ truncate customers;
```

Na presença do rastreamento de consulta, cada linha de log é anexada com um identificador único de 31 dígitos. Aqui, `11262533757194085027000000000011` e `11262533757194085027000000000010` representam os IDs de rastreamento para os nós coordenador e participante, respectivamente.

- Identificador da transação: `11262533757194085027`
- Identificador de comando: `1126253375719408502700000000001`
- Identificador do tipo de nó: o último dígito, `1` ou `0`, indica um roteador coordenador e um nó participante, respectivamente.

Correlacionar logs para encontrar o tempo de execução da consulta em vários nós

Neste exemplo, o parâmetro `log_min_duration_statement` foi atualizado para `0` para imprimir a duração de todas as consultas.

Sem rastreamento de consultas

```
2024-01-15 07:28:46 UTC:[local]:postgres@postgres_limitless:[178322]:LOG: duration:
 12.779 ms statement: select * from customers;
```

Com rastreamento de consultas

Arquivo de log do PostgreSQL no roteador coordenador:

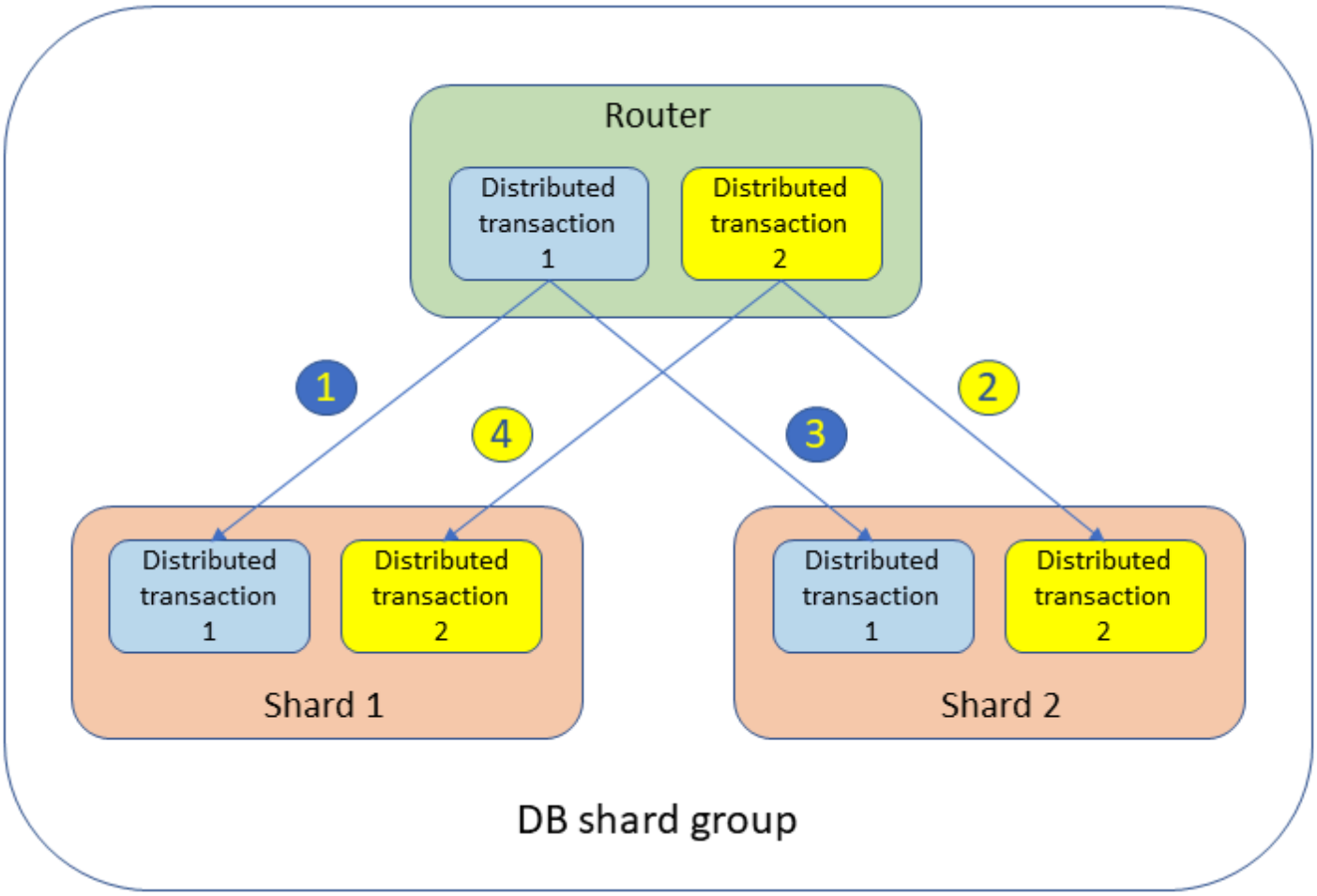
```
2024-01-15 07:32:08 UTC:[local]:postgres@postgres_limitless:[183877]:LOG: duration:
 12.618 ms statement: /* tid = 045766956624049708840000000011 */ select * from
 customers;
```

Arquivo de log do PostgreSQL em um fragmento participante:

```
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.279 ms statement: /* tid = 045766956624049708840000000010 */ START
 TRANSACTION ISOLATION LEVEL READ COMMITTED
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.249 ms parse <unnamed>: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.398 ms bind <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.019 ms execute <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.073 ms statement: /* tid = 045766956624049708840000000010 */ COMMIT
 TRANSACTION
```

Deadlocks distribuídos no Aurora PostgreSQL Limitless Database

Em um grupo de fragmentos de banco de dados, podem ocorrer deadlocks entre transações distribuídas entre diferentes roteadores e fragmentos. Por exemplo, duas transações distribuídas simultâneas que abrangem dois fragmentos são executadas, conforme apresentado na figura a seguir.



As transações bloqueiam tabelas e criam eventos de espera nos dois fragmentos da seguinte forma:

1. Transação distribuída 1:

   ```
   UPDATE table SET value = 1 WHERE key = 'shard1_key';
   ```

   Isso mantém um bloqueio no fragmento 1.

2. Transação distribuída 2:

```
UPDATE table SET value = 2 WHERE key = 'shard2_key';
```

Isso mantém um bloqueio no fragmento 2.

3. Transação distribuída 1:

```
UPDATE table SET value = 3 WHERE key = 'shard2_key';
```

A transação distribuída 1 está aguardando o fragmento 2.

4. Transação distribuída 2:

```
UPDATE table SET value = 4 WHERE key = 'shard1_key';
```

A transação distribuída 2 está aguardando o fragmento 1.

Nesse cenário, nem o fragmento 1 nem o fragmento 2 veem o problema: a transação 1 está aguardando a transação 2 no fragmento 2 e a transação 2 está aguardando a transação 1 no fragmento 1. De uma visão global, a transação 1 está aguardando a transação 2 e a transação 2 está aguardando a transação 1. Essa situação em que duas transações em dois fragmentos diferentes estão esperando uma pela outra é chamada de deadlock distribuído.

O Aurora PostgreSQL Limitless Database pode detectar e resolver deadlocks distribuídos automaticamente. Um roteador no grupo de fragmentos de banco de dados é notificado quando uma transação está esperando muito tempo para adquirir um recurso. O roteador que recebe a notificação começa a coletar as informações necessárias de todos os roteadores e fragmentos dentro do grupo de fragmentos de banco de dados. O roteador então finaliza as transações que estão participando de um deadlock distribuído, até que as demais transações no grupo de fragmentos de banco de dados possam prosseguir sem serem bloqueadas umas pelas outras.

Você recebe o seguinte erro quando sua transação faz parte de um deadlock distribuído e é encerrada pelo roteador:

```
ERROR: aborting transaction participating in a distributed deadlock
```

O parâmetro do cluster de banco de dados `rds_aurora.limitless_distributed_deadlock_timeout` define o tempo de espera de cada transação em um recurso antes de notificar o roteador para verificar se há um deadlock distribuído. É

possível aumentar o valor do parâmetro se a workload for menos propensa a situações de deadlock. O padrão é `1000` milissegundos (1 segundo).

O ciclo de deadlock distribuído é publicado nos logs do PostgreSQL quando um deadlock entre nós é encontrado e resolvido. As informações sobre cada processo que faz parte do deadlock incluem o seguinte:

- Nó coordenador que iniciou a transação
- ID de transação virtual (xid) da transação no nó coordenador, no formato *`backend_id`*`/`*`backend_local_xid`*
- ID de sessão distribuída da transação

Gerenciamento do Aurora PostgreSQL Limitless Database

Os tópicos a seguir descrevem como gerenciar clusters de banco de dados do Aurora PostgreSQL Limitless Database.

Tópicos

- Considerações sobre o tamanho do banco de dados e das tabelas
- Recuperar espaço de armazenamento por meio de limpeza

Considerações sobre o tamanho do banco de dados e das tabelas

Para o Aurora PostgreSQL Limitless Database, em cada fragmento, uma tabela fragmentada é dividida em várias fatias que variam dependendo de quantos fragmentos estão disponíveis no grupo de fragmentos do banco de dados. Cada fatia de tabela pode crescer até 32 TiB, mas cada fragmento tem uma capacidade máxima de 128 TiB. As tabelas de referência têm um limite de tamanho de 32 TiB para todo o grupo de fragmentos do banco de dados.

> **Note**
>
> A capacidade máxima de cada nó (roteador ou fragmento) é 128 TiB, pois essa é a capacidade máxima de um cluster de banco de dados Aurora PostgreSQL.

O número máximo de relações por banco de dados (incluindo tabelas, visualizações e índices) no Aurora PostgreSQL e no Aurora PostgreSQL Limitless Database é 1.431.650.303.

Para obter mais informações, consulte Apêndice K. Limites do PostgreSQL na documentação do PostgreSQL e Limites de tamanho do Amazon Aurora.

Recuperar espaço de armazenamento por meio de limpeza

O Controle de Simultaneidade de Várias Versões (MVCC) do PostgreSQL ajuda a preservar a integridade dos dados salvando uma cópia interna das linhas atualizadas ou excluídas até que uma transação seja confirmada ou revertida. Essas cópias, também chamadas de tuplas, podem causar sobrecarga na tabela se não forem limpas regularmente. As instâncias do PostgreSQL ordenam as transações por seus IDs de transação, e o PostgreSQL usa o MVCC baseado em ID de transação para controlar a visibilidade da tupla e fornecer isolamento de transações. Cada transação estabelece um snapshot dos dados e cada tupla tem uma versão. Tanto o snapshot quanto a versão são baseados no ID da transação.

Para limpar os dados, o utilitário `VACUUM` executa quatro funções principais no PostgreSQL:

- `VACUUM`: remove versões de linha expiradas, disponibilizando o espaço para reutilização.
- `VACUUM FULL`: fornece desfragmentação completa removendo versões de linha inativas e compactando as tabelas, reduzindo o tamanho e aumentando a eficiência.
- `VACUUM FREEZE`: protege contra problemas de conclusão do ID da transação marcando as versões mais antigas da linha como congeladas.
- `VACUUM ANALYZE`: remove versões de linhas inativas e atualiza as estatísticas de planejamento de consultas do banco de dados. É uma combinação das funções `VACUUM` e `ANALYZE`. Para obter mais informações sobre como `ANALYZE` funciona no Aurora PostgreSQL Limitless Database, consulte ANALYZE.

Assim como no MVCC, a limpeza no Aurora PostgreSQL é baseada no ID da transação. Se houver uma transação em andamento no início da limpeza, as linhas que ainda estiverem visíveis para essa transação não serão removidas.

Para obter mais informações sobre o utilitário `VACUUM`, consulte VACUUM na documentação do PostgreSQL. Para obter mais informações sobre o suporte a `VACUUM` no Aurora PostgreSQL Limitless Database, consulte VACUUM.

Tópicos

- AUTOVACUUM
- Limpeza baseada em tempo no Aurora PostgreSQL Limitless Database
- Usar estatísticas de banco de dados para limpeza

- Diferenças no comportamento de limpeza entre o Aurora PostgreSQL e o Aurora PostgreSQL Limitless Database

AUTOVACUUM

O Aurora PostgreSQL usa os utilitários `VACUUM` e `AUTOVACUUM` para remover tuplas desnecessárias. O mecanismo subjacente de `AUTOVACUUM` e `VACUUM` manual é o mesmo; a única diferença é a automação.

`AUTOVACUUM` no Aurora PostgreSQL e no Aurora PostgreSQL Limitless Database é uma combinação dos utilitários `VACUUM` e `ANALYZE`. `AUTOVACUUM` determina quais bancos de dados e tabelas limpar, de acordo com uma regra predefinida, como a porcentagem de tuplas inativas e o número de inserções.

Por exemplo, `AUTOVACUUM` "acorda" periodicamente para realizar a limpeza. O intervalo é controlado pelo parâmetro `autovacuum_naptime`. O valor padrão é 1 minuto. Os valores padrão dos parâmetros de configuração `AUTOVACUUM` e `VACUUM` são os mesmos do Aurora PostgreSQL Limitless Database e do Aurora PostgreSQL.

O daemon `AUTOVACUUM`, se habilitado, emite comandos `ANALYZE` automaticamente sempre que o conteúdo de uma tabela for suficientemente alterado. No Aurora PostgreSQL Limitless Database, `AUTOVACUUM` emite `ANALYZE` em roteadores e fragmentos.

Para obter mais informações sobre o daemon `AUTOVACUUM` e os parâmetros de armazenamento de tabela associados a `AUTOVACUUM`, consulte The autovacuum daemon e Storage parameters na documentação do PostgreSQL.

Limpeza baseada em tempo no Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database é um sistema distribuído, o que significa que várias instâncias podem estar envolvidas em uma transação. Portanto, a visibilidade baseada em ID de transação não se aplica. Em vez disso, o Aurora PostgreSQL Limitless Database usa visibilidade baseada em tempo, porque os IDs de transação não são "unificados" entre instâncias, mas o tempo pode ser "unificado" entre instâncias. Cada snapshot de transação e cada versão de tupla obedece ao tempo em vez do ID da transação. Mais especificamente, um snapshot de transação tem um horário de início de snapshot e uma tupla tem um horário de criação (quando `INSERT` ou `UPDATE` acontece) e um horário de exclusão (quando `DELETE` acontece).

Para manter a consistência de dados nas instâncias do grupo de fragmentos de banco de dados, o Aurora PostgreSQL Limitless Database precisa garantir que a limpeza não remova nenhuma tupla que ainda esteja visível para qualquer transação ativa no grupo de fragmentos de banco de dados. Portanto, a limpeza no Aurora PostgreSQL Limitless Database também é baseada em tempo. Outros aspectos do `VACUUM` permanecem os mesmos, incluindo que, para executar `VACUUM` em uma tabela específica, o usuário deve ter acesso a essa tabela.

> **Note**
>
> É altamente recomendável que você não deixe transações abertas por longos períodos. A limpeza baseada em tempo consome mais memória do que a limpeza baseada em ID de transação.

O exemplo a seguir ilustra como funciona a limpeza baseada em tempo.

1. Uma tabela de clientes é distribuída em quatro fragmentos.
2. A transação 1 começa com uma leitura repetível e tem como alvo apenas um fragmento (fragmento 1). Essa transação permanece aberta.

   A transação 1 é mais antiga do que as transações iniciadas depois dela.
3. A transação 2 começa mais tarde, exclui todas as tuplas de uma tabela e, em seguida, é confirmada.
4. Se `AUTOVACUUM` ou `VACUUM` manual tentar limpar tuplas inativas (inativas devido à transação 2), ele não removerá nada.

   Isso vale não apenas para o fragmento 1, mas também para os fragmentos 2–4, porque a transação 1 ainda pode precisar acessar essas tuplas. Elas ainda estão visíveis para a transação 1 por causa do MVCC.

A última etapa é realizada por meio da sincronização, para que todos os fragmentos estejam cientes da transação 1, mesmo que a transação 1 não afete todos eles.

Usar estatísticas de banco de dados para limpeza

Para obter informações sobre tuplas que talvez você precise limpar, use a visualização limitless_stat_all_tables, que funciona de forma semelhante a pg_stat_all_tables. O exemplo a seguir consulta a visualização.

```
SELECT * FROM rds_aurora.limitless_stat_all_tables WHERE relname LIKE '%customer%';
```

Da mesma forma, para estatísticas de banco de dados, use limitless_stat_database em vez de pg_stat_database, e limitless_stat_activity em vez de pg_stat_activity.

Para verificar o uso do disco da tabela, use a função limitless_stat_relation_sizes, que funciona de forma semelhante a pg_relation_size. O exemplo a seguir consulta a função.

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customer');
```

Para acompanhar o progresso de uma operação VACUUM no Aurora PostgreSQL Limitless Database, use a visualização limitless_stat_progress_vacuum em vez de pg_stat_progress_vacuum. O exemplo a seguir consulta a visualização.

```
SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;
```

Para ter mais informações, consulte Visualizações do Aurora PostgreSQL Limitless Database e Funções do Aurora PostgreSQL Limitless Database.

Diferenças no comportamento de limpeza entre o Aurora PostgreSQL e o Aurora PostgreSQL Limitless Database

Algumas outras diferenças entre o Aurora PostgreSQL e o Aurora PostgreSQL Limitless Database na forma como a limpeza funciona são as seguintes:

- O Aurora PostgreSQL executa operações VACUUM em IDs de transação até a transação mais antiga em andamento. Se não houver nenhuma transação em andamento no banco de dados, VACUUM executará a operação até a última transação.
- O Aurora PostgreSQL Limitless Database sincroniza o snapshot de tempo mais antigo a cada 10 segundos. Portanto, VACUUM talvez não realize a operação em nenhuma transação que tenha sido executada nos últimos 10 segundos.

Para obter informações sobre o suporte a VACUUM no Aurora PostgreSQL Limitless Database, consulte VACUUM em Referência do Aurora PostgreSQL Limitless Database.

Monitoramento do Aurora PostgreSQL Limitless Database

Você pode usar o Amazon CloudWatch, o Monitoramento aprimorado e o Insights de Performance para monitorar o Aurora PostgreSQL Limitless Database. Também há novas funções e visualizações de estatísticas, e eventos de espera, para o Aurora PostgreSQL Limitless Database que você pode usar para monitoramento e diagnóstico.

Tópicos

- Monitorar o Aurora PostgreSQL Limitless Database com o Amazon CloudWatch
- Monitorar o Aurora PostgreSQL Limitless Database com o CloudWatch Database Insights
- Monitoramento do Aurora PostgreSQL Limitless Database com o Amazon CloudWatch Logs
- Monitoramento do Aurora PostgreSQL Limitless Database com Monitoramento aprimorado
- Monitoramento do Aurora PostgreSQL Limitless Database com o Insights de Performance
- Monitoramento do Aurora PostgreSQL Limitless Database com o Amazon GuardDuty RDS Protection
- Funções e visualizações do Aurora PostgreSQL Limitless Database
- Eventos de espera do Aurora PostgreSQL Limitless Database
- Usar funções para aumentar a eficiência

Monitorar o Aurora PostgreSQL Limitless Database com o Amazon CloudWatch

As métricas do CloudWatch para o Aurora PostgreSQL Limitless Database são relatadas nas seguintes dimensões:

- DBShardGroup
- DBShardGroupRouterAggregation
- DBShardGroupInstance
- DBClusterIdentifier

Para obter mais informações sobre métricas do CloudWatch, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.

Métricas do DBShardGroup

Para ver métricas `DBShardGroup` do Aurora PostgreSQL Limitless Database no console do CloudWatch, escolha RDS e, em seguida, DBShardGroup.

Você pode acompanhar as seguintes métricas do CloudWatch:

- `DBShardGroupACUUtilization`: uso da unidade de capacidade do Aurora (ACU) como uma porcentagem calculada a partir de `DBShardGroupCapacity` dividido por `DBShardGroupMaxACU`.
- `DBShardGroupCapacity`: número de ACUs consumidas por instâncias do gravador do grupo de fragmentos de banco de dados.
- `DBShardGroupComputeRedundancyCapacity`: número de ACUs consumidas por instâncias em espera do gravador do grupo de fragmentos de banco de dados.
- `DBShardGroupMaxACU`: número máximo de ACUs configuradas para o grupo de fragmentos de banco de dados.
- `DBShardGroupMinACU`: número mínimo de ACUs exigido pelo grupo de fragmentos de banco de dados.

A chave de dimensão `DBShardGroupIdentifier` está disponível para agregar as métricas `DBShardGroup`.

Métricas do DBShardGroupRouterAggregation

Para ver métricas `DBShardGroupRouterAggregation` do Aurora PostgreSQL Limitless Database no console do CloudWatch, escolha RDS e, em seguida, DBShardGroupRouterAggregation.

Você pode acompanhar as seguintes métricas do CloudWatch:

- `CommitThroughput`: o número médio de operações de confirmação por segundo em todos os nós do roteador no grupo de fragmentos de banco de dados.
- `DatabaseConnections`: a soma de todas as conexões em todos os nós do roteador no grupo de fragmentos de banco de dados.

Métricas do DBShardGroupInstance

DBShardGroupInstance é a instância de banco de dados individual dentro de cada fragmento ou subcluster de roteador.

Para ver métricas `DBShardGroupInstance` do Aurora PostgreSQL Limitless Database no console do CloudWatch, escolha RDS e, em seguida, DBShardGroupInstance.

Você pode acompanhar as seguintes métricas do CloudWatch:

- `ACUUtilization`: a porcentagem calculada como a métrica `ServerlessDatabaseCapacity` dividida pelo valor máximo de ACU atribuído ao subcluster.
- `AuroraReplicaLag`: para clusters do Limitless com a redundância de computação habilitada, essa é a quantidade de atraso ao replicar atualizações da instância primária no subcluster.
- `AuroraReplicaLagMaximum`: para clusters do Limitless com redundância de computação habilitada, essa é a quantidade máxima de atraso ao replicar atualizações da instância primária no subcluster. Quando as réplicas de leitura são excluídas ou renomeadas, pode haver um aumento temporário no atraso de replicação à medida que o recurso antigo é reciclado. Use essa métrica para descobrir se ocorreu um failover devido a um alto atraso na replicação em um dos leitores.
- `AuroraReplicaLagMinimum`: para clusters do Limitless com redundância de computação habilitada, essa é a quantidade mínima de atraso ao replicar atualizações da instância primária no subcluster.
- `BufferCacheHitRatio`: a porcentagem de dados e índices fornecidos pelo cache de memória de uma instância (ao contrário do volume de armazenamento).
- `CommitLatency`: a duração média para o mecanismo e o armazenamento concluírem as operações de confirmação de um nó específico (roteador ou fragmento).

- `CommitThroughput`: o número médio de operações de confirmação por segundo.
- `CPUUtilization`: uso da CPU como porcentagem do valor máximo de ACU atribuído ao subcluster.
- `FreeableMemory`: a quantidade de memória não utilizada que está disponível quando o grupo de fragmentos é escalado para sua capacidade máxima. Isso é determinado pelas ACUs atribuídas do grupo de fragmentos. Para cada ACU em que a capacidade atual está abaixo da capacidade máxima, esse valor aumenta em aproximadamente 2 GiB. Desse modo, essa métrica não se aproxima de zero enquanto o grupo de fragmentos de banco de dados não for escalado até o limite máximo.
- `MaximumUsedTransactionIDs`: a idade do ID da transação mais antiga sem limpeza, em transações. Se esse valor atingir 2.146.483.648 (2^31 - 1.000.000), o banco de dados será forçado para o modo somente leitura a fim de evitar a conclusão do ID da transação. Para ter mais informações, consulte Evitar falhas de conclusão do ID da transação na documentação do PostgreSQL.
- `NetworkReceiveThroughput`: a quantidade de throughput de rede recebida dos clientes por cada instância no grupo de fragmentos de bancos de dados. Esse throughput não inclui o tráfego de rede entre instâncias no grupo de fragmentos de banco de dados e o volume do cluster.
- `NetworkThroughput`: o throughput da rede agregada (tanto transmitida quanto recebida) entre clientes e roteadores, e roteadores e fragmentos no grupo de fragmentos de banco de dados. Esse throughput não inclui o tráfego de rede entre instâncias no grupo de fragmentos de banco de dados e o volume do cluster.
- `NetworkTransmitThroughput`: a quantidade de throughput de rede enviada aos clientes por cada instância no grupo de fragmentos de bancos de dados. Esse throughput não inclui o tráfego de rede entre instâncias no grupo de fragmentos de banco de dados e o volume do cluster.
- `ReadIOPS`: o número médio de operações de entrada e saída por segundo (IOPS) de leitura de disco.
- `ReadLatency`: o tempo médio necessário por operação de entrada/saída (E/S) de leitura de disco.
- `ReadThroughput`: o número médio de bytes lidos do disco por segundo.
- `ServerlessDatabaseCapacity`: a capacidade atual do fragmento de banco de dados ou do subcluster de roteador dentro do grupo de fragmentos de banco de dados.
- `StorageNetworkReceiveThroughput`: a quantidade de throughput de rede recebida do subsistema de armazenamento do Aurora por cada instância no grupo de fragmentos de banco de dados.

- `StorageNetworkThroughput`: o throughput de rede agregado, tanto transmitido quanto recebido do subsistema de armazenamento do Aurora por cada instância no grupo de fragmentos de banco de dados.
- `StorageNetworkTransmitThroughput`: a quantidade de throughput de rede enviada ao subsistema de armazenamento do Aurora por cada instância no grupo de fragmentos de banco de dados.
- `SwapUsage`: a quantidade de espaço de troca usado pelo grupo de fragmentos do banco de dados.
- `TempStorageIOPS`: o número médio de operações de E/S realizadas no armazenamento local anexado à instância de banco de dados. Inclui operações de E/S de leitura e gravação.

  `TempStorageIOPS` pode ser usado com `TempStorageThroughput` para diagnosticar os casos raros em que a atividade de rede para transferências entre suas instâncias de banco de dados e dispositivos de armazenamento local é responsável por aumentos inesperados de capacidade.
- `TempStorageThroughput`: a quantidade de dados transferidos do armazenamento local associado a um roteador ou fragmento.
- `WriteIOPS`: o número médio de IOPS de gravação em disco.
- `WriteLatency`: o tempo médio necessário por operação de E/S de gravação em disco.
- `WriteThroughput`: o número médio de bytes gravados no disco por segundo.

As seguintes chaves de dimensão estão disponíveis para agregar as métricas `DBShardGroupInstance`:

- `DBClusterIdentifier`: o cluster de banco de dados do Aurora PostgreSQL.
- `DBShardGroupIdentifier`: o grupo de fragmentos de banco de dados ao qual a instância pertence.
- `DBShardGroupSubClusterType`: o tipo de nó, seja `Distributed Transaction Router` (roteador) ou `Data Access Shard` (fragmento).
- `DBShardGroupSubClusterIdentifier`: o nome do roteador ou fragmento ao qual a instância pertence.

A seguir estão exemplos de agregação de métricas do CloudWatch:

- Total `CPUUtilization` de todas as instâncias que pertencem a um fragmento ou roteador específico em um grupo de fragmentos de banco de dados.

- Total `CPUUtilization` de todas as instâncias em um grupo de fragmentos de banco de dados.

Métricas do DBClusterIdentifier

Para ver métricas `DBClusterIdentifier` do Aurora PostgreSQL Limitless Database no console do CloudWatch, escolha RDS e, em seguida, DBClusterIdentifier.

Ao usar o Aurora PostgreSQL Limitless Database, você pode ter mais operações de entrada/saída (E/S) do que teria com um cluster de banco de dados do Aurora. Você pode acompanhar as seguintes métricas do CloudWatch para seu cluster do Limitless Database:

- `VolumeReadIops`: o número de operações de E/S de leitura faturadas a partir de um volume de cluster, relatado em intervalos de 5 minutos.
- `VolumeWriteIops`: o número de operações de E/S do disco do gravador no volume de cluster, relatado em intervalos de 5 minutos.

O Aurora PostgreSQL Limitless Database usa a configuração de armazenamento em cluster Aurora I/O-Optimized. Com Aurora I/O-Optimized, você paga um único preço mensal por todas as operações de E/S, em vez de pagar por cada milhão de solicitações de E/S. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.

Você também pode usar mais armazenamento do que usaria em um cluster de banco de dados do Aurora. Você pode acompanhar as seguintes métricas de armazenamento do CloudWatch:

- `BackupRetentionPeriodStorageUsed`: o uso total faturado do armazenamento de backup contínuo do cluster do Aurora PostgreSQL Limitless Database.
- `SnapshotStorageUsed`: o uso total faturado do armazenamento de snapshots do cluster do Aurora PostgreSQL Limitless Database.
- `TotalBackupStorageBilled`: a soma dos custos de retenção automática de backup e snapshots de cluster de banco de dados.

  Para obter mais informações sobre os custos de armazenamento de backup, consulte Noções básicas do uso do armazenamento de backup do Amazon Aurora.
- `VolumeBytesUsed`: a quantidade de armazenamento usada pelo cluster do Aurora PostgreSQL Limitless Database, relatada em intervalos de 5 minutos.

Monitorar o Aurora PostgreSQL Limitless Database com o CloudWatch Database Insights

O modo padrão do Database Insights é necessário como parte da habilitação do Aurora PostgreSQL Limitless Database. Você pode usá-lo para monitorar em tempo real a carga do banco de dados das instâncias de banco de dados do Limitless Database. A carga do banco de dados avalia o nível de atividade de sessão em um banco de dados. Você pode usar o Database Insights para analisar e solucionar problemas de performance das instâncias de bancos de dados do Aurora PostgreSQL Limitless Database em grande escala.

Para ter mais informações sobre as métricas do CloudWatch Database Insights, consulte os tópicos a seguir.

- Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights
- CloudWatch Database Insights no Guia do usuário do Amazon CloudWatch
- Conceitos básicos do CloudWatch Database Insights no Guia do usuário do Amazon CloudWatch
- Configurar o banco de dados para monitorar consultas SQL lentas com o Database Insights para Amazon Aurora

Para ter informações sobre como ativar o modo avançado ou o modo padrão do Database Insights, consulte os tópicos a seguir.

Tópicos

- Ativar o modo avançado do Database Insights para o Aurora PostgreSQL Limitless Database
- Ativar o modo padrão do Database Insights para o Aurora PostgreSQL Limitless Database

Ativar o modo avançado do Database Insights para o Aurora PostgreSQL Limitless Database

Ativar o modo avançado do Database Insights para o Aurora PostgreSQL Limitless Database, use os procedimentos a seguir.

Ativar o modo avançado do Database Insights ao criar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database

Ative o modo avançado do Database Insights ao criar um banco de dados para o Aurora PostgreSQL Limitless Database.

Console

No console, você pode ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo avançado do Database Insights ao criar um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Selecione Criar banco de dados.
4. Na seção Database Insights, selecione Modo avançado. Em seguida, escolha as seguintes opções:
   - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de 15 a 24 meses para o modo avançado do Database Insights.
   - AWS KMS key: especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
5. Selecione Criar banco de dados.

AWS CLI

Para ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados, chame o comando create-db-cluster da AWS CLI e forneça os seguintes valores:

- `--db-cluster-identifier`: o identificador do cluster de banco de dados.
- `--database-insights-mode advanced` para ativar o modo Avançado do Database Insights.
- `--engine`: o cluster de banco de dados deve usar o mecanismo de banco de dados `aurora-postgresql`.
- `--engine-version`: o cluster de banco de dados deve usar uma das versões do mecanismo de banco de dados:

  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: o cluster de banco de dados deve usar a configuração de armazenamento `aurora-iopt1` do cluster de banco de dados.
- `--cluster-scalability-type`: especifica o modo de escalabilidade do cluster de banco de dados do Aurora. Quando definido como `limitless`, o cluster opera como um Aurora PostgreSQL Limitless Database. Quando definido como `standard` (o padrão), o cluster usa a criação normal de instância de banco de dados.

  > **Note**
  >
  > Não é possível modificar essa configuração após a criação do cluster de banco de dados.

- `--master-username`: o nome do usuário principal do cluster de banco de dados.
- `--master-user-password`: a senha do usuário principal.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.
- `--performance-insights-retention-period`: o período de retenção de dados para seu cluster de banco de dados. Para ativar o Database Insights, o período de retenção deve ser de pelo menos 465 dias.
- `--monitoring-interval`: o intervalo, em segundos, entre os pontos quando as métricas do Monitoramento aprimorado são coletadas para o cluster de banco de dados. Esse valor não pode ser `0`.
- `--monitoring-role-arn`: o nome do recurso da Amazon (ARN) do perfil do IAM que permite que o RDS envie métricas do - para o Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: você deve exportar logs `postgresql` para o CloudWatch Logs.

O exemplo a seguir ativa o modo Avançado do Database Insights ao criar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
```

```
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode advanced \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 465 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode advanced ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 465 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Para ativar o modo Avançado do Database Insights ao criar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API CreateDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `advanced`
- `Engine` para `aurora-postgresql`
- `EngineVersion` para uma versão de mecanismo disponível para o Limitless Database
- `StorageType` para `aurora-iopt1`
- `ClusterScalabilityType` para `limitless`

- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos `465` dias
- `MonitoringInterval` para um valor que não seja `0`
- `MonitoringRoleArn` para o nome do recurso da Amazon (ARN) do perfil do IAM que permite que o RDS envie métricas do monitoramento avançado para o Amazon CloudWatch Logs

Ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados do Aurora PostgreSQL Limitless Database

Ative o Database Insights ao modificar um banco de dados para o Aurora PostgreSQL Limitless Database.

> **Note**
>
> Para habilitar o Database Insights, cada instância de banco de dados em um cluster de banco de dados deve ter as mesmas configurações do Insights de Performance e do monitoramento avançado.

Console

No console, você pode ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo avançado do Database Insights ao modificar um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados e selecione Modificar.
4. Na seção Database Insights, selecione Modo avançado. Em seguida, escolha as seguintes opções:

- Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. O período de retenção deve ser de 15 a 24 meses para o modo avançado do Database Insights.
- AWS KMS key: especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.

5. Escolha Continue.
6. Em Programação das modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção programada, seu banco de dados vai ignorar essa configuração e ativar imediatamente o modo Avançado do Database Insights.
7. Escolha Modificar Cluster.

AWS CLI

Para ativar o modo Avançado do Database Insights ao modificar um cluster de banco de dados, chame o comando modify-db-cluster da AWS CLI e forneça os seguintes valores:

- `--database-insights-mode advanced` para ativar o modo Avançado do Database Insights.
- `--db-cluster-identifier`: o identificador do cluster de banco de dados.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.
- `--performance-insights-retention-period`: o período de retenção de dados do cluster de banco de dados. Para ativar o modo Avançado do Database Insights, o período de retenção deve ser de pelo menos 465 dias.

O exemplo a seguir ativa o modo Avançado do Database Insights ao modificar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
```

```
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para ativar o modo avançado do Database Insights ao modificar uma um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API ModifyDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `advanced`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos `465` dias

Ativar o modo padrão do Database Insights para o Aurora PostgreSQL Limitless Database

Ativar o modo padrão do Database Insights para o Aurora PostgreSQL Limitless Database, use os procedimentos a seguir.

Ativar o modo padrão do Database Insights ao criar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database

Ative o modo padrão do Database Insights ao criar um banco de dados para o Aurora PostgreSQL Limitless Database.

Console

No console, você pode ativar o modo Standard do Database Insights ao criar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo padrão do Database Insights ao criar um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Selecione Criar banco de dados.
4. Na seção Database Insights, selecione Modo Standard. Em seguida, escolha as seguintes opções:
   - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. Para criar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database, o período de retenção deve ser de pelo menos 31 dias.
   - AWS KMS key: especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.
5. Selecione Criar banco de dados.

AWS CLI

Para ativar o modo Standard do Database Insights ao criar um cluster de banco de dados, chame o comando create-db-cluster da AWS CLI e forneça os seguintes valores:

- `--db-cluster-identifier`: o identificador do cluster de banco de dados.
- `--database-insights-mode standard` para ativar o modo Standard do Database Insights.
- `--engine`: o cluster de banco de dados deve usar o mecanismo de banco de dados `aurora-postgresql`.
- `--engine-version`: o cluster de banco de dados deve usar uma das versões do mecanismo de banco de dados:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: o cluster de banco de dados deve usar a configuração de armazenamento `aurora-iopt1` do cluster de banco de dados.

- `--cluster-scalability-type`: especifica o modo de escalabilidade do cluster de banco de dados do Aurora. Quando definido como `limitless`, o cluster opera como um Aurora PostgreSQL Limitless Database. Quando definido como `standard` (o padrão), o cluster usa a criação normal de instância de banco de dados.

  > **Note**
  >
  > Não é possível modificar essa configuração após a criação do cluster de banco de dados.

- `--master-username`: o nome do usuário principal do cluster de banco de dados.
- `--master-user-password`: a senha do usuário principal.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.
- `--performance-insights-retention-period`: o período de retenção de dados para seu cluster de banco de dados. Para criar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database, o período de retenção deve ser de pelo menos 31 dias.
- `--monitoring-interval`: o intervalo, em segundos, entre os pontos quando as métricas do Monitoramento aprimorado são coletadas para o cluster de banco de dados. Esse valor não pode ser `0`.
- `--monitoring-role-arn`: o nome do recurso da Amazon (ARN) do perfil do IAM que permite que o RDS envie métricas do - para o Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: você deve exportar logs `postgresql` para o CloudWatch Logs.

O exemplo a seguir habilita o modo padrão do Database Insights ao criar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode standard \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
```

```
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 31 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode standard ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 31 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Para ativar o modo padrão do Database Insights ao criar , especifique os parâmetros a seguir para a operação de API CreateDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `standard`
- `Engine` para `aurora-postgresql`
- `EngineVersion` para uma versão de mecanismo disponível para o Limitless Database
- `StorageType` para `aurora-iopt1`
- `ClusterScalabilityType` para `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos 31 dias

- `MonitoringInterval` para um valor que não seja `0`
- `MonitoringRoleArn` para o nome do recurso da Amazon (ARN) do perfil do IAM que permite que o RDS envie métricas do monitoramento avançado para o Amazon CloudWatch Logs

Ativar o modo padrão do Database Insights ao modificar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database

Ative o Database Insights ao modificar um banco de dados para o Aurora PostgreSQL Limitless Database.

> **Note**
>
> Para habilitar o Database Insights, cada instância de banco de dados em um cluster de banco de dados deve ter as mesmas configurações do Insights de Performance e do monitoramento avançado.

Console

No console, você pode ativar o modo Standard do Database Insights ao criar um cluster de banco de dados. As configurações do Database Insights se aplicam a todas as instâncias de banco de dados em seu cluster de banco de dados.

Como ativar o modo padrão do Database Insights ao modificar um cluster de banco de dados usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados e selecione Modificar.
4. Na seção Database Insights, selecione Modo Standard. Em seguida, escolha as seguintes opções:

   - Retention (Retenção) – a duração do período de retenção de dados do Performance Insights. Para criar um cluster de banco de dados para o Aurora PostgreSQL Limitless Database, o período de retenção deve ser de pelo menos 31 dias.
   - AWS KMS key: especifique a sua chave do KMS. O Performance Insights criptografa todos os possíveis dados sigilosos usando a sua chave do KMS. Os dados são criptografados

em repouso e em trânsito. Para obter mais informações, consulte Criptografar recursos do Amazon Aurora.

5. Escolha Continue.
6. Em Programação das modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção programada, seu banco de dados vai ignorar essa configuração e ativar imediatamente o modo Padrão do Database Insights.
7. Escolha Modificar Cluster.

AWS CLI

Para ativar o modo Standard do Database Insights ao modificar um cluster de banco de dados, chame o comando modify-db-cluster da AWS CLI e forneça os seguintes valores:

- `--db-cluster-identifier`: o identificador do cluster de banco de dados.
- `--database-insights-mode standard` para ativar o modo Standard do Database Insights.
- `--enable-performance-insights` para ativar o Performance Insights para o Database Insights.
- `--performance-insights-retention-period`: o período de retenção de dados para seu cluster de banco de dados. Para ativar o modo padrão do Database Insights, o período de retenção deve ser de pelo menos 31 dias.

O exemplo a seguir habilita o modo padrão do Database Insights ao modificar um cluster de banco de dados.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 31
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
```

```
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 31
```

RDS API

Para ativar o modo padrão do Database Insights ao modificar um cluster de banco de dados, especifique os parâmetros a seguir para a operação de API ModifyDBCluster do Amazon RDS.

- `DatabaseInsightsMode` para `standard`
- `EnablePerformanceInsights` para `True`
- `PerformanceInsightsRetentionPeriod` para pelo menos `31` dias

Monitoramento do Aurora PostgreSQL Limitless Database com o Amazon CloudWatch Logs

A exportação de logs do PostgreSQL para o CloudWatch Logs é necessária como parte da habilitação do Aurora PostgreSQL Limitless Database. Você pode acessar e analisar esses logs no CloudWatch Logs Insights, de forma semelhante ao acesso aos logs do PostgreSQL para um cluster de banco de dados do Aurora PostgreSQL padrão. Para ter mais informações, consulte Analisar logs do PostgreSQL utilizando o CloudWatch Logs Insights.

O nome do grupo de logs para o cluster de banco de dados é o mesmo do Aurora PostgreSQL:

```
/aws/rds/cluster/DB_cluster_ID/postgresql
```

O nome do grupo de logs para o grupo de fragmentos de banco de dados tem o seguinte formato:

```
/aws/rds/cluster/DB_cluster_ID/DB_shard_group_ID/postgresql
```

Há fluxos de logs para cada nó (roteador ou fragmento). Os nomes têm o seguinte formato:

```
[DistributedTransactionRouter|
DataAccessShard]/node_cluster_serial_ID-node_instance_serial_ID/n
```

Por exemplo:

- Roteador: `DistributedTransactionRouter/6-6.2`
- Fragmento: `DataAccessShard/22-22.0`

> **Note**
>
> Não é possível visualizar os arquivos de log do PostgreSQL para o grupo de fragmentos de banco de dados diretamente no console do RDS, na AWS CLI ou na API do RDS, como você pode fazer para o cluster de banco de dados. Você deve usar o CloudWatch Logs Insights para visualizá-los.

Monitoramento do Aurora PostgreSQL Limitless Database com Monitoramento aprimorado

O Monitoramento aprimorado é necessário como parte da ativação do Aurora PostgreSQL Limitless Database. Você pode usá-lo para monitorar o sistema operacional das instâncias de banco de dados do Limitless Database em tempo real.

O Aurora publica métricas de Monitoramento aprimorado no CloudWatch Logs. Algumas das principais métricas disponíveis incluem conexões de banco de dados, uso de armazenamento e latência de consultas. Elas podem ajudar a identificar gargalos de desempenho.

Para ter mais informações sobre as métricas do Monitoramento Avançado, consulte Métricas do SO para Aurora.

Monitoramento do Aurora PostgreSQL Limitless Database com o Insights de Performance

Use o Insights de Performance para monitorar o cluster do Aurora PostgreSQL Limitless Database. O Insights de Performance funciona de forma semelhante para o Aurora PostgreSQL Limitless Database e para clusters de banco de dados do Aurora padrão. No entanto, você monitora métricas no nível do grupo de fragmentos para o Aurora PostgreSQL Limitless Database.

As duas principais métricas do Insights de Performance a serem monitoradas são as seguintes:

- Carga do banco de dados: mede o nível de atividade no seu banco de dados. A métrica chave do Performance Insights é `DBLoad`, que é coletada a cada segundo.

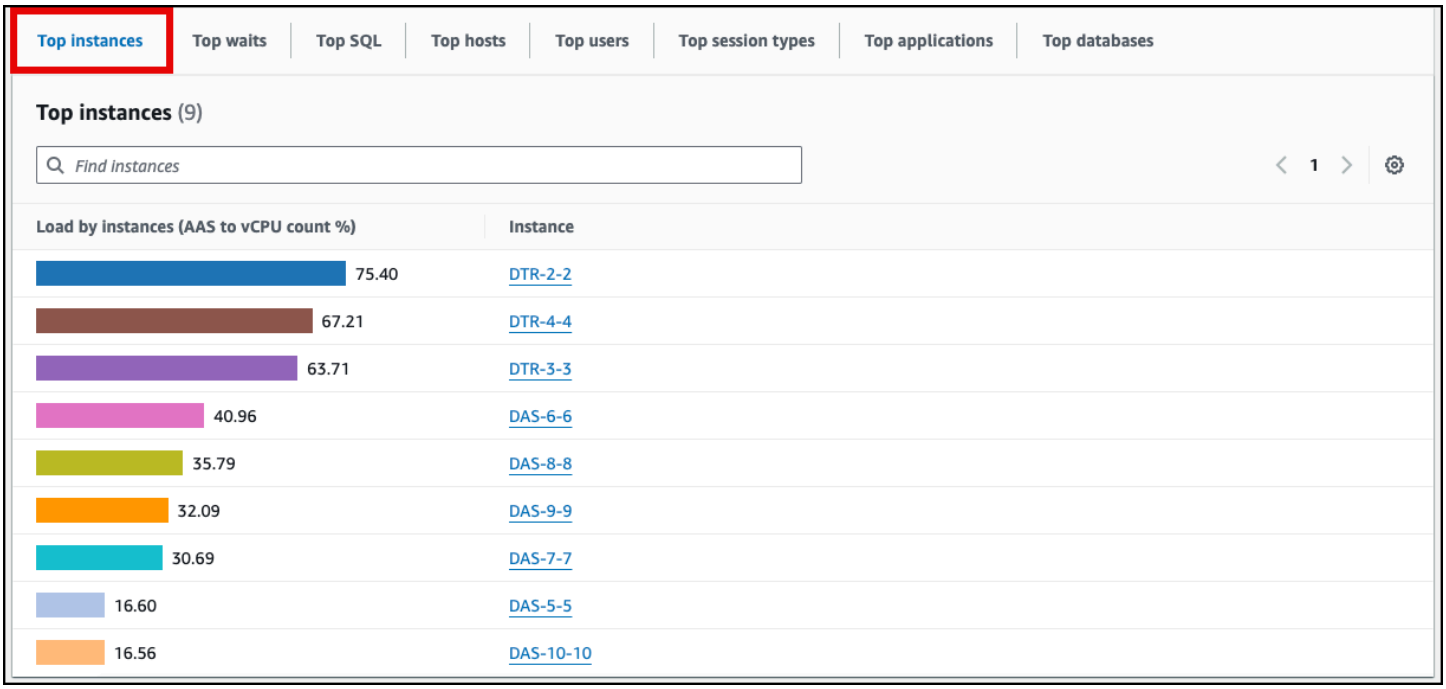
  A unidade da métrica `DBLoad` no Insights de Performance é a média de sessões ativas (AAS). Para obter a média de sessões ativas, o Performance Insights obtém uma amostra do número de sessões que executam simultaneamente uma consulta. A AAS é o número total de sessões, dividido pelo número total de amostras por um determinado período. Para obter mais informações sobre `DBLoad` e AAS, consulte Carga de banco de dados.
- CPU máxima: a potência computacional máxima disponível para seu banco de dados. Para ver se as sessões ativas estão excedendo o máximo de CPU, observe sua relação com a linha `Max vCPU`. O valor de `Max vCPU` é determinado pelo número de núcleos de vCPU (CPUs virtuais) da instância de banco de dados. Para obter mais informações sobre `Max vCPU`, consulte Máximo de CPU.

Além disso, você pode "dividir" a métrica `DBLoad` em dimensões, que são subcategorias da métrica. As dimensões mais úteis são as seguintes:

- Principais instâncias: mostra a carga relativa do banco de dados para suas instâncias (fragmentos e roteadores) em ordem decrescente.
- Eventos de espera: fazem com que instruções SQL aguardem que eventos específicos aconteçam antes que possam continuar a execução. Eventos de espera indicam onde o trabalho está impedido.
- Principal SQL: mostra quais consultas contribuem mais para a carga do banco de dados.

Para ter mais informações sobre o Insights de Performance, consulte Dimensões.

A figura a seguir mostra a dimensão Principais instâncias para um grupo de fragmentos de banco de dados.



Tópicos

- Analisar a carga do banco de dados do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

Analisar a carga do banco de dados do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

Com o Insights de Performance, é possível monitorar métricas no nível do grupo de fragmentos e no nível da instância para o Aurora PostgreSQL Limitless Database. Ao analisar a carga de banco de dados do Aurora PostgreSQL Limitless Database, talvez você queira comparar a carga de banco de dados de cada fragmento e roteador com a vCPU máxima.

> **Note**
>
> O Aurora PostgreSQL Limitless Database sempre tem o Insights de Performance e o Monitoramento aprimorado habilitados. O período mínimo de retenção dos dados do Insights de Performance para o Limitless Database é de 31 dias (1 mês).

A visualização Absoluto mostra o número da Média de sessões ativas (AAS) e a vCPU estimada. A visualização Relativo mostra a proporção de AAS em relação à vCPU estimada.

Tópicos

- Analisar a carga relativa de banco de dados do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance
- Análise da carga do banco de dados por esperas no Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance
- Analisar a distribuição de carga do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

Analisar a carga relativa de banco de dados do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

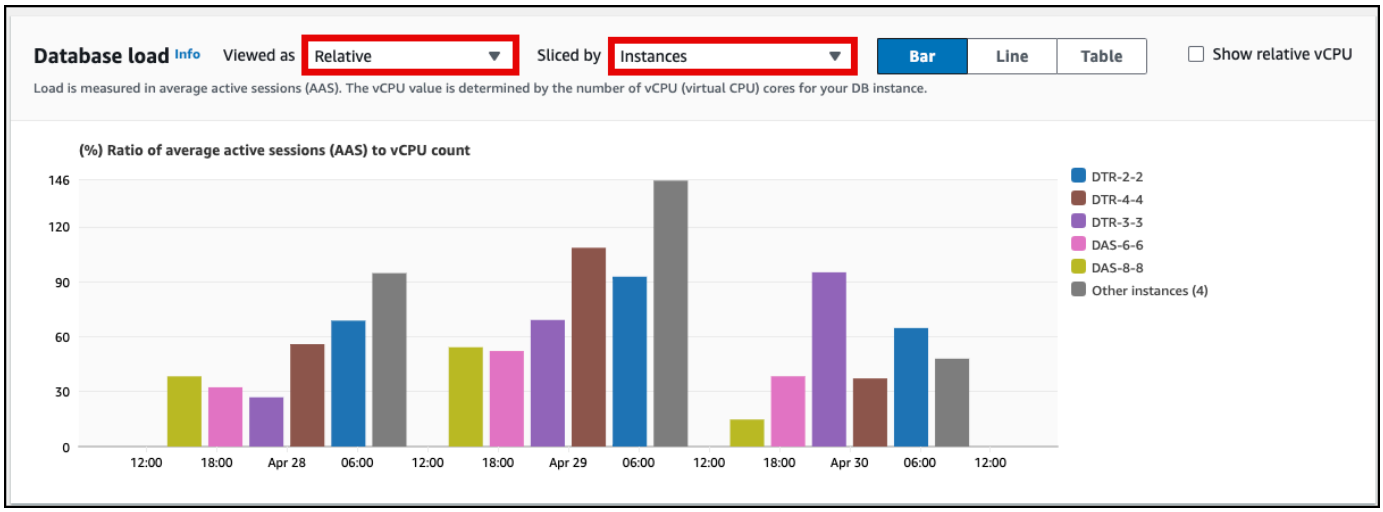
Talvez você queira melhorar o desempenho do Aurora PostgreSQL Limitless Database monitorando a carga relativa de banco de dados. Para analisar a carga relativa de banco de dados por instância do Aurora PostgreSQL Limitless Database, use o procedimento a seguir.

Como analisar a carga relativa de banco de dados do Aurora PostgreSQL Limitless Database usando o console

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

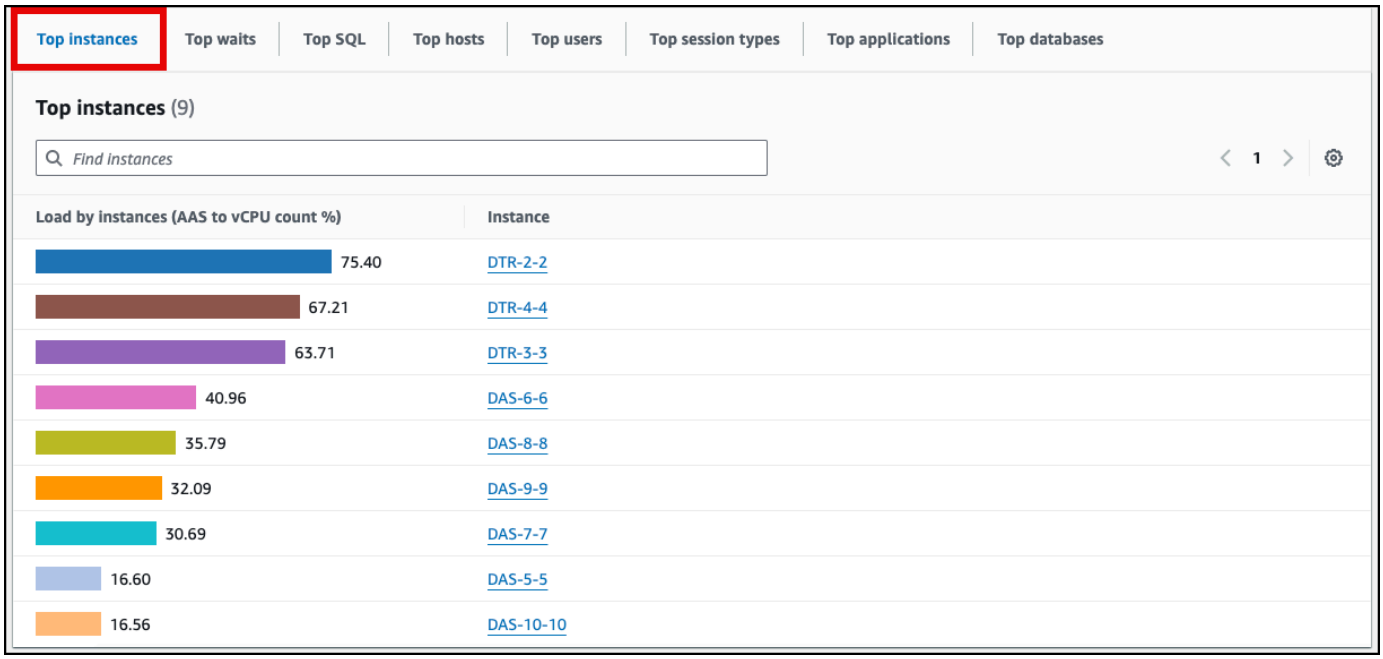
2. No painel de navegação, escolha Performance Insights.
3. Escolha um Aurora PostgreSQL Limitless Database. O painel do Insights de Performance será exibido para esse Aurora PostgreSQL Limitless Database.
4. Na seção Carga do banco de dados (carga do BC), escolha Instâncias em Fatiado por. Para ver a proporção entre a Média de sessões ativas (AAS) e os núcleos de vCPU de todas as instâncias no Aurora PostgreSQL Limitless Database, escolha Relativo em Visto como.

   O gráfico Média de sessões ativas (AAS) mostra a carga do banco de dados para instâncias no Aurora PostgreSQL Limitless Database.



5. Para ver as principais instâncias, escolha a guia Principais instâncias.

   No exemplo a seguir, a instância com a carga do banco de dados mais alta é `DTR-2-2`.



6. (Opcional) Para analisar a carga de banco de dados de uma instância no Aurora PostgreSQL Limitless Database, escolha o nome da instância na coluna Instâncias. Para visualizar a carga de banco de dados de `DTR-2-2`, escolha `DTR-2-2` na coluna Instâncias.

> **Note**
>
> Você pode visualizar as métricas do Insights de Performance somente para instâncias no Aurora PostgreSQL Limitless Database.

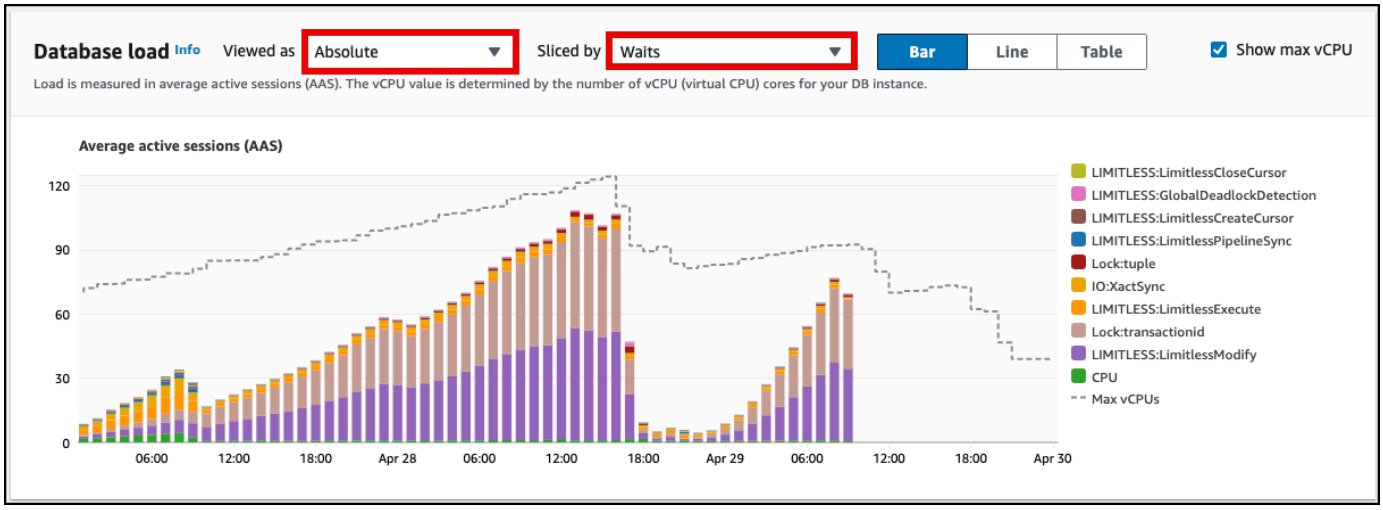
Análise da carga do banco de dados por esperas no Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

Talvez você queira melhorar o desempenho do Aurora PostgreSQL Limitless Database monitorando eventos de espera. Para analisar a carga do banco de dados por eventos de espera do Aurora PostgreSQL Limitless Database, use o procedimento a seguir.

Como analisar a carga do banco de dados por esperas do Aurora PostgreSQL Limitless Database usando o console

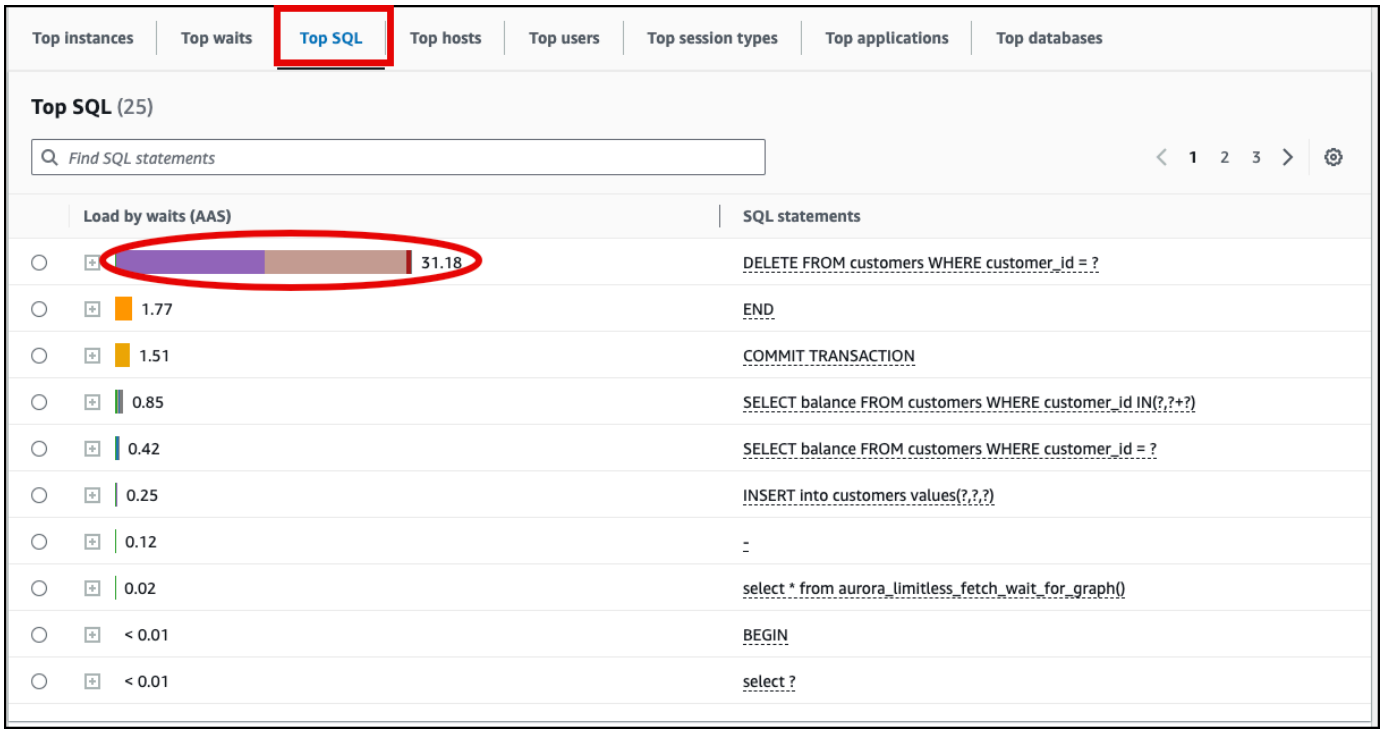
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha um Aurora PostgreSQL Limitless Database. O painel do Insights de Performance será exibido para esse Aurora PostgreSQL Limitless Database.
4. Na seção Carga do banco de dados (carga do BC), escolha Esperas em Fatiado por. Para visualizar o número de AAS e a vCPU estimada, escolha Absoluto para Visualizado como.

   O gráfico Média de sessões ativas (AAS) mostra a carga do banco de dados para instâncias no Aurora PostgreSQL Limitless Database.



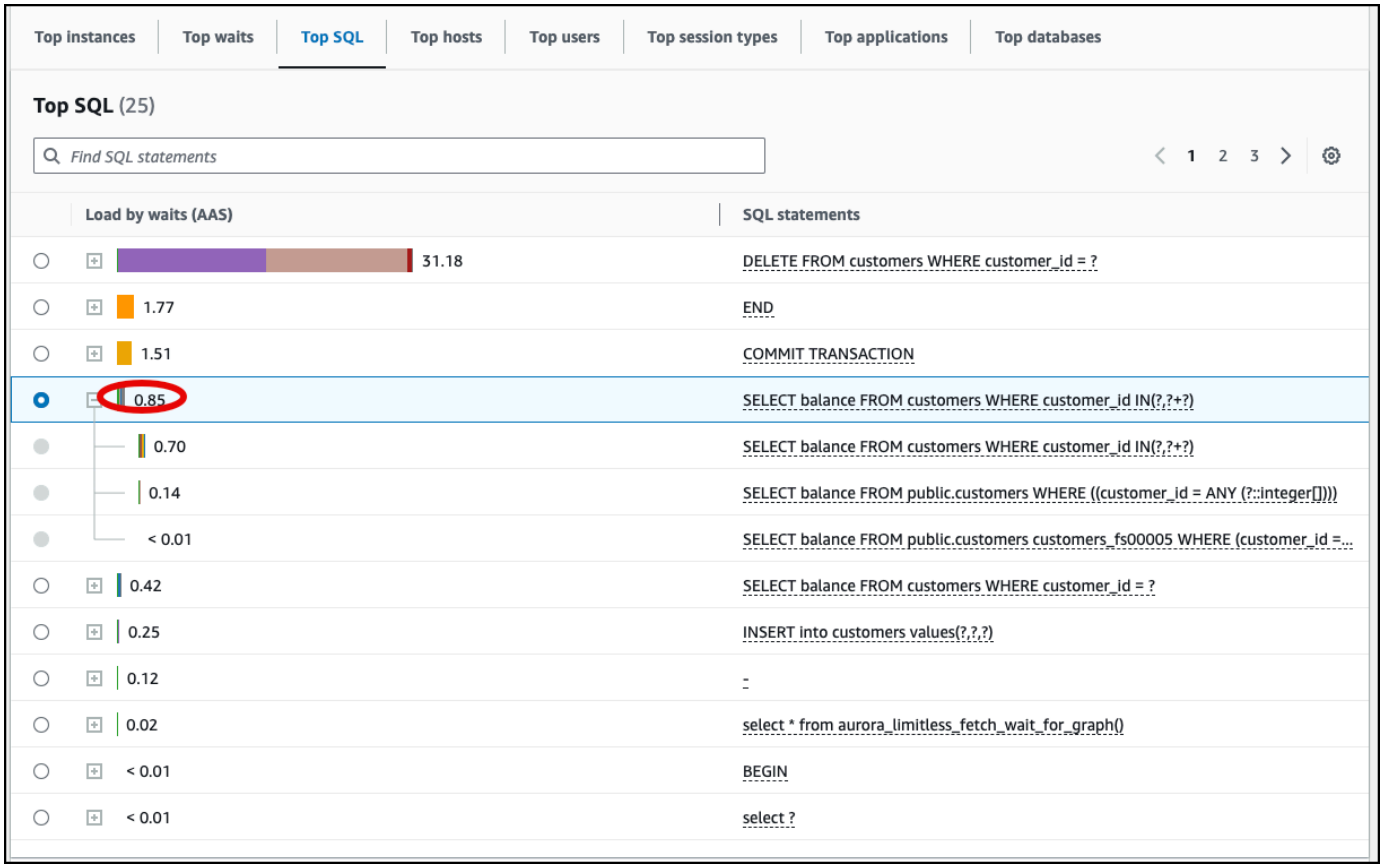
5. Role para baixo até a guia Top SQL (SQL principal).

   No exemplo a seguir, a instrução SQL com a maior carga por esperas é a instrução `DELETE`.



6. Escolha a instrução SQL para expandi-la nas instruções de componente.

   No exemplo a seguir, a instrução `SELECT` tem 3 instruções de componente.



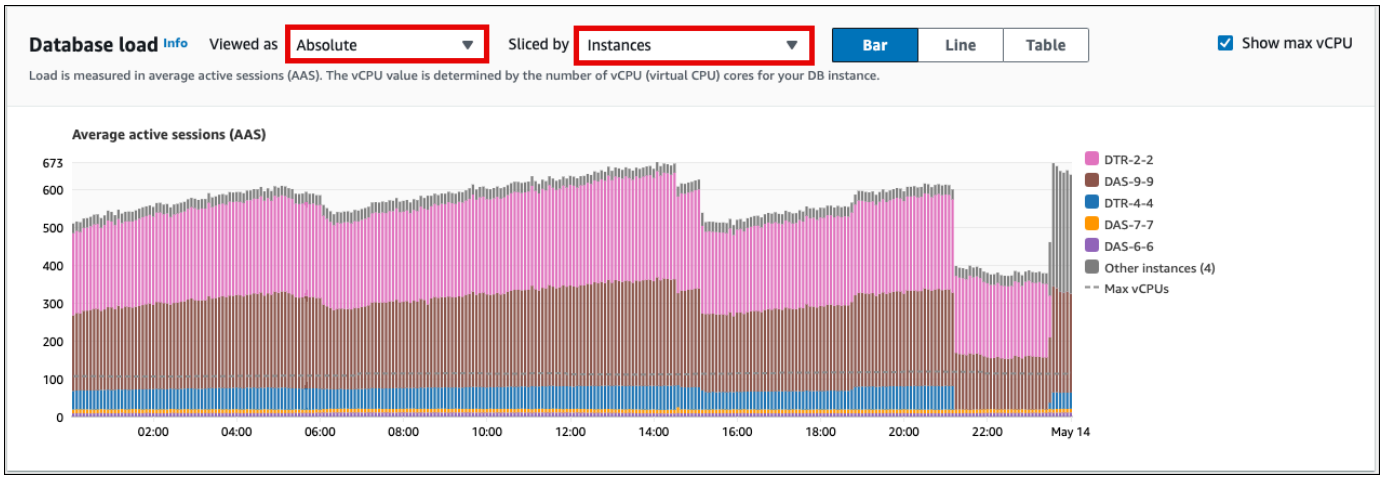
Analisar a distribuição de carga do Aurora PostgreSQL Limitless Database usando o painel do Insights de Performance

Talvez você queira equilibrar a distribuição de carga das instâncias no Aurora PostgreSQL Limitless Database. Para analisar a distribuição de carga das instâncias no Aurora PostgreSQL Limitless Database, use o procedimento a seguir.

Como analisar a distribuição de carga das instâncias no Aurora PostgreSQL Limitless Database usando o console

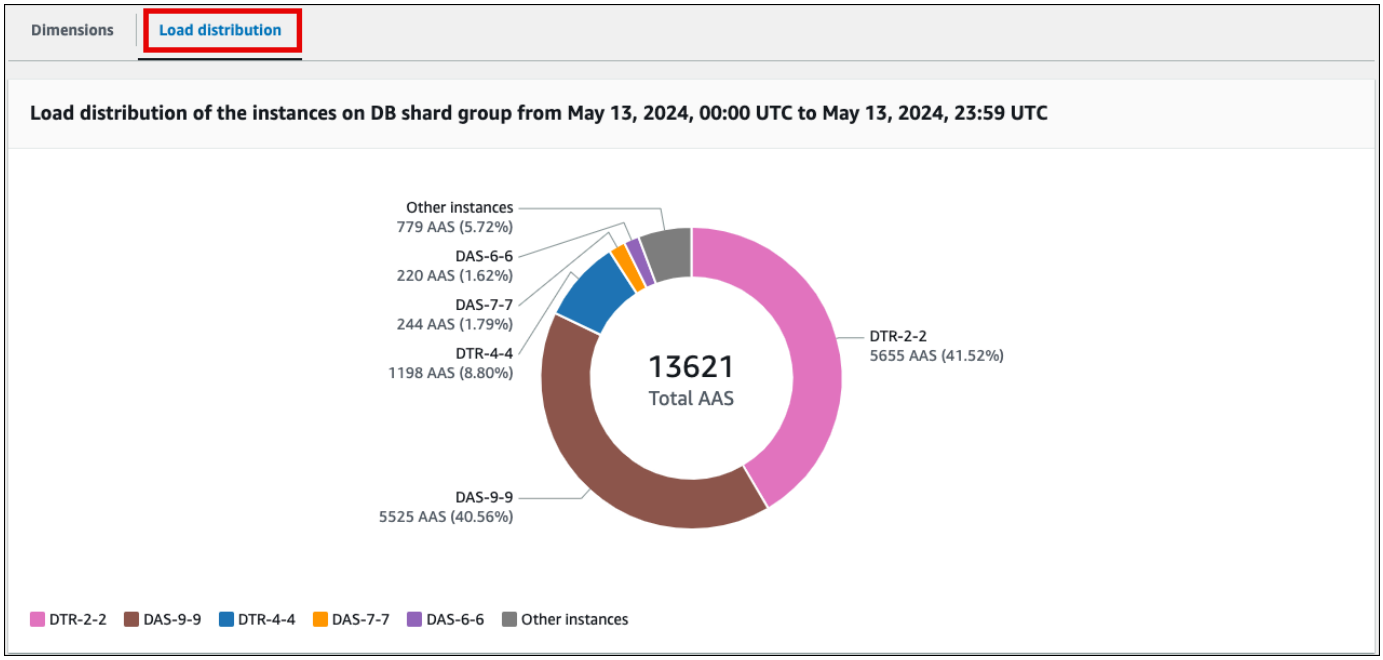
1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Performance Insights.
3. Escolha um Aurora PostgreSQL Limitless Database. O painel do Insights de Performance será exibido para esse Aurora PostgreSQL Limitless Database.
4. Na seção Carga do banco de dados (carga do BC), escolha Instâncias em Fatiado por. Para visualizar o número de AAS e a vCPU estimada para todas as instâncias no Aurora PostgreSQL Limitless Database, escolha Absoluto em Visto como.

O gráfico Média de sessões ativas (AAS) mostra a carga do banco de dados para instâncias no Aurora PostgreSQL Limitless Database.



5. Para ver um gráfico da distribuição de carga das instâncias no Aurora PostgreSQL Limitless Database, escolha a guia Distribuição de carga.

   No exemplo a seguir, a instância com a carga do banco de dados mais alta é `DTR-2-2`.



Monitoramento do Aurora PostgreSQL Limitless Database com o Amazon GuardDuty RDS Protection

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Usando modelos de machine learning (ML) e recursos de detecção de anomalias e ameaças, o GuardDuty monitora continuamente diferentes fontes de log e atividades de runtime para identificar e priorizar possíveis riscos de segurança e atividades maliciosas no seu ambiente.

O GuardDuty RDS Protection analisa e traça o perfil de eventos de login em busca de possíveis ameaças de acesso aos seus bancos de dados Amazon Aurora. Quando você ativa o RDS Protection, o GuardDuty consome eventos de login do RDS de seus bancos de dados Aurora. O RDS Protection monitora esses eventos e traça o perfil de possíveis ameaças internas ou agentes externos.

Para obter mais informações sobre o GuardDuty RDS Protection no Aurora, consulte Monitorar ameaças com o Amazon GuardDuty RDS Protection para Amazon Aurora.

Para ter mais informações sobre como habilitar a Proteção do GuardDuty RDS, consulte Proteção do GuardDuty RDS no Guia do usuário do Amazon GuardDuty.

Funções e visualizações do Aurora PostgreSQL Limitless Database

O Aurora PostgreSQL Limitless Database adicionou funções e visualizações. Elas são baseadas nas funções e visualizações correspondentes do Aurora PostgreSQL.

> **Note**
>
> Algumas estatísticas podem retornar resultados inconsistentes se houver transações em andamento.

Tópicos

- Funções do Aurora PostgreSQL Limitless Database
- Visualizações do Aurora PostgreSQL Limitless Database

Funções do Aurora PostgreSQL Limitless Database

A tabela a seguir mostra as novas funções do Aurora PostgreSQL Limitless Database.

> **Note**
>
> As funções listadas nesta tabela estão localizadas no esquema `rds_aurora`. Ao usar uma função do Limitless Database, inclua o nome do objeto totalmente qualificado: `rds_aurora.`*`object_name`*.

Função do Aurora PostgreSQL Limitless Database	Função correspondente do Aurora PostgreSQL
limitless_backend_dsid	pg_backend_pid
limitless_cancel_session	pg_cancel_backend
limitless_stat_clear_snapshot	pg_stat_clear_snapshot
limitless_stat_database_size	pg_database_size

Função do Aurora PostgreSQL Limitless Database	Função correspondente do Aurora PostgreSQL
limitless_stat_get_snapshot_timestamp	pg_stat_get_snapshot_timestamp
limitless_stat_prepared_xacts	pg_prepared_xacts
limitless_stat_relation_sizes	pg_indexes_size, pg_relation_size, pg_table_size, pg_total_relation_size
limitless_stat_reset	pg_stat_reset
limitless_stat_statements_reset	pg_stat_statements_reset
limitless_stat_system_waits	aurora_stat_system_waits
limitless_terminate_session	pg_terminate_backend
limitless_wait_report	aurora_wait_report

Os exemplos a seguir fornecem detalhes sobre as funções do Aurora PostgreSQL Limitless Database. Para obter mais informações sobre funções do PostgreSQL, consulte Functions and operators na documentação do PostgreSQL.

limitless_backend_dsid

A função `limitless_backend_dsid` retorna o ID da sessão distribuída para a sessão atual. Uma sessão distribuída é executada em um roteador em um grupo de fragmentos de banco de dados e envolve processos de backend em um ou mais fragmentos no grupo de fragmentos de banco de dados.

O exemplo a seguir mostra como usar a função `limitless_backend_dsid`.

```
SELECT rds_aurora.limitless_backend_dsid();

limitless_backend_dsid
------------------------
8CACD7B04D0FC2A5
(1 row)
```

limitless_cancel_session

A função `limitless_cancel_session` funciona de forma semelhante a `pg_cancel_backend`, mas tenta cancelar todos os processos de backend relacionados ao ID de sessão distribuído fornecido enviando um `SIGINT` (sinal de interrupção).

O parâmetro de entrada é o seguinte:

- `distributed_session_id` (texto): o ID da sessão distribuída a ser cancelada.

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `pid` (texto): o ID do processo de backend.
- `success` (booleano): se o cancelamento foi bem-sucedido.

O exemplo a seguir mostra como usar a função `limitless_cancel_session`.

```
SELECT * FROM rds_aurora.limitless_cancel_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_stat_clear_snapshot

A função `limitless_stat_clear_snapshot` descarta o snapshot das estatísticas atuais ou as informações em cache em todos os nós.

O exemplo a seguir mostra como usar a função `limitless_stat_clear_snapshot`.

```
SELECT rds_aurora.limitless_stat_clear_snapshot();
```

limitless_stat_database_size

A função `limitless_stat_database_size` retorna os tamanhos de um banco de dados no grupo de fragmentos de banco de dados.

O parâmetro de entrada é o seguinte:

- `dbname` (nome): o banco de dados para o qual obter os tamanhos.

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `db_size`: o tamanho do banco de dados nesse subcluster em bytes.

O exemplo a seguir mostra como usar a função `limitless_stat_database_size`.

```
SELECT * FROM rds_aurora.limitless_stat_database_size('postgres_limitless');

 subcluster_id | subcluster_type | db_size
---------------+-----------------+----------
             1 | router          |  8895919
             2 | router          |  8904111
             3 | shard           | 21929391
             4 | shard           | 21913007
             5 | shard           | 21831087
(5 rows)
```

limitless_stat_get_snapshot_timestamp

A função `limitless_stat_get_snapshot_timestamp` retorna a data e hora do snapshot de estatísticas atual ou `NULL` se nenhum instantâneo de estatísticas foi obtido. Um snapshot é obtido na primeira vez que as estatísticas cumulativas são acessadas em uma transação, se `stats_fetch_consistency` estiver definido como `snapshot`. Retorna uma visão consolidada dos registros de data e hora dos snapshots de todos os nós. As colunas `subcluster_id` e `subcluster_type` mostram de qual nó os dados são provenientes.

O exemplo a seguir mostra como usar a função `limitless_stat_get_snapshot_timestamp`.

```
SELECT * FROM rds_aurora.limitless_stat_get_snapshot_timestamp();

 subcluster_id | subcluster_type | snapshot_timestamp
---------------+-----------------+--------------------
             1 | router          |
             2 | router          |
             3 | shard           |
             4 | shard           |
             5 | shard           |
(5 rows)
```

limitless_stat_prepared_xacts

A função `limitless_stat_prepared_xacts` retorna informações sobre transações em todos os nós que estão atualmente preparados para a confirmação em duas fases. Para obter mais informações, consulte pg_prepared_xacts na documentação do PostgreSQL.

O exemplo a seguir mostra como usar a função `limitless_stat_prepared_xacts`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_prepared_xacts;

 subcluster_id | subcluster_type | transaction_id |             gid              |
         prepared            |  owner_id  |    database_id
---------------+-----------------+----------------+------------------------------
+-------------------------------+------------+--------------------
 8             | shard           |        5815978 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659603+00 | auroraperf | postgres_limitless
 12            | shard           |        4599138 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659637+00 | auroraperf | postgres_limitless
(2 rows)
```

limitless_stat_relation_sizes

A função `limitless_stat_relation_sizes` retorna os diferentes tamanhos de uma tabela no grupo de fragmentos de banco de dados.

Os parâmetros de entrada são os seguintes:

- `relnspname` (nome): o nome do esquema que contém a tabela.
- `relname` (nome): o nome da tabela.

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `main_size`: o tamanho em bytes da bifurcação de dados principal nesse nó.
- `fsm_size`: o tamanho em bytes do mapa de espaço livre da tabela nesse nó.
- `vm_size`: o tamanho em bytes do mapa de visibilidade da tabela nesse nó.
- `init_size`: o tamanho em bytes da inicialização da tabela nesse nó.
- `toast_size`: o tamanho em bytes da tabela toast associada à tabela nessa bifurcação.

- `index_size`: o tamanho em bytes de todos os índices da tabela nesse nó.
- `total_size`: o tamanho em bytes de todos os segmentos da tabela nesse nó.

O exemplo a seguir mostra como usar a função `limitless_stat_relation_sizes` (algumas colunas são omitidas).

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customers');

 subcluster_id | subcluster_type | main_size | fsm_size | vm_size | toast_size |
 table_size | total_size
---------------+-----------------+-----------+----------+---------+------------
+------------+------------
             1 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             2 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             3 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             4 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             5 | shard           |   3981312 |  4227072 | 1409024 |    1409024 |
 11026432 |   11026432
(5 rows)
```

limitless_stat_reset

A função `limitless_stat_reset` redefine todos os contadores de estatísticas do banco de dados atual para zero (0). Se `track_functions` estiver habilitado, a coluna `stats_reset` em `limitless_stat_database` mostrará a última vez em que as estatísticas foram redefinidas para o banco de dados. Por padrão, `limitless_stat_reset` pode ser executado apenas por um superusuário. Outros usuários podem receber permissão usando o privilégio `EXECUTE`.

O exemplo a seguir mostra como usar a função `limitless_stat_reset`.

```
SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

 tup_inserted | tup_deleted
--------------+-------------
          896 |           0
(1 row)
```

```
SELECT rds_aurora.limitless_stat_reset();

limitless_stat_reset
--------------------
(1 row)

SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

tup_inserted | tup_deleted
-------------+-------------
           0 |           0
(1 row)
```

limitless_stat_statements_reset

A função `limitless_stat_statements_reset` descarta as estatísticas coletadas até o momento por `limitless_stat_statements` correspondentes aos parâmetros `username`, `dbname`, `distributed_query_id` e `queryid` especificados. Se algum dos parâmetros não for especificado, o valor padrão `""` ou `0` (inválido) será usado para cada um deles e as estatísticas que corresponderem a outros parâmetros serão redefinidas. Se nenhum parâmetro for especificado, ou se todos os parâmetros especificados forem `""` ou `0` (inválidos), a função descartará todas as estatísticas. Se todas as estatísticas na visualização `limitless_stat_statements` forem descartadas, a função também redefinirá as estatísticas na visualização `limitless_stat_statements_info`.

Os parâmetros de entrada são os seguintes:

- `username` (nome): o usuário que consultou a instrução.
- `dbname` (nome): o banco de dados onde a consulta foi executada.
- `distributed_query_id` (bigint): o ID da consulta principal do nó coordenador. Essa coluna é `NULL` se for a consulta principal. O nó coordenador envia o ID da consulta distribuída para os nós participantes. Portanto, para os nós participantes, os valores para o ID de consulta distribuída e o ID de consulta são diferentes.
- `queryid` (bigint): o ID de consulta da instrução.

O exemplo a seguir mostra como usar a função `limitless_stat_statements_reset` para redefinir todas as estatísticas coletadas pelo `limitless_stat_statements`.

```
SELECT rds_aurora.limitless_stat_statements_reset();
```

limitless_stat_system_waits

A função `limitless_stat_system_waits` retorna uma visão consolidada dos dados do evento de espera de `aurora_stat_system_waits`, que relata a atividade de espera em todo o sistema em uma instância, de todos os nós. As colunas `subcluster_id` e `subcluster_type` mostram de qual nó os dados são provenientes.

O exemplo a seguir mostra como usar a função `limitless_stat_system_waits`.

```
postgres_limitless=> SELECT *
FROM rds_aurora.limitless_stat_system_waits() lssw, pg_catalog.aurora_stat_wait_event()
 aswe
WHERE lssw.event_id=aswe.event_id and aswe.event_name='LimitlessTaskScheduler';

 subcluster_id | subcluster_type | type_id | event_id  | waits  |  wait_time   |
 event_name
---------------+-----------------+---------+-----------+--------+--------------
+------------------------
             1 | router          |      12 | 201326607 | 677068 | 616942216307 |
 LimitlessTaskScheduler
             2 | router          |      12 | 201326607 | 678586 | 616939897111 |
 LimitlessTaskScheduler
             3 | shard           |      12 | 201326607 | 756640 | 616965545172 |
 LimitlessTaskScheduler
             4 | shard           |      12 | 201326607 | 755184 | 616958057620 |
 LimitlessTaskScheduler
             5 | shard           |      12 | 201326607 | 757522 | 616963183539 |
 LimitlessTaskScheduler
(5 rows)
```

limitless_terminate_session

A função `limitless_terminate_session` funciona de forma semelhante a `pg_terminate_backend`, mas tenta encerrar todos os processos de backend relacionados ao ID de sessão distribuído fornecido enviando um `SIGTERM` (sinal de término).

O parâmetro de entrada é o seguinte:

- `distributed_session_id` (texto): o ID da sessão distribuída a ser encerrada.

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.

- `pid` (texto): o ID do processo de backend.
- `success` (booleano): se o processo foi encerrado com sucesso.

O exemplo a seguir mostra como usar a função `limitless_terminate_session`.

```
SELECT * FROM rds_aurora.limitless_terminate_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_wait_report

A função `limitless_wait_report` retorna a atividade de eventos de espera por um período de todos os nós. As colunas `subcluster_id` e `subcluster_type` mostram de qual nó os dados são provenientes.

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.

As demais colunas são as mesmas que em `aurora_wait_report`.

O exemplo a seguir mostra como usar a função `limitless_wait_report`.

```
postgres_limitless=> select * from rds_aurora.limitless_wait_report();

 subcluster_id | subcluster_type | type_name | event_name | waits | wait_time |
 ms_per_wait | waits_per_xact | ms_per_xact
---------------+-----------------+-----------+------------+-------+-----------
+-------------+--------------- +-------------
             1 | router          | Client    | ClientRead |    57 | 741550.14 |
 13009.652 |           0.19 |    2505.237
             5 | shard           | Client    | ClientRead |    54 | 738897.68 |
 13683.290 |           0.18 |    2496.276
             4 | shard           | Client    | ClientRead |    54 | 738859.53 |
 13682.584 |           0.18 |    2496.147
             2 | router          | Client    | ClientRead |    53 | 719223.64 |
 13570.257 |           0.18 |    2429.810
```

```
            3 | shard           | Client    | ClientRead |    54 | 461720.40 |
8550.378 |           0.18 |    1559.86
```

Visualizações do Aurora PostgreSQL Limitless Database

A tabela a seguir mostra as novas visualizações do Aurora PostgreSQL Limitless Database.

> **Note**
>
> As visualizações listadas nessa tabela estão localizadas no esquema `rds_aurora`. Ao usar uma visualização do Limitless Database, inclua o nome do objeto totalmente qualificado: `rds_aurora.object_name`.

Visualização do Aurora PostgreSQL Limitless Database	Visualização correspondente do Aurora PostgreSQL
limitless_database	pg_database
limitless_locks	pg_locks
limitless_stat_activity	pg_stat_activity
limitless_stat_all_indexes	pg_stat_all_indexes
limitless_stat_all_tables	pg_stat_all_tables
limitless_stat_database	pg_stat_database
limitless_stat_progress_vacuum	pg_stat_progress_vacuum
limitless_stat_statements	pg_stat_statements
limitless_stat_subclusters	Nenhum
limitless_stat_statements_info	pg_stat_statements_info
limitless_statio_all_indexes	pg_statio_all_indexes
limitless_statio_all_tables	pg_statio_all_tables
limitless_tables	pg_tables
limitless_table_collocations	Nenhum

Visualização do Aurora PostgreSQL Limitless Database	Visualização correspondente do Aurora PostgreSQL
limitless_table_collocation_distributions	Nenhum

Os exemplos a seguir fornecem detalhes sobre as visualizações do Aurora PostgreSQL Limitless Database. Para obter mais informações sobre visualizações do PostgreSQL, consulte Viewing statistics na documentação do PostgreSQL.

> **Note**
>
> Algumas visualizações de estatísticas podem retornar resultados inconsistentes se você tiver transações em andamento.

limitless_database

Essa visualização contém informações sobre os bancos de dados disponíveis no grupo de fragmentos de banco de dados. Por exemplo:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, oid, datname, datacl
 FROM rds_aurora.limitless_database;

 subcluster_id | subcluster_type |  oid  |      datname       |
                                 datacl

---------------+-----------------+-------+--------------------
+-------------------------------------------------------------------------------------------
 2             | router          |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 2             | router          |     5 | postgres           |
 2             | router          | 16384 | rdsadmin           |
 {rdsadmin=CTc/rdsadmin,rds_aurora_limitless_metadata_admin=c/
rdsadmin,rds_aurora_limitless_heat_mgmt_admin=c/rdsadmin}
 2             | router          | 16477 | postgres_limitless |
 2             | router          |     1 | template1          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 6             | shard           |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster (nó)
- `subcluster_type` (texto): o tipo de subcluster (nó), roteador ou fragmento

As demais colunas são as mesmas que em `pg_database`.

limitless_locks

Essa visualização contém uma linha por processo por nó. Ela fornece acesso às informações sobre os bloqueios mantidos pelos processos ativos no servidor do banco de dados.

Example de criação de um bloqueio com duas transações

Neste exemplo, executamos duas transações simultaneamente em dois roteadores.

```
# Transaction 1 (run on router 1)
BEGIN;
SET search_path = public;
SELECT * FROM customers;
INSERT INTO customers VALUES (400,'foo','bar');

# Transaction 2 (run on router 2)
BEGIN;
SET search_path = public;
ALTER TABLE customers ADD COLUMN phone VARCHAR;
```

A primeira transação é executada. As transações subsequentes precisam esperar até que a primeira transação seja concluída. Portanto, a segunda transação é bloqueada com um bloqueio. Para verificar a causa raiz disso, executamos um comando unindo `limitless_locks` com `limitless_stat_activity`.

```
# Run on router 2
SELECT distributed_session_id, state, usename, query, query_start
FROM rds_aurora.limitless_stat_activity
WHERE distributed_session_id in (
SELECT distributed_session_id
FROM rds_aurora.limitless_locks
WHERE relname = 'customers'
);

 distributed_session_id | state               | usename                | query
                                  | query_start
```

```
-----------------------+--------------------+-------------------------
+-------------------------------- -------------+-----------------------------
 47BDE66E9A5E8477       | idle in transaction | limitless_metadata_admin | INSERT INTO
 customers VALUES (400,'foo','bar'); | 2023-04-13 17:44:45.152244+00
 2AD7F370202D0FA9       | active              | limitless_metadata_admin | ALTER TABLE
 customers ADD COLUMN phone VARCHAR; | 2023-04-13 17:44:55.113388+00
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
(6 rows)
```

Example de criação de um bloqueio de forma explícita

Neste exemplo, criamos um bloqueio explicitamente e, em seguida, usamos a visualização `limitless_locks` para ver os bloqueios (algumas colunas são omitidas).

```
BEGIN;
SET search_path = public;
LOCK TABLE customers IN ACCESS SHARE MODE;
SELECT * FROM rds_aurora.limitless_locks WHERE relname = 'customers';

 subcluster_id | subcluster_type | distributed_session_id | locktype |      datname
   | relnspname |  relname  | virtualtransaction |  pid  |      mode
---------------+-----------------+------------------------+----------
+--------------------+------------+ ----------+--------------------+-------
+-----------------
             1 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600787          | 59564 |
 AccessShareLock
             2 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600405          | 67130 |
 AccessShareLock
             3 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 15/473401          | 27735 |
 AccessShareLock
             4 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473524          | 27734 |
 AccessShareLock
```

```
             5 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/472935          | 27737 |
 AccessShareLock
             6 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473015          | 48660 |
 AccessShareLock
(6 rows)
```

limitless_stat_activity

Essa visualização contém uma linha por processo por nó. Ela pode ser usada para monitorar a integridade geral do sistema e triar processos que estão demorando muito. Por exemplo:

```
postgres=# SELECT
    subcluster_id,
    subcluster_type,
    distributed_session_id,
    distributed_session_state,
    datname,
    distributed_query_id,
    is_sso_query
FROM
    rds_aurora.limitless_stat_activity
WHERE
    distributed_session_id in ('D2470C97E3D07E06', '5A3CD7B8E5FD13FF')
    order by  distributed_session_id;

 subcluster_id | subcluster_type | distributed_session_id | distributed_session_state |
      datname       | distributed_query_id | is_sso_query
---------------+-----------------+------------------------+---------------------------
+--------------------+----------------------+-------------
 2             | router          | 5A3CD7B8E5FD13FF       | coordinator               |
 postgres_limitless |                      | f
 3             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 4             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 2             | router          | D2470C97E3D07E06       | coordinator               |
 postgres_limitless |                      | t
 3             | shard           | D2470C97E3D07E06       | participant               |
 postgres_limitless |  4058400544464210222 |
(5 rows)
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `distributed_session_id` (texto): o ID da sessão distribuída à qual esse processo pertence.
- `distributed_session_state` (texto): se este é um coordenador, participante ou processo autônomo/não distribuído (mostrado como `NULL`).
- `datname` (texto): o banco de dados ao qual esse processo está conectado.
- `distributed_query_id` (bigint): o ID da consulta principal do nó coordenador. Essa coluna é `NULL` se for a consulta principal. O nó coordenador envia o ID da consulta distribuída para os nós participantes. Portanto, para os nós participantes, os valores para o ID de consulta distribuída e o ID de consulta são diferentes.
- `is_sso_query` (texto) — isso nos permite saber se a consulta está otimizada para um único fragmento ou não.

As demais colunas são as mesmas que em `pg_stat_activity`.

limitless_stat_all_indexes

Essa visualização contém estatísticas de uso em índices no grupo de fragmentos de banco de dados. Por exemplo:

```
postgres_limitless=> SELECT schemaname, relname, indexrelname, idx_scan
  FROM rds_aurora.limitless_stat_all_indexes
  WHERE relname LIKE 'orders_ts%' ORDER BY indexrelname LIMIT 10;

 schemaname |    relname     |    indexrelname     | idx_scan
------------+----------------+---------------------+----------
 ec_sample  | orders_ts00001 | orders_ts00001_pkey |   196801
 ec_sample  | orders_ts00002 | orders_ts00002_pkey |   196703
 ec_sample  | orders_ts00003 | orders_ts00003_pkey |   196376
 ec_sample  | orders_ts00004 | orders_ts00004_pkey |   197966
 ec_sample  | orders_ts00005 | orders_ts00005_pkey |   195301
 ec_sample  | orders_ts00006 | orders_ts00006_pkey |   195673
 ec_sample  | orders_ts00007 | orders_ts00007_pkey |   194475
 ec_sample  | orders_ts00008 | orders_ts00008_pkey |   191694
 ec_sample  | orders_ts00009 | orders_ts00009_pkey |   193744
 ec_sample  | orders_ts00010 | orders_ts00010_pkey |   195421
```

```
(10 rows)
```

limitless_stat_all_tables

Essa visualização contém estatísticas sobre todas as tabelas no banco de dados atual no grupo de fragmentos de banco de dados. Isso é útil ao rastrear operações de limpeza e operações de Linguagem de Manipulação de Dados (DML). Por exemplo:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, relname,
 n_ins_since_vacuum, n_tup_ins, last_vacuum
  FROM rds_aurora.limitless_stat_all_tables
  WHERE relname LIKE 'orders_ts%' ORDER BY relname LIMIT 10;

 subcluster_id | subcluster_type |    relname     | n_ins_since_vacuum | n_tup_ins |
 last_vacuum
---------------+-----------------+----------------+--------------------+-----------
+-------------
 5             | shard           | orders_ts00001 |              34779 |    196083 |
 5             | shard           | orders_ts00002 |              34632 |    194721 |
 5             | shard           | orders_ts00003 |              34950 |    195965 |
 5             | shard           | orders_ts00004 |              34745 |    197283 |
 5             | shard           | orders_ts00005 |              34879 |    195754 |
 5             | shard           | orders_ts00006 |              34340 |    194605 |
 5             | shard           | orders_ts00007 |              33779 |    192203 |
 5             | shard           | orders_ts00008 |              33826 |    191293 |
 5             | shard           | orders_ts00009 |              34660 |    194117 |
 5             | shard           | orders_ts00010 |              34569 |    195560 |
(10 rows)
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `relname` (nome): o nome da tabela.

As demais colunas são as mesmas que em `pg_stat_all_tables`.

limitless_stat_database

Essa visualização contém estatísticas sobre todos os bancos de dados no grupo de fragmentos de banco de dados. Retorna uma linha por banco de dados por nó. Por exemplo:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    datname,
    blks_read,
    blks_hit
FROM
    rds_aurora.limitless_stat_database
WHERE
    datname='postgres_limitless';
 subcluster_id | subcluster_type |      datname       | blks_read | blks_hit
---------------+-----------------+--------------------+-----------+----------
             1 | router          | postgres_limitless |       484 | 34371314
             2 | router          | postgres_limitless |       673 | 33859317
             3 | shard           | postgres_limitless |      1299 | 17749550
             4 | shard           | postgres_limitless |      1094 | 17492849
             5 | shard           | postgres_limitless |      1036 | 17485098
             6 | shard           | postgres_limitless |      1040 | 17437257
(6 rows)
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `datname` (nome): o nome do banco de dados.

As demais colunas são as mesmas que em `pg_stat_database`.

limitless_stat_progress_vacuum

Essa visualização contém informações sobre as operações de limpeza em andamento. Por exemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;

-[ RECORD 1 ]----------+-----------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | A56D96E2A5C9F426
pid                    | 5270
datname                | postgres
```

```
nspname                | public
relname                | customer_ts2
phase                  | vacuuming heap
heap_blks_total        | 130500
heap_blks_scanned      | 100036
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 11184810
num_dead_tuples        | 0

-[ RECORD 2 ]----------+-----------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | 56DF26A89EC23AB5
pid                    | 6854
datname                | postgres
nspname                | public
relname                | sales_ts1
phase                  | vacuuming heap
heap_blks_total        | 43058
heap_blks_scanned      | 24868
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 8569523
num_dead_tuples        | 0
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` ou `shard`.
- `distributed_session_id` (texto): o identificador da sessão que iniciou a operação de limpeza.
- `datname` (nome): o banco de dados em que a limpeza está sendo feita.
- `nspname` (nome): o nome do esquema da tabela que está sendo limpa. É `null` se a tabela que está sendo limpa não estiver no mesmo banco de dados ao qual o usuário está conectado.
- `relname` (nome): o nome da tabela que está sendo limpa. É `null` se a tabela que está sendo limpa não estiver no mesmo banco de dados ao qual o usuário está conectado.

As demais colunas são as mesmas que em `pg_stat_progress_vacuum`.

limitless_stat_statements

Essa visualização fornece um meio de rastrear o planejamento e a execução de estatísticas de todas as instruções SQL executadas em todos os nós.

> **Note**
>
> É necessário instalar a extensão pg_stat_statements para usar a visualização `limitless_stat_statements`.
>
> ```
> -- CREATE EXTENSION must be run by a superuser
> CREATE EXTENSION pg_stat_statements;
>
> -- Verify that the extension is created on all nodes in the DB shard group
> SELECT distinct node_id
>     FROM rds_aurora.limitless_stat_statements
>     LIMIT 10;
> ```

O exemplo a seguir mostra o uso da visualização `limitless_stat_statements`.

```
postgres_limitless=> SELECT
 subcluster_id,
 subcluster_type,
 distributedqueryid,
 username,
 dbname,
 sso_calls
FROM
 rds_aurora.limitless_stat_statements;

 subcluster_id | subcluster_type |  distributedqueryid  |              username
       |       dbname       | sso_calls
---------------+-----------------+----------------------
+-------------------------------------+--------------------+-----------
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
```

```
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         1
 3             | shard           | -7975178695405682176 | postgres
       | postgres_limitless |
[...]
```

Os parâmetros de saída são os seguintes:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.
- `subcluster_type` (texto): o tipo de subcluster ao qual esse processo pertence: `router` para ou `shard`.
- `distributedqueryid` (bigint): o ID da consulta principal do nó coordenador. Essa coluna é `NULL` se for a consulta principal. O nó coordenador envia o ID da consulta distribuída para os nós participantes. Portanto, para os nós participantes, os valores para o ID de consulta distribuída e o ID de consulta são diferentes.
- `username` (nome): o usuário que consultou a instrução.
- `dbname` (nome): o banco de dados onde a consulta foi executada.
- `sso_calls` (nome) — o número de vezes que a instrução foi otimizada para um único fragmento.

As demais colunas são as mesmas que em pg_stat_statements.

limitless_stat_statements_info

Essa visualização contém estatísticas para a visualização `limitless_stat_statements`. Cada linha contém dados para a visualização pg_stat_statements_info de cada nó. A coluna `subcluster_id` identifica cada nó.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_statements_info;

 subcluster_id | subcluster_type | dealloc |          stats_reset
---------------+-----------------+---------+-------------------------------
             1 | router          |       0 | 2023-06-30 21:22:09.524781+00
             2 | router          |       0 | 2023-06-30 21:21:40.834111+00
             3 | shard           |       0 | 2023-06-30 21:22:10.709942+00
             4 | shard           |       0 | 2023-06-30 21:22:10.740179+00
```

```
             5 | shard           |       0 | 2023-06-30 21:22:10.774282+00
             6 | shard           |       0 | 2023-06-30 21:22:10.808267+00
(6 rows)
```

O parâmetro de saída é o seguinte:

- `subcluster_id` (texto): o ID do subcluster ao qual esse processo pertence.

As demais colunas são as mesmas que em pg_stat_statements_info.

limitless_stat_subclusters

Essa visualização contém estatísticas de rede entre roteadores e outros nós. Ela contém uma linha por par de roteador e outro nó, por exemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_subclusters;

 orig_subcluster | orig_instance_az | dest_subcluster | dest_instance_az |
 latency_us |       latest_collection       | failed_requests | received_bytes |
 sent_bytes | same_az_requests | cross_az_requests |     stat_reset_timestamp
-----------------+------------------+-----------------+------------------
+------------+-------------------------------+-----------------+---------------
+------------+------------------+-------------------+------------------------------
 3               | us-west-2b       | 2               | us-west-2a       |
 847 | 2024-10-07 17:25:39.518617+00 |               0 |       35668633 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.239675+00
 3               | us-west-2b       | 4               | us-west-2b       |
 419 | 2024-10-07 17:25:39.546376+00 |               0 |      101190464 |  248795719
 |           883478 |                0 | 2024-10-05 12:39:55.231218+00
 3               | us-west-2b       | 5               | us-west-2c       |
    1396 | 2024-10-07 17:25:39.52122+00  |               0 |       72864849 |
 172086292 |               0 |           557726 | 2024-10-05 12:39:55.196412+00
 3               | us-west-2b       | 6               | us-west-2c       |
 729 | 2024-10-07 17:25:39.54828+00  |               0 |       35668584 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.247334+00
 3               | us-west-2b       | 7               | us-west-2a       |
    1702 | 2024-10-07 17:25:39.545307+00 |               0 |       71699576 |
 171634844 |               0 |           556278 | 2024-10-05 12:39:52.715168+00
 2               | us-west-2a       | 3               | us-west-2b       |
 868 | 2024-10-07 17:25:40.293927+00 |               0 |       35659611 |   92011872
 |                0 |           302817 | 2024-10-05 12:39:54.420758+00
 2               | us-west-2a       | 4               | us-west-2b       |
 786 | 2024-10-07 17:25:40.296863+00 |               0 |      102437253 |  251838024
 |                0 |           895060 | 2024-10-05 12:39:54.404081+00
```

```
2               | us-west-2a       | 5               | us-west-2c       |
   1232 | 2024-10-07 17:25:40.292021+00 |               0 |       71990027 |
168828110 |               0 |            545453 | 2024-10-05 12:39:36.769549+00
```

Os parâmetros de saída são os seguintes:

- `orig_subcluster` (texto): o ID do roteador de onde as comunicações se originam
- `orig_subcluster_az` (texto): a zona de disponibilidade (AZ) do roteador de origem
- `dest_subcluster` (texto): o ID do nó de destino
- `dest_subcluster_az` (texto): a última AZ coletada do nó de destino
- `latency_us` (bigint): a última latência de rede coletada entre os nós, em microssegundos. O valor é `0` se o nó estiver inacessível.
- `latest_collection` (data e hora): a data e hora da coleta mais recente de AZ e latência para o nó de destino
- `failed_requests` (bigint): a contagem cumulativa de solicitações internas com falha
- `received_bytes` (bigint): o número cumulativo estimado de bytes recebidos desse nó
- `sent_bytes` (bigint): o número cumulativo estimado de bytes enviados para esse nó
- `same_az_requests` (bigint): o número cumulativo de solicitações internas de banco de dados para esse nó quando ele está na mesma AZ do roteador de origem
- `cross_az_requests` (bigint): o número cumulativo de solicitações internas de banco de dados para esse nó quando ele está em uma AZ diferente do roteador de origem
- `stat_reset_timestamp` (data e hora): a data e hora de quando as estatísticas cumulativas dessa visualização foram redefinidas pela última vez

limitless_statio_all_indexes

Essa visualização contém estatísticas de entrada/saída (E/S) de todos os índices do grupo de fragmentos de banco de dados. Por exemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_statio_all_indexes WHERE
 relname like'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      |
 indexrelname             | idx_blks_read | idx_blks_hit
---------------+-----------------+------------+-------------------
+------------------------------------+ --------------+--------------
             3 | shard           | public     | customers_ts00002 |
 customers_ts00002_customer_name_idx |             1 |            0
```

```
             3 | shard           | public     | customers_ts00001 |
 customers_ts00001_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00003 |
 customers_ts00003_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00004 |
 customers_ts00004_customer_name_idx |             1 |           0
             5 | shard           | public     | customers_ts00005 |
 customers_ts00005_customer_name_idx |             1 |           0
             5 | shard           | public     | customers_ts00006 |
 customers_ts00006_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00007 |
 customers_ts00007_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00008 |
 customers_ts00008_customer_name_idx |             1 |           0
(8 rows)
```

limitless_statio_all_tables

Essa visualização contém estatísticas de entrada/saída (E/S) de todas as tabelas do grupo de fragmentos de banco de dados. Por exemplo:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    schemaname,
    relname,
    heap_blks_read,
    heap_blks_hit
FROM
    rds_aurora.limitless_statio_all_tables
WHERE
    relname LIKE 'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      | heap_blks_read |
 heap_blks_hit
---------------+-----------------+------------+-------------------+----------------
+---------------
             3 | shard           | public     | customers_ts00002 |            305 |
      57780
             3 | shard           | public     | customers_ts00001 |            300 |
      56972
             4 | shard           | public     | customers_ts00004 |            302 |
      57291
```

```
             4 | shard           | public     | customers_ts00003 |            302 |
      57178
             5 | shard           | public     | customers_ts00006 |            300 |
      56932
             5 | shard           | public     | customers_ts00005 |            302 |
      57386
             6 | shard           | public     | customers_ts00008 |            300 |
      56881
             6 | shard           | public     | customers_ts00007 |            304 |
      57635
(8 rows)
```

limitless_tables

Essa visualização contém informações sobre tabelas no Aurora PostgreSQL Limitless Database.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | placeholder | active       | placeholder |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     | HASH
 (order_id)
         2 |     18579 | public      | customer    | active       | sharded     | HASH
 (cust_id)
(4 rows)
```

limitless_table_collocations

Essa visualização contém informações sobre tabelas fragmentadas colocalizadas.

No exemplo a seguir, as tabelas `orders` e `customers` são colocalizadas, e as tabelas `users` e `followers` são colocalizadas. As tabelas colocalizadas têm o mesmo `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
              2 | public      | orders
```

```
             2 | public      | customers
             5 | public      | users
             5 | public      | followers
(4 rows)
```

limitless_table_collocation_distributions

Essa visualização mostra a distribuição de chaves para cada colocação.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
             2 |             6 | -9223372036854775808 | -4611686018427387904
             2 |             5 | -4611686018427387904 |                    0
             2 |             4 |                    0 |  4611686018427387904
             2 |             3 |  4611686018427387904 |  9223372036854775807
             5 |             6 | -9223372036854775808 | -4611686018427387904
             5 |             5 | -4611686018427387904 |                    0
             5 |             4 |                    0 |  4611686018427387904
             5 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Eventos de espera do Aurora PostgreSQL Limitless Database

Um evento de espera no Aurora PostgreSQL indica um recurso pelo qual uma sessão está aguardando, como entrada/saída (E/S) e bloqueios. Os eventos de espera são úteis para descobrir por que as sessões estão aguardando recursos e identificar gargalos. Para ter mais informações, consulte Eventos de espera do Aurora PostgreSQL.

O Aurora PostgreSQL Limitless Database tem seus próprios eventos de espera relacionados a roteadores e fragmentos. Muitos deles são para roteadores que aguardam fragmentos para concluir tarefas. Os eventos de espera de fragmento contêm detalhes sobre as tarefas que estão sendo realizadas.

Tópicos

- Consulta de eventos de espera
- Eventos de espera do Limitless Database

Consulta de eventos de espera

Você pode usar a visualização limitless_stat_activity para consultar eventos de espera, conforme mostrado no exemplo a seguir.

```
SELECT wait_event FROM rds_aurora.limitless_stat_activity WHERE
 wait_event_type='AuroraLimitless';

      wait_event
----------------------
 RemoteStatementSetup
 RemoteStatementSetup
(2 rows)
```

Você também pode usar a função `aurora_stat_system_waits` para listar o número de esperas e o tempo total gasto em cada evento de espera, conforme mostrado no exemplo a seguir.

```
postgres_limitless=> SELECT type_name,event_name,waits,wait_time
    FROM aurora_stat_system_waits()
    NATURAL JOIN aurora_stat_wait_event()
    NATURAL JOIN aurora_stat_wait_type()
    WHERE type_name='AuroraLimitless'
    ORDER BY wait_time DESC;
```

```
    type_name    |       event_name          |  waits  |  wait_time
-----------------+---------------------------+---------+-------------
 AuroraLimitless | RemoteStatementSetup      |    7518 | 75236507897
 AuroraLimitless | RemoteStatementExecution  |      40 |      132986
 AuroraLimitless | Connect                   |       5 |        1453
(3 rows)
```

Eventos de espera do Limitless Database

Os eventos de espera a seguir se aplicam ao Aurora PostgreSQL Limitless Database. Você pode monitorar esses eventos de espera para identificar gargalos no processamento do Aurora PostgreSQL Limitless Database.

Tópicos

- Evento de espera IO:TwophaseFilePoolWrite
- Evento de espera IO:TwophaseFilePoolRead
- Evento de espera AuroraLimitless
- Evento de espera AuroraLimitless:AsyncConnect
- Evento de espera AuroraLimitless:RemoteStatementSetup
- Evento de espera AuroraLimitless:RemoteDDLExecution
- Evento de espera AuroraLimitless:RemoteStatementExecution
- Evento de espera AuroraLimitless:FetchRemoteResults
- Evento de espera AuroraLimitless:AsyncGetInitialResponse
- Evento de espera AuroraLimitless:AsyncGetNextResponse
- Evento de espera AuroraLimitless:AbortedCommandCleanup
- Evento de espera AuroraLimitless:DistributedCommitPrepare
- Evento de espera AuroraLimitless:DistributedCommit
- Evento de espera AuroraLimitless:DistributedCommitPrepareThrottle
- Evento de espera AuroraLimitless:PreparedTransactionResolution
- Evento de espera AuroraLimitless:SendPreparedTransactionOutcome
- Evento de espera AuroraLimitless:CommitClockBarrier
- Evento de espera AuroraLimitless:SnapshotClockBarrier

- Evento de espera AuroraLimitless:ReaderSnapshotClockBarrier
- Evento de espera AuroraLimitless:GatherDistributedDeadlockGraph
- Evento de espera AuroraLimitless:DistributedDeadlockDetection
- Evento de espera AuroraLimitless:DistributedDeadlockAbort
- Evento de espera AuroraLimitless:GatherRemoteStats
- Evento de espera AuroraLimitless:GlobalSequenceRefresh
- Evento de espera AuroraLimitless:GlobalVacuumTimeExchange
- Evento de espera AuroraLimitless:DistributedTransactionMonitorGather
- Evento de espera AuroraLimitlessActivity:AdminTaskSchedulerMain
- Evento de espera AuroraLimitlessActivity:AdminTaskExecutorMain
- Evento de espera AuroraLimitlessActivity:AdminTaskMonitorMain
- Evento de espera AuroraLimitlessActivity:DatabaseCleanupMonitorMain
- Evento de espera AuroraLimitlessActivity:TopologyCleanupMonitorMain
- Evento de espera AuroraLimitlessActivity:ToplogyChangeMonitorMain
- Evento de espera AuroraLimitlessActivity:DistributedTransactionMonitorMain
- Evento de espera AuroraLimitlessActivity:GlobalVacuumMonitorMain

Evento de espera IO:TwophaseFilePoolWrite

Esperar por uma gravação de um arquivo de estado de duas fases dentro do grupo de arquivos de estado de duas fases. Este é um evento específico do Aurora.

Causas

Os processos que executam um comando `PREPARED TRANSACTION`, incluindo participantes em uma transação distribuída do Limitless Database, devem persistir o estado da transação em um arquivo de duas fases. O Aurora usa um grupo de arquivos para melhorar o desempenho dessa operação.

Ação

Essa é uma operação de E/S de gravação síncrona e, portanto, uma alta latência nesse evento tem causas semelhantes a `IO:XactSync` e pode ser investigada da mesma forma. Se estiver

usando o Limitless Database, talvez seja necessário reduzir o número de transações distribuídas em execução.

Evento de espera IO:TwophaseFilePoolRead

Esperar pela leitura de um arquivo de estado de duas fases dentro do grupo de arquivos de estado de duas fases.

Causas

Os processos que executam um comando `COMMIT PREPARED` em uma transação previamente preparada, incluindo participantes de uma transação distribuída do Limitless Database, talvez precisem ler o estado da transação anteriormente persistido em um arquivo de duas fases. O Aurora usa um grupo de arquivos para melhorar o desempenho dessa operação.

Ação

Esta é uma operação de E/S de leitura. Portanto, uma alta latência nesse evento tem causas semelhantes às de `IO:DataFileRead` e pode ser investigada da mesma forma. Se estiver usando o Limitless Database, talvez seja necessário reduzir o número de transações distribuídas em execução.

Evento de espera AuroraLimitless

O processo está esperando o estabelecimento de uma conexão com outro nó no cluster.

Causas

As conexões são estabelecidas entre processos e nós remotos para executar consultas, transações distribuídas e realizar DDLs.

Ação

Reduza o número de conexões simultâneas ao cluster ou ajuste o uso de consultas entre fragmentos.

Evento de espera AuroraLimitless:AsyncConnect

Esse evento é semelhante a `Connect`, mas representa um processo que espera o estabelecimento de conexões paralelas com um conjunto de nós.

Causas

O estabelecimento de conexão paralela é mais comumente feito ao executar instruções DDL.

Ação

Reduza o número de instruções DDL ou combine várias DDLs na mesma sessão para melhorar a reutilização da conexão.

Evento de espera AuroraLimitless:RemoteStatementSetup

O processo está esperando a configuração da execução da consulta remota, como abertura, fechamento do cursor ou criação de uma instrução preparada.

Causas

Esse evento de espera aumenta com o número de varreduras em tabelas fragmentadas em que a instrução não pôde ser otimizada para um único fragmento.

Ação

Otimize as consultas para reduzir o número de operações de varredura ou aumentar a elegibilidade para a otimização de um único fragmento.

Evento de espera AuroraLimitless:RemoteDDLExecution

O processo está esperando a conclusão de um comando DDL remoto.

Causas

Ao emitir um comando DDL em um grupo de fragmentos de banco de dados, ele deve ser distribuído para outros roteadores e nós de fragmento antes de confirmar a operação. Algumas operações de DDL podem ser executadas por um longo período, porque os dados devem ser adaptados às alterações do esquema.

Ação

Identifique comandos DDL de longa duração para poder otimizá-los.

Evento de espera AuroraLimitless:RemoteStatementExecution

Um processo está esperando a conclusão de um comando remoto.

Causas

Um comando SQL está sendo executado em um nó remoto. Esse evento aparecerá com frequência em comunicações internas, como verificações de `auto_analyze` e de pulsação.

Ação

Identifique comandos de longa execução usando a exibição limitless_stat_statements. Em muitos casos, esse é um evento esperado, principalmente em processos em segundo plano ou processos internos, e nenhuma ação é necessária.

Evento de espera AuroraLimitless:FetchRemoteResults

Um processo está esperando para recuperar linhas de um nó remoto.

Causas

Esse evento de espera pode aumentar ao buscar muitas linhas de uma tabela remota, como uma tabela fragmentada ou de referência.

Ação

Identifique consultas `SELECT` não otimizadas usando a exibição `limitless_stat_statements`. Otimize as consultas para recuperar somente os dados necessários. Também é possível ajustar o parâmetro `rds_aurora.limitless_maximum_adaptive_fetch_size`.

Evento de espera AuroraLimitless:AsyncGetInitialResponse

O processo está esperando uma resposta inicial quando o modo pipeline é usado na execução da consulta.

Causas

Isso normalmente será observado durante a execução do roteador para o fragmento em consultas com posicionamento de dados em um único fragmento e é uma parte esperada da execução normal.

Ação

Nenhuma outra ação é necessária.

Evento de espera AuroraLimitless:AsyncGetNextResponse

O processo está esperando respostas adicionais quando o modo pipeline é usado na execução da consulta.

Causas

Isso normalmente será observado durante a execução do roteador para o fragmento em consultas com posicionamento de dados em um único fragmento e é uma parte esperada da execução normal.

Ação

Nenhuma outra ação é necessária.

Evento de espera AuroraLimitless:AbortedCommandCleanup

O processo está esperando o resultado de uma consulta de limpeza remota. As consultas de limpeza são emitidas para os nós de fragmento para retorná-los ao estado apropriado quando uma transação distribuída é encerrada.

Causas

A limpeza de transação é feita quando uma transação é abortada porque um erro foi encontrado ou porque um usuário emitiu um comando ABORT explícito ou cancelou a consulta em execução.

Ação

Investigue a causa do cancelamento da transação.

Evento de espera AuroraLimitless:DistributedCommitPrepare

O processo está confirmando uma transação distribuída e esperando que todos os participantes reconheçam o comando de preparo.

Causas

As transações que modificam vários nós devem realizar uma confirmação distribuída. Uma longa espera em `DistributedCommitPrepare` pode ser causada por longas esperas no evento `IO:TwophaseFilePoolWrite` nos nós participantes.

Ação

Reduza o número de transações que modificam dados em vários nós. Investigue eventos `IO:TwophaseFilePoolWrite` em outros nós do cluster.

Evento de espera AuroraLimitless:DistributedCommit

O processo está confirmando uma transação distribuída e esperando que o participante principal reconheça o comando de confirmação.

Causas

As transações que modificam vários nós devem realizar uma confirmação distribuída. Uma longa espera em `DistributedCommit` pode ser causada por longas esperas no evento `IO:XactSync` no participante principal.

Ação

Reduza o número de transações que modificam dados em vários nós. Investigue eventos `IO:XactSync` em outros nós do cluster.

Evento de espera AuroraLimitless:DistributedCommitPrepareThrottle

O processo está tentando preparar uma transação distribuída e está com controle de utilização devido às transações preparadas existentes.

Causas

As transações que modificam vários nós devem realizar uma confirmação distribuída. Os participantes dessas transações devem realizar uma operação de preparação como parte do protocolo de confirmação. O Aurora limita o número de preparações simultâneas. Se esse limite for excedido, o processo aguardará no evento `DistributedCommitPrepareThrottle`.

Ação

Reduza o número de transações que modificam dados em vários nós. Investigue os eventos `IO:TwophaseFilePoolWrite`, pois o aumento do tempo nesses eventos pode fazer com que as transações preparadas existentes se acumulem, resultando em controle de utilização de novas tentativas de preparação.

Evento de espera AuroraLimitless:PreparedTransactionResolution

O processo encontrou uma tupla modificada por uma transação distribuída que está no estado de preparo. O processo deve determinar se a transação distribuída ficará visível no snapshot.

Causas

As transações que modificam vários nós devem realizar uma confirmação distribuída que inclui uma fase de preparo. Um alto número de transações distribuídas ou o aumento da latência em confirmações distribuídas podem fazer com que outros processos encontrem o evento de espera `PreparedTransactionResolution`.

Ação

Reduza o número de transações que modificam dados em vários nós. Investigue eventos relacionados a confirmações distribuídas, pois o aumento do tempo nesses eventos pode aumentar a latência no caminho de confirmação das transações distribuídas. Também investigue as cargas da rede e da CPU.

Evento de espera AuroraLimitless:SendPreparedTransactionOutcome

O processo está sendo executado em um nó que está coordenando uma transação distribuída e outro processo perguntou sobre o estado dessa transação, ou o processo confirmou uma transação distribuída e está enviando o resultado aos participantes.

Causas

Os processos que encontrarem o evento de espera `PreparedTransactionResolution` consultarão o coordenador da transação. A resposta no coordenador da transação encontrará SendPreparedTransactionOutcome.

Ação

Reduza o número de transações que modificam dados em vários nós. Investigue eventos relacionados à confirmação distribuída e eventos `IO:TwophaseFilePoolWrite` e `IO:TwophaseFilePoolRead`, pois esses eventos podem aumentar a latência no caminho de confirmação das transações distribuídas. Também investigue as cargas da rede e da CPU.

Evento de espera AuroraLimitless:CommitClockBarrier

O processo está confirmando uma transação e deve esperar para garantir que o tempo de confirmação atribuído seja garantidamente passado para todos os nós do cluster.

Causas

A saturação da CPU ou da rede pode causar um aumento no desvio do relógio, resultando no tempo gasto nesse evento de espera.

Ação

Investigue a saturação da CPU ou da rede no cluster.

Evento de espera AuroraLimitless:SnapshotClockBarrier

O processo recebeu um horário de snapshot de outro nó com um relógio no futuro e está esperando que seu próprio relógio atinja esse horário.

Causas

Isso geralmente ocorre depois que o processo recebe resultados de uma função que foi enviada para um fragmento e há um desvio de relógio entre os nós. A saturação da CPU ou da rede pode causar um aumento no desvio do relógio, resultando no tempo gasto nesse evento de espera.

Ação

Investigue a saturação da CPU ou da rede no cluster.

Evento de espera AuroraLimitless:ReaderSnapshotClockBarrier

Esse evento ocorre nos nós de leitura. O processo está esperando que o nó de leitura reproduza o fluxo de gravação para que todas as gravações que ocorreram antes do horário do snapshot do processo sejam aplicadas.

Causas

Um aumento no atraso da réplica do Aurora pode causar um aumento no tempo de espera nesse evento.

Ação

Investigue o atraso da réplica do Aurora.

Evento de espera AuroraLimitless:GatherDistributedDeadlockGraph

O processo está se comunicando com outros nós para coletar gráficos de bloqueio como parte da detecção de impasse distribuído.

Causas

Quando um processo está esperando um bloqueio, ele executa uma verificação de impasse distribuído após esperar mais de `rds_aurora.limitless_distributed_deadlock_timeout`.

Ação

Investigue as causas da contenção de bloqueios na aplicação e considere ajustar o `rds_aurora.limitless_distributed_deadlock_timeout`.

Evento de espera AuroraLimitless:DistributedDeadlockDetection

O processo está se comunicando com outros nós para detectar um impasse distribuído.

Causas

Quando um processo está esperando um bloqueio, ele executa uma verificação de impasse distribuído após esperar mais de `rds_aurora.limitless_distributed_deadlock_timeout`.

Ação

Investigue as causas da contenção de bloqueios na aplicação e considere ajustar o `rds_aurora.limitless_distributed_deadlock_timeout`.

Evento de espera AuroraLimitless:DistributedDeadlockAbort

O processo está se comunicando com outro nó para abortar uma sessão escolhida como vítima em um impasse distribuído.

Causas

Os padrões de aplicações estão resultando em impasses distribuídos.

Ação

Investigue os padrões de aplicações que resultam em impasses distribuídos.

Evento de espera AuroraLimitless:GatherRemoteStats

O processo está coletando estatísticas de outros nós no cluster.

Causas

As consultas e visualizações de monitoramento ou atividade, como `limitless_stat_activity`, recuperarão estatísticas de outros nós.

Ação

Nenhuma outra ação é necessária.

Evento de espera AuroraLimitless:GlobalSequenceRefresh

O processo está gerando um novo valor de sequência e deve solicitar uma nova parte da sequência global.

Causas

Uma alta taxa de geração de valor de sequência pode resultar em paralisações nesse evento se `rds_aurora.limitless_sequence_chunk_size` for insuficiente.

Ação

Essa é uma ocorrência normal. Se você observar um tempo excessivo nesse evento, considere ajustar o `rds_aurora.limitless_sequence_chunk_size`. Consulte a documentação sobre sequências no Limitless Database.

Evento de espera AuroraLimitless:GlobalVacuumTimeExchange

O processo está trocando dados de snapshot para oferecer suporte à limpeza.

Causas

Os nós no Limitless Database trocam dados de horário do snapshot ativo mais antigo com outros nós para calcular o tempo de corte correto para a execução da limpeza.

Ação

Nenhuma outra ação é necessária.

Evento de espera AuroraLimitless:DistributedTransactionMonitorGather

O processo está reunindo metadados de transações de outros nós para oferecer suporte à limpeza de transações distribuídas.

Causas

Os nós no Limitless Database trocam metadados de transações com outros nós para determinar quando o estado de transação distribuída pode ser eliminado.

Ação

Nenhuma outra ação é necessária.

Evento de espera AuroraLimitlessActivity:AdminTaskSchedulerMain

Esperando no loop principal do processo de agendamento de tarefas.

Evento de espera AuroraLimitlessActivity:AdminTaskExecutorMain

Esperando no loop principal do processo do executor de tarefas.

Evento de espera AuroraLimitlessActivity:AdminTaskMonitorMain

Esperando no loop principal do processo de monitoramento de tarefas.

Evento de espera AuroraLimitlessActivity:DatabaseCleanupMonitorMain

Esperando no loop principal do processo de monitoramento de limpeza do banco de dados.

Evento de espera AuroraLimitlessActivity:TopologyCleanupMonitorMain

Esperando no loop principal do processo de monitoramento de limpeza de topologia.

Evento de espera AuroraLimitlessActivity:ToplogyChangeMonitorMain

Esperando no loop principal do processo de monitoramento de alterações de topologia.

Evento de espera AuroraLimitlessActivity:DistributedTransactionMonitorMain

Esperando no loop principal do processo de monitoramento de transações distribuídas.

Evento de espera AuroraLimitlessActivity:GlobalVacuumMonitorMain

Esperando no loop principal do processo de monitoramento de limpeza global.

Usar funções para aumentar a eficiência

Por padrão, as funções definidas pelo usuário não são otimizadas para um único fragmento, mas podem ser configuradas para serem executadas como operações de fragmento único. As funções podem encapsular a lógica e garantir que ela seja executada de maneira otimizada para um único fragmento.

Por que as operações de fragmento único são importantes

A utilização de recursos é importante para o desempenho e o custo-benefício. As operações de fragmento único usam significativamente menos recursos em comparação com as operações entre fragmentos. Por exemplo, ao executar uma função para inserir 1 milhão de linhas, a execução de um único fragmento usa aproximadamente 90,5 ACUs, enquanto a execução entre fragmentos usa 126,5 ACUs, uma melhoria de 35% na eficiência dos recursos.

A execução de um único fragmento também oferece:

- Throughput 35% mais alto que as operações entre fragmentos.
- Tempos de resposta mais previsíveis.
- Melhor escalabilidade à medida que os dados crescem.

Operações e funções de fragmento único

As funções são executadas em fragmentos quando um destes pré-requisitos é atendido:

- A função é criada como imutável e incluída em uma consulta otimizada para um único fragmento.
- A função é distribuída por um usuário.

As funções executadas em fragmentos têm melhor desempenho e escalam melhor porque são executadas onde os dados estão localizados.

Funções e volatilidade

Para verificar a volatilidade de uma função, use esta consulta em tabelas de sistema do PostgreSQL:

```
SELECT DISTINCT nspname, proname, provolatile
FROM pg_proc PRO
JOIN pg_namespace NSP ON PRO.pronamespace = NSP.oid
```

```
WHERE proname IN ('random', 'md5');
```

Resultado do exemplo:

```
  nspname   | proname | provolatile
------------+---------+-------------
 pg_catalog | md5     | i
 pg_catalog | random  | v
(2 rows)
```

Neste exemplo, `md5()` é imutável e `random()` é volátil. Isso significa que uma instrução otimizada para um único fragmento que inclua `md5()` permanece otimizada para um único fragmento, ao contrário de uma instrução que inclua `random()`.

Exemplo com função imutável:

```
EXPLAIN ANALYZE
SELECT pg_catalog.md5('123')
FROM s1.t1
WHERE col_a = 776586194
  AND col_b = 654849524
  AND col_c = '3ac2f2affb02987159ccd6ebd23e1ae5';
```

```
                          QUERY PLAN
---------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
               (actual time=3.409..3.409 rows=1 loops=1)
 Single Shard Optimized
 Planning Time: 0.313 ms
 Execution Time: 4.253 ms
(4 rows)
```

Exemplo com função volátil:

```
EXPLAIN ANALYZE
SELECT pg_catalog.random()
FROM s1.t1
WHERE col_a = 776586194
  AND col_b = 654849524
  AND col_c = '3ac2f2affb02987159ccd6ebd23e1ae5';
```

```
                          QUERY PLAN
-----------------------------------------------------
 Foreign Scan on t1_fs00001 t1
   (cost=100.00..15905.15 rows=1 width=8)
   (actual time=0.658..0.658 rows=1 loops=1)
 Planning Time: 0.263 ms
 Execution Time: 2.892 ms
(3 rows)
```

A saída mostra que, na função `md5()`, aplica-se pushdown para executá-la como otimizada para um único fragmento, ao contrário de `random()`.

Funções de distribuição

Uma função que acessa dados em apenas um fragmento deve ser executada nesse fragmento para obter benefícios de desempenho. A função deve ser distribuída e a assinatura da função deve incluir a chave de fragmento completa. Todas as colunas na chave de fragmento devem ser transmitidas como parâmetros à função.

Exemplos de função:

```
CREATE OR REPLACE FUNCTION s1.func1(
    param_a bigint,
    param_b bigint,
    param_c char(100)
)
RETURNS int AS $$
DECLARE
    res int;
BEGIN
    SELECT COUNT(*) INTO res
    FROM s1.t1
    WHERE s1.t1.col_a = param_a
      AND s1.t1.col_b = param_b
      AND s1.t1.col_c = param_c;

    RETURN res;
END
$$ LANGUAGE plpgsql;
```

Antes da distribuição, a função não é otimizada para um único fragmento:

```
EXPLAIN ANALYZE
SELECT * FROM s1.func1(776586194, 654849524, '3ac2f2affb02987159ccd6ebd23e1ae5');
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Function Scan on func1  (cost=0.25..0.26 rows=1 width=4)
                         (actual time=37.503..37.503 rows=1 loops=1)
 Planning Time: 0.901 ms
 Execution Time: 51.647 ms
(3 rows)
```

Para distribuir a função:

```
SELECT rds_aurora.limitless_distribute_function(
    's1.func1(bigint,bigint,character)',
    ARRAY['param_a','param_b','param_c'],
    's1.t1'
);
```

Após distribuição, a função é otimizada para um único fragmento:

```
EXPLAIN ANALYZE
SELECT * FROM s1.func1(776586194, 654849524, '3ac2f2affb02987159ccd6ebd23e1ae5');
```

```
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
               (actual time=4.332..4.333 rows=1 loops=1)
 Single Shard Optimized
 Planning Time: 0.857 ms
 Execution Time: 5.116 ms
(4 rows)
```

É possível confirmar a otimização de fragmento único verificando a coluna `sso_calls` em `rds_aurora.limitless_stat_statements`:

```
subcluster_id | subcluster_type | calls | sso_calls |               query
--------------+-----------------+-------+-----------
+-------------------------------------
```

```
 2            | router          |     2 |         1 | SELECT * FROM s1.func1( $1, $2,
 $3 )
 3            | router          |     1 |         1 | SELECT * FROM s1.func1( $1, $2,
 $3 )
(2 rows)
```

Funções e padrões de eficiência

Executar a lógica próximo aos dados é mais eficiente, e as funções desempenham um papel fundamental para conseguir isso. Há dois casos de uso principais para melhorar a eficiência com funções:

1. Extrair a chave de fragmento de dados complexos para invocar uma função separada otimizada para um único fragmento
2. Transformar workloads entre fragmentos em otimizadas para um único fragmento separando a lógica entre fragmentos das instruções otimizadas para um único fragmento

Extrair a chave de fragmento de dados complexos

Considere uma função com assinatura `s3.func3(p_json_doc json)` que executa várias operações de banco de dados. Essas operações serão executadas em todos os fragmentos em uma transação que abrange todos os fragmentos. Se o documento JSON contiver a chave de fragmento, será possível criar uma função otimizada para um único fragmento e realizar as operações do banco de dados.

Padrão original:

```
s3.func3(p_json_doc json)
    database operation 1;
    database operation 2;
    database operation 3;
```

Padrão otimizado:

```
s3.func3(p_json_doc json)
DECLARE
    v_a bigint;
BEGIN
    v_a := (p_json_doc->>'field_a')::bigint;
    SELECT s3.func3_INNER(v_a, p_json_doc);
```

```
END;
```

O que a função interna faz:

```
s3.func3_INNER(p_a, p_json_doc)
    database operation 1 WHERE shard_key = p_a;
    database operation 2 WHERE shard_key = p_a;
    database operation 3 WHERE shard_key = p_a;
```

Nesse padrão, a chave de fragmento é encapsulada em um tipo de dados complexo ou pode ser deduzida de outros parâmetros. A lógica, o acesso aos dados e as funções podem determinar, extrair ou criar a chave de fragmento e, em seguida, invocar uma função otimizada para um único fragmento que execute operações relacionadas apenas a um único fragmento. Como a interface da aplicação não muda, a otimização é comparativamente fácil de testar.

Adiar uma chave de fragmento de outras funções ou dados

Outro padrão de design se aplica quando a lógica ou o acesso aos dados calcula ou determina a chave de fragmento. Isso é útil quando uma função pode ser executada em um único fragmento para a maioria das invocações, mas ocasionalmente requeira a execução entre fragmentos.

Padrão original:

```
NEWORD(INTEGER, …) RETURNS NUMERIC
DECLARE
    all_whid_local := true;
    LOOP through the order lines
        Generate warehouse ID;
        IF generated warehouse ID == input warehouse ID
        THEN
            ol_supply_whid := input warehouse ID;
        ELSE
            all_whid_local := false;
            ol_supply_whid := generated warehouse ID;
        END IF;
        …
    END LOOP;
    …
    RETURN no_s_quantity;
```

Padrão otimizado com funções separadas:

```
CREATE OR REPLACE FUNCTION NEWORD_sso(no_w_id INTEGER, …)
RETURNS NUMERIC
…
    RETURN no_s_quantity;
    …
END;
LANGUAGE 'plpgsql';

SELECT rds_aurora.limitless_distribute_function(
    'NEWORD_sso(int,…)',
    ARRAY['no_w_id'],
    'warehouse'
);

CREATE OR REPLACE FUNCTION NEWORD_crosshard(no_w_id INTEGER, …)
RETURNS NUMERIC
…
    RETURN no_s_quantity;
    …
END;
LANGUAGE 'plpgsql';
```

Em seguida, faça com que a função principal chame a versão otimizada para um único fragmento ou a versão entre fragmentos:

```
IF all_whid_local THEN
    SELECT NEWORD_sso(…) INTO no_s_quantity;
ELSE
    SELECT NEWORD_crosshard(…) INTO no_s_quantity;
END IF;
```

Essa abordagem permite que a maioria das invocações se beneficie da otimização de fragmento único, mantendo o comportamento correto para casos que exigem execução entre fragmentos.

Verificar operações de fragmento único

Use `EXPLAIN` para verificar se uma instrução é otimizada para um único fragmento. A saída relata explicitamente “Single Shard Optimized” para operações otimizadas.

Invocação entre fragmentos antes da distribuição:

```
                        QUERY PLAN
```

```
-------------------------------------------------------------------------
 Function Scan on func1  (cost=0.25..0.26 rows=1 width=4)
                         (actual time=59.622..59.623 rows=1 loops=1)
 Planning Time: 0.925 ms
 Execution Time: 60.211 ms
```

Invocação de fragmento único após a distribuição:

```
                       QUERY PLAN
-------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
               (actual time=4.576..4.577 rows=1 loops=1)
 Single Shard Optimized
 Planning Time: 1.483 ms
 Execution Time: 5.404 ms
```

A diferença nos tempos de execução demonstra o benefício de desempenho da otimização de fragmento único.

Monitoramento de desempenho e recursos específicos da instância

O monitoramento em nível de instância é fundamental para entender as distorções de conexão, de workload e de dados, bem como saber quando adicionar roteadores ou fragmentos divididos para aumentar a escala verticalmente e obter um throughput mais alto com a mesma latência.

Visão geral

Quando a aplicação emite uma consulta, essa solicitação percorre um sofisticado sistema distribuído antes de exibir os resultados. Uma instrução `SELECT` aparentemente simples pode envolver várias instâncias de banco de dados, cada uma desempenhando um papel distinto no processamento da solicitação. Entender esse processo, e as instâncias envolvidas para realizá-lo, transforma a maneira como você projeta aplicações, interpreta dados de monitoramento e diagnostica problemas de desempenho.

Este guia apresenta uma visão técnica detalhada sobre a arquitetura de instâncias:

- Atualização ilimitada sobre a arquitetura, roteadores e fragmentos
- Quando e como escalar cada tipo de instância para atender a requisitos de desempenho e capacidade
- Como monitorar, solucionar problemas e otimizar o desempenho em nível de instância
- Práticas recomendadas de design de aplicações que utilizam a arquitetura distribuída de forma eficaz

Fundamentos da arquitetura de instâncias

alcança escalabilidade horizontal por meio da separação funcional em dois tipos de instância especializados:

- As instâncias de roteador oferecem a camada de orquestração. Elas aceitam conexões de clientes, analisam consultas, coordenam operações distribuídas e agregam resultados. Os roteadores não têm estado, o que significa que não armazenam dados e podem ser adicionados ou removidos sem migração de dados.
- As instâncias de fragmento fornecem os dados e a camada de computação. Elas armazenam dados de tabela, executam consultas em dados locais e lidam com transações. Os fragmentos têm estado e cada um tem um subconjunto específico de dados determinado por um hashing consistente.

Essa separação permite escalar o tratamento de conexões, a coordenação de consultas e o armazenamento de dados de forma independente, com base nas características da workload.

Comparação entre roteadores e fragmentos

Característica	Instâncias de roteador	Instâncias de fragmento
Função principal	Coordenação e distribuição de consultas	Armazenamento de dados e execução de consultas
Estado	Sem estado (não há armazenamento de dados)	Com estado (possui dados)
Escalabilidade	Adicionar/remover instantaneamente	Requer rebalanceamento de dados
Foco do recurso	CPU para coordenação; memória moderada	CPU para consultas; alta memória para cache
Acionadores de ajuste de escala	Alta contagem de conexões, taxa de transação distribuída	CPU, volume de dados e throughput de consultas altos

Monitorar o desempenho da instância

Compreender o desempenho em nível de instância é essencial para operar com eficiência. O monitoramento específico da instância revela os padrões de distribuição que afetam o desempenho: distorções de conexão, de workload e de dados.

Detecção de distorções

Em uma implantação ideal, a workload e os recursos são distribuídos uniformemente entre as instâncias. Na prática, as aplicações experimentam frequentemente uma distribuição distorcida e desigual que concentra a carga em instâncias específicas.

Três tipos de distorção que devem ser monitorados:

- Distorção de conexão: distribuição desigual de conexões de cliente entre os roteadores.
- Distorção de workload: carga de consulta desigual entre fragmentos devido a chaves de fragmentos “quentes”.

- Distorção de dados: volume de dados desigual entre fragmentos devido à frequência da chave de fragmento

Distribuição de carga do Database Insights

A maneira mais rápida de avaliar a integridade em nível de instância é por meio da visualização de distribuição de carga do Database Insights, que oferece visibilidade imediata sobre como as sessões ativas se distribuem entre as instâncias.

Para acessar as distribuição de carga:

1. Acesse o console do RDS → Seu cluster ilimitado.
2. Selecione a guia “Insights de Performance”.
3. Clique na seção “Distribuição de carga”.

Padrão íntegro: a carga é distribuída de maneira relativamente uniforme entre as instâncias.

- Os roteadores podem mostrar uma média de sessões ativas (AAS) um pouco mais alta do que os fragmentos (custos indiretos de coordenação).
- Valores de AAS de 20% entre os fragmentos indicam um bom equilíbrio.

Padrão preocupante: concentração significativa em instâncias específicas.

- Um roteador com >70% de carga de roteador → Distorção de conexão.
- Um fragmento com >50% de carga de fragmento → Distorção de workload ou de dados.
- Grande variação entre fragmentos → Investigar a distribuição de chaves de fragmentos.

Métricas do CloudWatch

Para oferecer uma análise mais detalhada que a do Database Insights, o CloudWatch disponibiliza métricas específicas da instância que revelam padrões de utilização de recursos.

A métrica `ServerlessDatabaseCapacity` com dimensão `DBShardGroupInstance` mostra o consumo de ACU por instância, oferecendo a visualização mais direta da utilização de recursos.

Quando investigar:

- Variação de ACU do roteador >30% → Distorção de conexão ou concentração de workload entre fragmentos.
- Variação da ACU do fragmento > 40% → Distorção de dados ou workload.
- Qualquer instância que use a ACU máxima de forma consistente → Restrição de capacidade.

Monitoramento de roteadores e solução de problemas

Os roteadores podem enfrentar problemas de desempenho por duas causas principais: distribuição desigual de conexões e concentração de workload entre fragmentos.

Sessões distribuídas de forma desigual

Sintoma: um roteador lida com uma parcela desproporcional de conexões.

Causa-raiz: o armazenamento em cache de DNS faz com que várias solicitações de conexão sejam resolvidas no mesmo endpoint do roteador.

Mais comum durante:

- Avaliação comparativa com ferramentas como o pgbench.
- Inicialização do grupo de conexões (várias conexões são estabelecidas rapidamente).
- O servidor de aplicação é reiniciado.

Correções:

- Use o endpoint ilimitado especificado no console.
- Balanceamento manual: extraia endpoints de roteador e conecte diferentes aplicações a roteadores diferentes.
- Para aplicativos libpq, use o recurso `LOADBALANCEHOSTS`.
- Para aplicações JDBC, use o plug-in de conexão ilimitada.
- Use um NLB para gerenciar sessões e distribuições.

Monitoramento de fragmentos e solução de problemas

Os fragmentos enfrentam problemas de desempenho por três causas principais: restrições de recurso, distorção de dados e distorção de workload.

Utilização de recursos de fragmento

Um fragmento com chaves de fragmento conhecidas terá mais dados e maiores workloads. Isso se manifesta como utilização de recursos, ou seja, a instância consumirá mais ACUs.

Estratégias de correção:

1. Reavalie a seleção da chave de fragmento: analise a cardinalidade da chave de fragmento e os padrões de acesso. Considere a possibilidade de usar chaves de fragmento compostas para melhorar a distribuição.
2. Divida o fragmento: distribua a carga entre mais instâncias de fragmento.

Quando dividir os fragmentos:

- Fragmento único que use >80% da ACU máxima de forma consistente.
- Throughput de consulta limitado pela capacidade de um único fragmento.

Volumes de dados de fragmentos

Use funções SQL para consultar volumes de dados:

```
SELECT subcluster_id, subcluster_type, pg_size_pretty(db_size)
FROM rds_aurora.limitless_stat_database_size('postgres_limitless')
ORDER BY 1;
```

Para visualizar dados por tabela e por fragmento:

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public', 'table_name');
```

Resolver a utilização desigual

Quando a distorção de workload ou de dados se concentra em fragmentos específicos, a divisão de fragmentos redistribui a carga entre mais instâncias.

Considerações importantes:

- Não é possível controlar quais chaves de fragmento devem ser movidas.
- Não há como desfazer uma divisão sem recuperar um snapshot manual tirado antes da divisão.
- Todas as instâncias, inclusive um novo fragmento, consomem a ACU mínima quando ociosas.

A divisão de fragmentos permite maior ajuste de escala, e divisões consecutivas de fragmentos são o caminho para obter um throughput mais alto e maior ajuste de escala, mantendo baixa latência.

Limitações

Esteja ciente destas restrições operacionais:

Limitações dos roteadores:

- Não é possível remover roteadores: uma vez adicionados, os roteadores permanecem no cluster.
- Planeje a adição de roteadores com cuidado para evitar custos básicos desnecessários.

Limitações dos fragmentos:

- Não é possível mesclar fragmentos: as divisões de fragmentos são operações unidirecionais.
- Há apenas uma opção de recuperação: restaurar com base em um snapshot tirado antes da divisão.

Estratégias de mitigação:

- Iniciar com uma contagem mínima de instâncias viáveis.
- Adicionar capacidade incrementalmente conforme necessário.
- Tirar snapshots antes de grandes alterações na topologia.
- Monitorar os custos básicos à medida que o cluster cresce

Fazer backup e restaurar o Aurora PostgreSQL Limitless Database

É possível fazer backup e restaurar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database.

Sumário

- Fazer backup de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database
  - Criar um snapshot de cluster de banco de dados
- Restaurar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database
  - Restaurar um cluster de banco de dados a partir de um snapshot de banco de dados
  - Restaurar um cluster de banco de dados usando a recuperação para um ponto no tempo
- Utilitários de backup e restauração do PostgreSQL não são compatíveis

Fazer backup de um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database

O backup de um cluster de banco de dados com o Aurora PostgreSQL Limitless Database tem semelhanças e diferenças de funcionalidade em comparação ao backup de um cluster de banco de dados padrão do Aurora.

- Quando você tira um snapshot manual de cluster de banco de dados de um cluster de banco de dados do Aurora que usa o Limitless Database, o snapshot inclui dados do grupo de fragmentos de banco de dados.
- Os backups contínuos incluem dados do grupo de fragmentos de banco de dados.
- Os snapshots diários automatizados incluem dados do grupo de fragmentos de banco de dados.
- A cópia de snapshots de cluster de banco de dados é compatível. Para ter mais informações, consulte Cópia de snapshot de cluster de banco de dados.
- O compartilhamento de snapshots de cluster de banco de dados é compatível. Para ter mais informações, consulte Compartilhar um snapshot do cluster de banco de dados.
- Não é possível usar o utilitário `pg_dump` ou `pg_dumpall` para fazer backup de bancos de dados no grupo de fragmentos de banco de dados.
- A captura de snapshots finais ao excluir clusters de banco de dados é compatível com o Aurora PostgreSQL Limitless Database.

- A retenção de backups automatizados ao excluir clusters de banco de dados não é compatível com o Aurora PostgreSQL Limitless Database.

Criar um snapshot de cluster de banco de dados

Você cria um snapshot de cluster de banco de dados do Aurora PostgreSQL Limitless Database da mesma forma que um cluster de banco de dados padrão do Aurora, conforme mostrado no seguinte exemplo banco de dados AWS CLI:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier my-db-cluster \
    --db-cluster-snapshot-identifier my-db-cluster-snapshot
```

Para obter mais informações sobre o backup de clusters de banco de dados, consulte Visão geral do backup e da restauração de um cluster de banco de dados do Aurora.

Restaurar um cluster de banco de dados que usa o Aurora PostgreSQL Limitless Database

A restauração de um cluster de banco de dados com o Aurora PostgreSQL Limitless Database tem semelhanças e diferenças de funcionalidade em comparação à restauração de um cluster de banco de dados padrão do Aurora.

- É possível restaurar um cluster de banco de dados do Limitless Database somente a partir de um cluster de banco de dados de origem que usa uma versão de mecanismo de banco de dados compatível com o Limitless Database, como `16.4-limitless`.
- Quando você restaura um cluster de banco de dados de um snapshot manual de um cluster de banco de dados que usa o Limitless Database, todo o armazenamento do cluster de banco de dados é restaurado. Isso inclui o armazenamento do grupo de fragmentos de banco de dados.

  É necessário criar um grupo de fragmentos de banco de dados para acessar o armazenamento do Limitless Database.
- É possível restaurar um cluster de banco de dados usando a recuperação para um ponto no tempo (PITR) para qualquer ponto dentro do período de retenção. O cluster de banco de dados restaurado inclui o armazenamento do grupo de fragmentos de banco de dados.

  É necessário criar um grupo de fragmentos de banco de dados para acessar o armazenamento do Limitless Database.
- A PITR não é compatível com clusters de banco de dados do Aurora PostgreSQL Limitless Database excluídos.
- Quando você restaura um cluster de banco de dados de um snapshot diário automatizado, o armazenamento do grupo de fragmentos de banco de dados também é restaurado.
- Ao restaurar um cluster de banco de dados do Aurora PostgreSQL Limitless Database, você deve habilitar o Monitoramento aprimorado e o Insights de Performance. Inclua o ID da chave do KMS do Insights de Performance.

Depois de restaurar um cluster de banco de dados do Aurora PostgreSQL Limitless Database, verifique sua funcionalidade executando consultas nele.

Restaurar um cluster de banco de dados a partir de um snapshot de banco de dados

Os exemplos de AWS CLI a seguir mostram como restaurar um cluster de banco de dados do Aurora PostgreSQL Limitless Database a partir de um snapshot de cluster de banco de dados.

É necessário usar a versão `16.4-limitless` do mecanismo de banco de dados.

Como restaurar um cluster de banco de dados do Limitless Database a partir de um snapshot de cluster de banco de dados.

1. Restaure o cluster de banco de dados:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-db-cluster \
    --snapshot-identifier my-db-cluster-snapshot \
    --engine aurora-postgresql \
    --engine-version 16.4-limitless \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crie o grupo de fragmentos de banco de dados:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Para ter mais informações, consulte Adição de um grupo de fragmentos de banco de dados a um cluster de banco de dados do Aurora PostgreSQL Limitless Database existente.

Para obter mais informações sobre a restauração de clusters de banco de dados do Aurora a partir de snapshots de cluster de banco de dados, consulte Restauração de um snapshot de um cluster de banco de dados.

Restaurar um cluster de banco de dados usando a recuperação para um ponto no tempo

Os exemplos de AWS CLI a seguir mostram como restaurar um cluster de banco de dados do Aurora PostgreSQL Limitless Database usando a recuperação para um ponto no tempo (PITR).

Como restaurar um cluster de banco de dados do Limitless Database usando PITR

1. Restaure o cluster de banco de dados:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-db-cluster \
    --db-cluster-identifier my-new-db-cluster \
    --use-latest-restorable-time \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crie o grupo de fragmentos de banco de dados:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Para ter mais informações, consulte Adição de um grupo de fragmentos de banco de dados a um cluster de banco de dados do Aurora PostgreSQL Limitless Database existente.

Para ter mais informações sobre o PITR, consulte Restaurar um cluster de banco de dados para um horário especificado.

Utilitários de backup e restauração do PostgreSQL não são compatíveis

Os seguintes utilitários do PostgreSQL não são compatíveis nem para o cluster de banco de dados primário nem para o grupo de fragmentos de banco de dados:

- `pg_dump`
- `pg_dumpall`
- `pg_restore`

Embora você possa usá-los por meio de binários de código aberto ou métodos alternativos, isso pode gerar resultados inconsistentes.

Atualização do Amazon Aurora PostgreSQL Limitless Database

No momento, as seguintes considerações se aplicam à atualização do Aurora PostgreSQL Limitless Database:

- Atualizações de versão secundária são compatíveis.
- Há suporte para a aplicação de patches no Aurora PostgreSQL Limitless Database. Os patches aparecem como manutenção pendente para serem aplicados durante sua janela de manutenção.

Atualização de clusters de banco de dados que usam o Amazon Aurora PostgreSQL Limitless Database

Você atualiza um cluster de banco de dados modificando-o e escolhendo uma nova versão do mecanismo de banco de dados. É possível usar o Console de gerenciamento da AWS ou a AWS CLI.

Console

Como atualizar seu cluster de banco de dados do Limitless Database

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Navegue até a página Databases (Bancos de dados).
3. Selecione o cluster de banco de dados do Limitless Database que você deseja atualizar.
4. Escolha Modificar.

   A página Modificar cluster de banco de dados é exibida.
5. Em Versão do mecanismo de banco de dados, escolha a nova versão do mecanismo de banco de dados, por exemplo, Aurora PostgreSQL com Limitless Database (compatível com PostgreSQL 16.6).
6. Escolha Continuar.
7. Na página de resumo, selecione se deseja aplicar as alterações imediatamente ou durante a próxima janela de manutenção programada e, em seguida, escolha Modificar cluster.

CLI

Use o comando modify-db-cluster da AWS CLI, como é mostrado no exemplo a seguir.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-sv2-cluster \
    --engine-version 16.6-limitless \
    --apply-immediately
```

Referência do Aurora PostgreSQL Limitless Database

Fornecemos os seguintes tópicos de referência para o Aurora PostgreSQL Limitless Database.

Tópicos

- Comandos SQL de Linguagem de Definição de Dados (DDL) com e sem suporte
- Limitações de DDL e outras informações do Aurora PostgreSQL Limitless Database
- Comandos SQL de Linguagem de Manipulação de Dados (DML) e processamento de consultas com e sem suporte
- Limitações e outras informações de DML no Aurora PostgreSQL Limitless Database
- Variáveis no Aurora PostgreSQL Limitless Database
- Parâmetros do cluster de banco de dados no Aurora PostgreSQL Limitless Database

Comandos SQL de Linguagem de Definição de Dados (DDL) com e sem suporte

A tabela a seguir lista os comandos de DDL com e sem suporte do Aurora PostgreSQL Limitless Database, com referências a limitações ou mais informações.

Command	Compatível?	Limitações ou mais informaçõ es
ALTER AGGREGATE	Não	Não aplicável
ALTER COLLATION	Sim	Nenhum
ALTER CONVERSION	Sim	Nenhum
ALTER DATABASE	Não	Não aplicável
ALTER DEFAULT PRIVILEGE S	Não	Não aplicável
ALTER DOMAIN	Não	Não aplicável
ALTER EVENT TRIGGER	Não	Não aplicável

Command	Compatível?	Limitações ou mais informaçõ es
ALTERAR EXTENSÃO	Sim	Extensões
ALTER FOREIGN DATA WRAPPER	Não	Não aplicável
ALTER FOREIGN TABLE	Não	Não aplicável
ALTER FUNCTION	Sim	Funções
ALTER GROUP	Sim	Nenhum
ALTER INDEX	Sim	Nenhum
ALTER LANGUAGE	Não	Não aplicável
ALTER LARGE OBJECT	Não	Não aplicável
ALTER MATERIALIZED VIEW	Não	Não aplicável
ALTER OPERATOR	Sim	Nenhum
ALTER OPERATOR CLASS	Sim	Nenhum
ALTER OPERATOR FAMILY	Sim	Nenhum
ALTER POLICY	Não	Não aplicável
ALTER PROCEDURE	Sim	Nenhum
ALTER PUBLICATION	Não	Não aplicável
ALTER ROLE	Sim	Nenhum
ALTER ROUTINE	Não	Não aplicável
ALTER RULE	Não	Não aplicável
ALTER SCHEMA	Sim	Nenhum

Command	Compatível?	Limitações ou mais informaçõ es
ALTER SEQUENCE	Sim	Sequências
ALTER SERVER	Não	Não aplicável
ALTER STATISTICS	Não	Não aplicável
ALTER SUBSCRIPTION	Não	Não aplicável
ALTER SYSTEM	Não	Não aplicável
ALTER TABLE	Sim	ALTER TABLE
ALTER TABLESPACE	Não	Não aplicável
ALTER TEXT SEARCH CONFIGURATION	Não	Não aplicável
ALTER TEXT SEARCH DICTIONARY	Não	Não aplicável
ALTER TEXT SEARCH PARSER	Não	Não aplicável
ALTER TEXT SEARCH TEMPLATE	Não	Não aplicável
ALTER TRIGGER	Não	Não aplicável
ALTER TYPE	Sim	Nenhum
ALTER USER	Sim	Nenhum
ALTER USER MAPPING	Não	Não aplicável
ALTER VIEW	Sim	Nenhum
COMMENT	Não	Não aplicável

Command	Compatível?	Limitações ou mais informaçõ es
CREATE ACCESS METHOD	Não	Não aplicável
CREATE AGGREGATE	Não	Não aplicável
CREATE CAST	Sim	Nenhum
CREATE COLLATION	Sim	Nenhum
CREATE CONVERSION	Sim	Nenhum
CREATE DATABASE	Sim	CREATE DATABASE
CREATE DOMAIN	Não	Não aplicável
CREATE EVENT TRIGGER	Não	Não aplicável
CRIAR EXTENSÃO	Sim	Extensões
CREATE FOREIGN DATA WRAPPER	Não	Não aplicável
CREATE FOREIGN TABLE	Não	Não aplicável
CREATE FUNCTION	Sim	Funções
CREATE GROUP	Sim	Nenhum
CREATE INDEX	Sim	CREATE INDEX
CREATE LANGUAGE	Não	Não aplicável
CREATE MATERIALIZED VIEW	Não	Não aplicável
CREATE OPERATOR	Sim	Nenhum
CREATE OPERATOR CLASS	Sim	Nenhum

Command	Compatível?	Limitações ou mais informaçõ es
CREATE OPERATOR FAMILY	Sim	Nenhum
CREATE POLICY	Sim	Nenhum
CREATE PROCEDURE	Sim	Nenhum
CREATE PUBLICATION	Não	Não aplicável
CRIAR PERFIL	Sim	Nenhum
CREATE RULE	Não	Não aplicável
CREATE SCHEMA	Sim	CREATE SCHEMA
CREATE SEQUENCE	Sim	Sequências
CREATE SERVER	Não	Não aplicável
CREATE STATISTICS	Não	Não aplicável
CREATE SUBSCRIPTION	Não	Não aplicável
CRIAR TABELA	Sim	CRIAR TABELA
CREATE TABLE AS	Sim	CREATE TABLE AS
CREATE TABLESPACE	Não	Não aplicável
CREATE TEMPORARY TABLE	Não	Não aplicável
CREATE TEMPORARY TABLE AS	Não	Não aplicável
CREATE TEXT SEARCH CONFIGURATION	Não	Não aplicável

Command	Compatível?	Limitações ou mais informaçõ es
CREATE TEXT SEARCH DICTIONARY	Não	Não aplicável
CREATE TEXT SEARCH PARSER	Não	Não aplicável
CREATE TEXT SEARCH TEMPLATE	Não	Não aplicável
CREATE TRANSFORM	Não	Não aplicável
CREATE TRIGGER	Não	Não aplicável
CREATE TYPE	Sim	Nenhum
CRIAR USUÁRIO	Sim	Nenhum
CREATE USER MAPPING	Não	Não aplicável
CREATE VIEW	Sim	Nenhum
DROP ACCESS METHOD	Não	Não aplicável
DROP AGGREGATE	Sim	Nenhum
DROP CAST	Sim	Nenhum
DROP COLLATION	Sim	Nenhum
DROP CONVERSION	Sim	Nenhum
DROP DATABASE	Sim	DROP DATABASE
DROP DOMAIN	Não	Não aplicável
DROP EVENT TRIGGER	Não	Não aplicável
EXTENSÃO DROP	Sim	Extensões

Command	Compatível?	Limitações ou mais informaçõ es
DROP FOREIGN DATA WRAPPER	Não	Não aplicável
DROP FOREIGN TABLE	Não	Não aplicável
DROP FUNCTION	Sim	Funções
DROP GROUP	Sim	Nenhum
DROP INDEX	Sim	Nenhum
DROP LANGUAGE	Não	Não aplicável
DROP MATERIALIZED VIEW	Não	Não aplicável
DROP OPERATOR	Sim	Nenhum
DROP OPERATOR CLASS	Sim	Nenhum
DROP OPERATOR FAMILY	Sim	Nenhum
DROP OWNED	Não	Não aplicável
DROP POLICY	Não	Não aplicável
DROP PROCEDURE	Sim	Nenhum
DROP PUBLICATION	Não	Não aplicável
DROP ROLE	Sim	Nenhum
DROP ROUTINE	Não	Não aplicável
DROP RULE	Não	Não aplicável
DROP SCHEMA	Sim	Nenhum
DROP SEQUENCE	Sim	Nenhum

Command	Compatível?	Limitações ou mais informaçõ es
DROP SERVER	Não	Não aplicável
DROP STATISTICS	Não	Não aplicável
DROP SUBSCRIPTION	Não	Nenhum
DESCARTAR TABELA	Sim	Nenhum
DROP TABLESPACE	Não	Não aplicável
DROP TEXT SEARCH CONFIGURATION	Não	Não aplicável
DROP TEXT SEARCH DICTIONARY	Não	Não aplicável
DROP TEXT SEARCH PARSER	Não	Não aplicável
DROP TEXT SEARCH TEMPLATE	Não	Não aplicável
DROP TRANSFORM	Não	Não aplicável
DROP TRIGGER	Não	Não aplicável
DROP TYPE	Sim	Nenhum
DROP USER	Sim	Nenhum
DROP USER MAPPING	Não	Não aplicável
DROP VIEW	Sim	Nenhum
GRANT	Sim	Nenhum
REASSIGN OWNED	Não	Não aplicável

Command	Compatível?	Limitações ou mais informaçõ es
REVOKE	Sim	Nenhum
SECURITY LABEL	Não	Não aplicável
SELECT INTO	Sim	SELECT INTO
SET	Sim	Nenhum
SET CONSTRAINTS	Não	Não aplicável
SET ROLE	Sim	Nenhum
SET SESSION AUTHORIZA TION	Sim	Nenhum
SET TRANSACTION	Sim	Nenhum
TRUNCATE	Sim	Nenhum

Limitações de DDL e outras informações do Aurora PostgreSQL Limitless Database

Os tópicos a seguir descrevem as limitações ou fornecem mais informações sobre os comandos SQL para DDL no Aurora PostgreSQL Limitless Database.

Tópicos

- ALTER TABLE
- CREATE DATABASE
- CREATE INDEX
- CREATE SCHEMA
- CRIAR TABELA
- CREATE TABLE AS
- DROP DATABASE
- SELECT INTO
- Restrições
- Valores padrão
- Extensões
- Chaves externas
- Funções
- Sequências

ALTER TABLE

O comando `ALTER TABLE` geralmente é compatível com o Aurora PostgreSQL Limitless Database. Consulte ALTER TABLE na documentação do PostgreSQL para obter mais informações.

Limitações

`ALTER TABLE` tem as limitações a seguir para as opções compatíveis.

Remover uma coluna

- Em tabelas fragmentadas, não é possível remover colunas que fazem parte da chave de fragmento.
- Em tabelas de referência, não é possível remover colunas de chave primária.

Alterar um tipo de dado da coluna

- A expressão `USING` não é compatível.
- Em tabelas fragmentadas, não é possível alterar o tipo das colunas que fazem parte da chave de fragmento.

Adicionar ou remover uma restrição

Consulte Restrições para obter detalhes sobre o que não é compatível.

Alterar o valor padrão de uma coluna

Há suporte para valores padrão. Para obter mais informações, consulte Valores padrão.

Opções não compatíveis

Algumas opções não são compatíveis porque elas dependem de recursos incompatíveis, como gatilhos.

As seguintes opções no nível de tabela para `ALTER TABLE` não são compatíveis:

- `ALL IN TABLESPACE`
- `ATTACH PARTITION`
- `DETACH PARTITION`
- sinalizador `ONLY`
- `RENAME CONSTRAINT`

As seguintes opções no nível de coluna para `ALTER TABLE` não são compatíveis:

- ADD GENERATED
- DROP EXPRESSION [IF EXISTS]
- DROP IDENTITY [IF EXISTS]
- RESET
- RESTART
- SET
- SET COMPRESSION
- SET STATISTICS

CREATE DATABASE

No Aurora PostgreSQL Limitless Database, somente bancos de dados ilimitados são compatíveis.

Enquanto `CREATE DATABASE` estiver em execução, os bancos de dados que foram criados com sucesso em um ou mais nós podem falhar em outros nós, visto que a criação do banco de dados é uma operação não transacional. Nesse caso, os objetos do banco de dados que foram criados com sucesso são automaticamente removidos de todos os nós em um período predeterminado para manter a consistência no grupo de fragmentos do banco de dados. Durante esse período, a recriação de um banco de dados com o mesmo nome pode resultar em um erro indicando que o banco de dados já existe.

Há compatibilidade com as seguintes opções:

- Agrupamento:

```
CREATE DATABASE name WITH
    [LOCALE = locale]
    [LC_COLLATE = lc_collate]
    [LC_CTYPE = lc_ctype]
    [ICU_LOCALE = icu_locale]
    [ICU_RULES = icu_rules]
    [LOCALE_PROVIDER = locale_provider]
    [COLLATION_VERSION = collation_version];
```

- `CREATE DATABASE WITH OWNER`:

```
CREATE DATABASE name WITH OWNER = user_name;
```

As seguintes opções não são compatíveis:

- `CREATE DATABASE WITH TABLESPACE`:

```
CREATE DATABASE name WITH TABLESPACE = tablespace_name;
```

- `CREATE DATABASE WITH TEMPLATE`:

```
CREATE DATABASE name WITH TEMPLATE = template;
```

CREATE INDEX

`CREATE INDEX CONCURRENTLY` é compatível com tabelas fragmentadas:

```
CREATE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX` é compatível com todos os tipos de tabela:

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX CONCURRENTLY` não é compatível:

```
CREATE UNIQUE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Para obter mais informações, consulte UNIQUE. Para obter informações gerais sobre a criação de índices, consulte CREATE INDEX na documentação do PostgreSQL.

Exibir índices

> Nem todos os índices são visíveis nos roteadores ao usar `\d table_name` ou comandos semelhantes. Em vez disso, use a exibição `pg_catalog.pg_indexes` para obter índices, conforme mostrado no exemplo a seguir.

```
SET rds_aurora.limitless_create_table_mode='sharded';
```

```
SET rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items (id int PRIMARY KEY, val int);
CREATE INDEX items_my_index on items (id, val);

postgres_limitless=> SELECT * FROM pg_catalog.pg_indexes WHERE tablename='items';

 schemaname | tablename |   indexname    | tablespace |
   indexdef
------------+-----------+----------------+------------
+-------------------------------------------------------------------------
 public     | items     | items_my_index |            | CREATE INDEX items_my_index
 ON ONLY public.items USING btree (id, val)
 public     | items     | items_pkey     |            | CREATE UNIQUE INDEX
 items_pkey ON ONLY public.items USING btree (id)
(2 rows)
```

CREATE SCHEMA

Não há suporte para `CREATE SCHEMA` com um elemento de esquema:

```
CREATE SCHEMA my_schema CREATE TABLE (column_name INT);
```

Isso gera um erro semelhante ao seguinte:

```
ERROR: CREATE SCHEMA with schema elements is not supported
```

CRIAR TABELA

Não há suporte para relações em declarações `CREATE TABLE`, por exemplo:

```
CREATE TABLE orders (orderid int, customerId int, orderDate date) WITH
 (autovacuum_enabled = false);
```

Não há suporte para colunas `IDENTITY`, por exemplo:

```
CREATE TABLE orders (orderid INT GENERATED ALWAYS AS IDENTITY);
```

O Aurora PostgreSQL Limitless Database aceita nomes de tabela fragmentada com até 54 caracteres.

CREATE TABLE AS

Para criar uma tabela usando `CREATE TABLE AS`, use a variável `rds_aurora.limitless_create_table_mode`. Para tabelas fragmentadas, você também deve usar a variável `rds_aurora.limitless_create_table_shard_key`. Para obter mais informações, consulte Criação de tabelas ilimitadas usando variáveis.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

CREATE TABLE ctas_table AS SELECT 1 a;

-- "source" is the source table whose columns and data types are used to create the new
 "ctas_table2" table.
CREATE TABLE ctas_table2 AS SELECT a,b FROM source;
```

Não é possível usar `CREATE TABLE AS` para criar tabelas de referência porque elas exigem restrições de chave primária. `CREATE TABLE AS` não propaga chaves primárias para novas tabelas.

Para obter informações gerais, consulte CREATE TABLE AS na documentação do PostgreSQL.

DROP DATABASE

É possível descartar bancos de dados que você criou.

O comando `DROP DATABASE` é executado de forma assíncrona em segundo plano. Enquanto estiver em execução, um erro será retornado se você tentar criar um banco de dados com o mesmo nome.

SELECT INTO

`SELECT INTO` é funcionalmente semelhante a CREATE TABLE AS. Você deve usar a variável `rds_aurora.limitless_create_table_mode`. Para tabelas fragmentadas, você também deve usar a variável `rds_aurora.limitless_create_table_shard_key`. Para obter mais informações, consulte Criação de tabelas ilimitadas usando variáveis.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';
```

```
-- "source" is the source table whose columns and data types are used to create the new
 "destination" table.
SELECT * INTO destination FROM source;
```

Atualmente, a operação `SELECT INTO` é realizada por meio do roteador, não diretamente pelos fragmentos. Portanto, o desempenho pode ser lento.

Para obter informações gerais, consulte SELECT INTO na documentação do PostgreSQL.

Restrições

As limitações a seguir se aplicam às restrições no Aurora PostgreSQL Limitless Database.

CHECK

Há suporte para restrições simples que envolvem operadores de comparação com literais. Não há suporte para expressões e restrições mais complexas que exigem validações de funções, conforme mostrado nos exemplos a seguir.

```
CREATE TABLE my_table (
    id  INT CHECK (id > 0)                                        -- supported
  , val INT CHECK (val > 0 AND val < 1000)                        -- supported
  , tag TEXT CHECK (length(tag) > 0)                              -- not supported:
 throws "Expression inside CHECK constraint is not supported"
  , op_date TIMESTAMP WITH TIME ZONE CHECK (op_date <= now())  -- not supported:
 throws "Expression inside CHECK constraint is not supported"
);
```

Você pode dar nomes explícitos às restrições, conforme mostrado no exemplo a seguir.

```
CREATE TABLE my_table (
    id  INT CONSTRAINT positive_id  CHECK (id > 0)
  , val INT CONSTRAINT val_in_range CHECK (val > 0 AND val < 1000)
);
```

É possível utilizar a sintaxe de restrição em nível de tabela com a restrição `CHECK`, conforme mostrado no exemplo a seguir.

```
CREATE TABLE my_table (
    id INT CONSTRAINT positive_id  CHECK (id > 0)
  , min_val INT CONSTRAINT min_val_in_range CHECK (min_val > 0 AND min_val < 1000)
  , max_val INT
  , CONSTRAINT max_val_in_range CHECK (max_val > 0 AND max_val < 1000 AND max_val >
 min_val)
);
```

EXCLUDE

Não há suporte para restrições de exclusão no Aurora PostgreSQL Limitless Database.

FOREIGN KEY

Para obter mais informações, consulte Chaves externas.

NOT NULL

Há suporte para restrições `NOT NULL` sem limitações.

PRIMARY KEY

A chave primária implica restrições únicas e, portanto, as mesmas restrições aplicadas às restrições únicas se aplicam à chave primária. Isso significa que:

- Se uma tabela for convertida em uma tabela fragmentada, a chave de fragmento deve ser um subconjunto da chave primária. Ou seja, a chave primária contém todas as colunas da chave de fragmento.
- Se uma tabela for convertida em uma tabela de referência, ela deverá ter uma chave primária.

Os exemplos a seguir ilustram o uso de chaves primárias.

```
-- Create a standard table.
CREATE TABLE public.my_table (
    item_id INT
  , location_code INT
  , val INT
  , comment text
);

-- Change the table to a sharded table using the 'item_id' and 'location_code'
 columns as shard keys.
CALL rds_aurora.limitless_alter_table_type_sharded('public.my_table',
 ARRAY['item_id', 'location_code']);
```

Tentando adicionar uma chave primária que não contém uma chave de fragmento:

```
-- Add column 'item_id' as the primary key.
-- Invalid because the primary key doesnt include all columns from the shard key:
-- 'location_code' is part of the shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR

-- add column "val" as primary key
-- Invalid because primary key does not include all columns from shard key:
--  item_id and location_code iare part of shard key but not part of the primary key
```

```
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR
```

Tentando adicionar uma chave primária que contém uma chave de fragmento:

```
-- Add the 'item_id' and 'location_code' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code); -- OK

-- Add the 'item_id', 'location_code', and 'val' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code, val); -- OK
```

Alterar uma tabela padrão para uma tabela de referência.

```
-- Create a standard table.
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);

-- Convert the table to a reference table.
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcode');
```

Consulte Criação de tabelas do Aurora PostgreSQL Limitless Database para obter mais informações sobre a criação de tabelas fragmentadas e de referência.

UNIQUE

Em tabelas fragmentadas, a chave única deve conter a chave de fragmento. Ou seja, a chave de fragmento deve ser um subconjunto da chave única. Isso é verificado ao alterar o tipo de tabela para fragmentada. Não há restrição nas tabelas de referência.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT UNIQUE
);
```

Há suporte para restrições em nível de tabela UNIQUE, conforme mostrado no exemplo a seguir.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT
  , CONSTRAINT zipcode_and_email UNIQUE (zipcode, email)
```

```
);
```

O exemplo a seguir mostra o uso de uma chave primária e uma chave única juntas. Ambas as chaves devem incluir a chave de fragmento.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"p_id"}';
CREATE TABLE t1 (
p_id BIGINT NOT NULL,
c_id BIGINT NOT NULL,
PRIMARY KEY (p_id),
UNIQUE (p_id, c_id)
);
```

Para obter mais informações, consulte Constraints na documentação do PostgreSQL.

Valores padrão

O Aurora PostgreSQL Limitless Database oferece suporte a expressões em valores padrão.

O exemplo a seguir mostra o uso de valores padrão.

```
CREATE TABLE t (
    a INT DEFAULT 5,
    b TEXT DEFAULT 'NAN',
    c NUMERIC
);

CALL rds_aurora.limitless_alter_table_type_sharded('t', ARRAY['a']);
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c
---+-----+---
 5 | NAN |
(1 row)
```

Há suporte para expressões, conforme mostrado no exemplo a seguir.

```
CREATE TABLE t1 (a NUMERIC DEFAULT random());
```

O exemplo a seguir adiciona uma nova coluna que é NOT NULL e tem um valor padrão.

```
ALTER TABLE t ADD COLUMN d BOOLEAN NOT NULL DEFAULT FALSE;
SELECT * FROM t;

 a |  b  | c | d
---+-----+---+---
 5 | NAN |   | f
(1 row)
```

O exemplo a seguir altera uma coluna existente com um valor padrão.

```
ALTER TABLE t ALTER COLUMN c SET DEFAULT 0.0;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
(2 rows)
```

O exemplo a seguir descarta um valor padrão.

```
ALTER TABLE t ALTER COLUMN a DROP DEFAULT;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
   | NAN | 0.0 | f
(3 rows)
```

Para obter mais informações, consulte Default values na documentação do PostgreSQL.

Extensões

Há suporte para as seguintes extensões do PostgreSQL no Aurora PostgreSQL Limitless Database:

- `aurora_limitless_fdw`: essa extensão é pré-instalada. Não é possível descartá-la.
- `aws_s3`: essa extensão funciona no Aurora PostgreSQL Limitless Database de maneira semelhante ao Aurora PostgreSQL.

  É possível importar dados de um bucket do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL Limitless Database ou vice-versa. Para obter mais informações, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL e Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3.
- `btree_gin`
- `citext`
- `ip4r`
- `pg_buffercache`: essa extensão se comporta de forma diferente no Aurora PostgreSQL Limitless Database em comparação com o PostgreSQL da comunidade. Para obter mais informações, consulte Diferenças do pg_buffercache no Aurora PostgreSQL Limitless Database.
- `pg_stat_statements`
- `pg_trgm`
- `pgcrypto`
- `pgstattuple`: essa extensão se comporta de forma diferente no Aurora PostgreSQL Limitless Database em comparação com o PostgreSQL da comunidade. Para obter mais informações, consulte Diferenças do pgstattuple no Aurora PostgreSQL Limitless Database.
- `pgvector`
- `plpgsql`: essa extensão é pré-instalada, mas é possível descartá-la.
- `PostGIS`: não há suporte para transações longas e funções de gerenciamento de tabelas. Não há suporte para a modificação da tabela de referência espacial.
- `unaccent`
- `uuid`

Atualmente, a maioria das extensões do PostgreSQL não é compatível com o Aurora PostgreSQL Limitless Database. No entanto, ainda é possível usar a configuração shared_preload_librarie (SPL)

para carregar extensões no cluster de banco de dados primário do Aurora PostgreSQL. Elas também são carregadas no Aurora PostgreSQL Limitless Database, mas podem não funcionar corretamente.

Por exemplo, você pode carregar a extensão `pg_hint_plan`, mas carregá-la não garante que as dicas passadas nos comentários da consulta sejam usadas.

> **Note**
>
> Não é possível modificar objetos associados à extensão pg_stat_statements.
> Para obter informações sobre a instalação de `pg_stat_statements`, consulte limitless_stat_statements.

Você pode usar as funções `pg_available_extensions` e `pg_available_extension_versions` para encontrar extensões compatíveis com o Aurora PostgreSQL Limitless Database.

As seguintes DDLs são compatíveis com extensões:

CRIAR EXTENSÃO

É possível criar extensões, como no PostgreSQL.

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

Para obter mais informações, consulte CREATE EXTENSION na documentação do PostgreSQL.

ALTERAR EXTENSÃO

As seguintes DDLs são compatíveis:

```
ALTER EXTENSION name UPDATE [ TO new_version ]

ALTER EXTENSION name SET SCHEMA new_schema
```

Para obter mais informações, consulte ALTER EXTENSION na documentação do PostgreSQL.

EXTENSÃO DROP

É possível descartar extensões, como no PostgreSQL.

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Para obter mais informações, consulte DROP EXTENSION na documentação do PostgreSQL.

As seguintes DDLs não são compatíveis com extensões:

ALTERAR EXTENSÃO

Não é possível adicionar ou descartar objetos membros das extensões.

```
ALTER EXTENSION name ADD member_object

ALTER EXTENSION name DROP member_object
```

Diferenças do pg_buffercache no Aurora PostgreSQL Limitless Database

No Aurora PostgreSQL Limitless Database, ao instalar a extensão pg_buffercache e usar a visualização `pg_buffercache`, você recebe informações relacionadas ao buffer somente do nó ao qual está conectado atualmente: o roteador. Da mesma forma, o uso da função `pg_buffercache_summary` ou `pg_buffercache_usage_counts` fornece informações somente do nó conectado.

É possível ter vários nós. Talvez seja preciso acessar as informações do buffer de qualquer nó para diagnosticar problemas de forma eficaz. Portanto, o Limitless Database fornece as seguintes funções:

- `rds_aurora.limitless_pg_buffercache(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_summary(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_usage_counts(subcluster_id)`

Ao inserir o ID do subcluster de qualquer nó, seja um roteador ou um fragmento, é possível acessar facilmente as informações do buffer específicas desse nó. Essas funções estão diretamente disponíveis ao instalar a extensão `pg_buffercache` no Limitless Database.

> **Note**
>
> O Aurora PostgreSQL Limitless Database oferece suporte a essas funções na versão 1.4 e superiores da extensão `pg_buffercache`.

As colunas mostradas na exibição `limitless_pg_buffercache` diferem um pouco das da exibição `pg_buffercache`:

- `bufferid`: permanece inalterado em relação a `pg_buffercache`.
- `relname`: em vez de exibir o número do nó do arquivo como em `pg_buffercache`, o `limitless_pg_buffercache` apresenta o `relname` associado, se disponível no banco de dados atual ou nos catálogos do sistema compartilhado. Caso contrário, exibe `NULL`.
- `parent_relname`: essa nova coluna, não presente em `pg_buffercache`, exibe o `relname` pai se o valor na coluna `relname` representar uma tabela particionada (no caso de tabelas fragmentadas). Caso contrário, exibe `NULL`.
- `spcname`: em vez de exibir o identificador de objeto (OID) do espaço de tabela como em `pg_buffercache`, `limitless_pg_buffercache` exibe o nome do espaço de tabela.
- `datname`: em vez de exibir o OID do banco de dados como em `pg_buffercache`, `limitless_pg_buffercache` exibe o nome do banco de dados.
- `relforknumber`: permanece inalterado em relação a `pg_buffercache`.
- `relblocknumber`: permanece inalterado em relação a `pg_buffercache`.
- `isdirty`: permanece inalterado em relação a `pg_buffercache`.
- `usagecount`: permanece inalterado em relação a `pg_buffercache`.
- `pinning_backends`: permanece inalterado em relação a `pg_buffercache`.

As colunas nas exibições `limitless_pg_buffercache_summary` e `limitless_pg_buffercache_usage_counts` são as mesmas que nas exibições regulares `pg_buffercache_summary` e `pg_buffercache_usage_counts`, respectivamente.

Ao usar essas funções, é possível acessar informações detalhadas do cache de buffer em todos os nós no ambiente do Limitless Database, o que facilita o diagnóstico e o gerenciamento mais eficazes dos sistemas de banco de dados.

Diferenças do pgstattuple no Aurora PostgreSQL Limitless Database

No Aurora PostgreSQL, a extensão pgstattuple atualmente não é compatível com tabelas externas, tabelas particionadas ou índices particionados. No entanto, no Aurora PostgreSQL Limitless Database, objetos criados pelo usuário geralmente estão entre esses tipos que não têm suporte. Embora existam tabelas e índices regulares (por exemplo, tabelas de catálogo e seus índices), a maioria dos objetos reside em nós externos, o que o roteador considera como objetos externos.

Reconhecemos a importância dessa extensão para obter estatísticas no nível da tupla, o que é crucial para tarefas como remoção de sobrecarga e coleta de informações de diagnóstico. Portanto, o Aurora PostgreSQL Limitless Database fornece suporte para a extensão `pgstattuple` em bancos de dados ilimitados.

O Aurora PostgreSQL Limitless Database inclui as seguintes funções no esquema `rds_aurora`:

Funções estatísticas no nível da tupla

`rds_aurora.limitless_pgstattuple(relation_name)`

- Objetivo: extrair estatísticas no nível da tupla para tabelas padrão e seus índices
- Entrada: `relation_name` (texto) — o nome da relação
- Saída: colunas consistentes com as retornadas pela função `pgstattuple` no Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple(relation_name, subcluster_id)`

- Objetivo: extrair estatísticas no nível da tupla para tabelas de referência, tabelas fragmentadas, tabelas de catálogo e seus índices
- Entrada:
  - `relation_name` (texto) — o nome da relação
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde as estatísticas devem ser extraídas
- Saída:
  - Em tabelas de referência e de catálogo (incluindo seus índices), as colunas são consistentes com as do Aurora PostgreSQL.
  - Em tabelas fragmentadas, as estatísticas representam somente a partição da tabela fragmentada que reside no subcluster especificado.

Funções das estatísticas de índice

`rds_aurora.limitless_pgstatindex(`*`relation_name`*`)`

- Objetivo: extrair estatísticas para índices de árvore B em tabelas padrão
- Entrada: `relation_name` (texto) — o nome do índice de árvore B
- Saída: todas as colunas, exceto `root_block_no`, são retornadas. As colunas retornadas são consistentes com a função `pgstatindex` no Aurora PostgreSQL.

`rds_aurora.limitless_pgstatindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extrair estatísticas para índices de árvore B em tabelas de referência, tabelas fragmentadas e tabelas de catálogo.
- Entrada:
  - `relation_name` (texto) — o nome do índice de árvore B
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde as estatísticas devem ser extraídas
- Saída:
  - Em índices de tabelas de referência e catálogo, todas as colunas (exceto `root_block_no`) são retornadas. As colunas retornadas são consistentes com o Aurora PostgreSQL.
  - Em tabelas fragmentadas, as estatísticas representam somente a partição do índice da tabela fragmentada que reside no subcluster especificado. A coluna `tree_level` mostra a média de todas as partes da tabela no subcluster solicitado.

`rds_aurora.limitless_pgstatginindex(`*`relation_name`*`)`

- Objetivo: extrair estatísticas para Índices invertidos generalizados (GINs) em tabelas padrão
- Entrada: `relation_name` (texto) — o nome do GIN
- Saída: colunas consistentes com as retornadas pela função `pgstatginindex` no Aurora PostgreSQL

`rds_aurora.limitless_pgstatginindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extrair estatísticas para GINs em tabelas de referência, tabelas fragmentadas e tabelas de catálogo.
- Entrada:
  - `relation_name` (texto) — o nome do índice
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde as estatísticas devem ser extraídas

- Saída:
  - Nos GINs das tabelas de referência e de catálogo, as colunas são consistentes com as do Aurora PostgreSQL.
  - Em tabelas fragmentadas, as estatísticas representam somente a partição do índice da tabela fragmentada que reside no subcluster especificado.

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`)`

- Objetivo: extrair estatísticas para índices de hash em tabelas padrão
- Entrada: `relation_name` (texto) — o nome do índice de hash
- Saída: colunas consistentes com as retornadas pela função `pgstathashindex` no Aurora PostgreSQL

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extrair estatísticas para índices de hash em tabelas de referência, tabelas fragmentadas e tabelas de catálogo.
- Entrada:
  - `relation_name` (texto) — o nome do índice
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde as estatísticas devem ser extraídas
- Saída:
  - Para índices de hash em tabelas de referência e catálogo, as colunas são consistentes com as do Aurora PostgreSQL.
  - Em tabelas fragmentadas, as estatísticas representam somente a partição do índice da tabela fragmentada que reside no subcluster especificado.

Funções de contagem de páginas

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`)`

- Objetivo: extrair a contagem de páginas em tabelas padrão e seus índices
- Entrada: `relation_name` (texto) — o nome da relação
- Saída: contagem de páginas da relação especificada

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extrair a contagem de páginas das tabelas de referência, tabelas fragmentadas e tabelas de catálogo (incluindo seus índices)

- Entrada:
  - `relation_name` (texto) — o nome da relação
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde a contagem de páginas deve ser extraída
- Saída: em tabelas fragmentadas, a contagem de páginas é a soma das páginas em todas as partes da tabela no subcluster especificado.

Funções estatísticas aproximadas no nível da tupla

`rds_aurora.limitless_pgstattuple_approx(relation_name)`

- Objetivo: extrair estatísticas aproximadas no nível da tupla para tabelas padrão e seus índices
- Entrada: `relation_name` (texto) — o nome da relação
- Saída: colunas consistentes com as retornadas pela função pgstattuple_approx no Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple_approx(relation_name, subcluster_id)`

- Objetivo: extrair estatísticas aproximadas no nível da tupla em tabelas de referência, tabelas fragmentadas e tabelas de catálogo (incluindo seus índices)
- Entrada:
  - `relation_name` (texto) — o nome da relação
  - `subcluster_id` (texto) — o ID do subcluster do nó de onde as estatísticas devem ser extraídas
- Saída:
  - Em tabelas de referência e de catálogo (incluindo seus índices), as colunas são consistentes com as do Aurora PostgreSQL.
  - Em tabelas fragmentadas, as estatísticas representam somente a partição da tabela fragmentada que reside no subcluster especificado.

> **Note**
>
> Atualmente, o Aurora PostgreSQL Limitless Database não oferece suporte à extensão `pgstattuple` em visões materializadas, tabelas TOAST ou tabelas temporárias.
> No Aurora PostgreSQL Limitless Database, você deve fornecer a entrada como texto. O Aurora PostgreSQL oferece suporte a outros formatos.

Chaves externas

Há suporte para restrições de chave externa (`FOREIGN KEY`) com algumas limitações:

- Há suporte para `CREATE TABLE` com `FOREIGN KEY` somente para tabelas padrão. Para criar uma tabela fragmentada ou de referência com `FOREIGN KEY`, primeiro crie a tabela sem uma restrição de chave externa. Em seguida, modifique-a usando a seguinte instrução:

  ```
  ALTER TABLE ADD CONSTRAINT;
  ```

- Não há suporte para a conversão de uma tabela padrão em uma tabela fragmentada ou de referência quando a tabela tem uma restrição de chave externa. Descarte a restrição e adicione-a após a conversão.
- As seguintes limitações se aplicam aos tipos de tabela para restrições de chave externa:
  - Uma tabela padrão pode ter uma restrição de chave externa para outra tabela padrão.
  - Uma tabela fragmentada pode ter uma restrição de chave externa se as tabelas pai e filha tiverem localização compartilhada e a chave externa for um superconjunto da chave de fragmento.
  - Uma tabela fragmentada pode ter uma restrição de chave externa para uma tabela de referência.
  - Uma tabela de referência pode ter uma restrição de chave externa para outra tabela de referência.

Tópicos

- Opções de chave externa
- Exemplos

Opções de chave externa

Há suporte para chaves externas no Aurora PostgreSQL Limitless Database para algumas opções de DDL. A tabela a seguir lista as opções com suporte e sem suporte entre as tabelas do Aurora PostgreSQL Limitless Database.

Opção de DDL	Referência para referência	Fragmentada para fragmenta da (colocalizado)	Fragmentada para referência	Padrão para padrão
`DEFERRABLE`	Sim	Sim	Sim	Sim
`INITIALLY DEFERRED`	Sim	Sim	Sim	Sim
`INITIALLY IMMEDIATE`	Sim	Sim	Sim	Sim
`MATCH FULL`	Sim	Sim	Sim	Sim
`MATCH PARTIAL`	Não	Não	Não	Não
`MATCH SIMPLE`	Sim	Sim	Sim	Sim
`NOT DEFERRABLE`	Sim	Sim	Sim	Sim
`NOT VALID`	Sim	Não	Não	Sim
`ON DELETE CASCADE`	Sim	Sim	Sim	Sim
`ON DELETE NO ACTION`	Sim	Sim	Sim	Sim
`ON DELETE RESTRICT`	Sim	Sim	Sim	Sim
`ON DELETE SET DEFAULT`	Não	Não	Não	Não
`ON DELETE SET NULL`	Sim	Não	Não	Sim

Opção de DDL	Referência para referência	Fragmentada para fragmenta da (colocalizado)	Fragmentada para referência	Padrão para padrão
`ON UPDATE CASCADE`	Não	Não	Não	Sim
`ON UPDATE NO ACTION`	Sim	Sim	Sim	Sim
`ON UPDATE RESTRICT`	Sim	Sim	Sim	Sim
`ON UPDATE SET DEFAULT`	Não	Não	Não	Não
`ON UPDATE SET NULL`	Sim	Não	Não	Sim

Exemplos

- Padrão para padrão:

```
set rds_aurora.limitless_create_table_mode='standard';

CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
    quantity integer
);

SELECT constraint_name, table_name, constraint_type
FROM information_schema.table_constraints WHERE constraint_type='FOREIGN KEY';
```

```
 constraint_name         | table_name  | constraint_type
-------------------------+-------------+-----------------
 orders_product_no_fkey  | orders      | FOREIGN KEY
(1 row)
```

- Fragmentada para fragmentada (colocalizado):

```
set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"product_no"}';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
set rds_aurora.limitless_create_table_collocate_with='products';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Fragmentada para referência:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);
```

```
ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Referência para referência:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

Funções

As funções são compatíveis no Aurora PostgreSQL Limitless Database.

As seguintes DDLs são compatíveis com funções:

CREATE FUNCTION

Assim como no Aurora PostgreSQL, é possível criar funções, com exceção de alterar sua volatilidade ao substituí-las.

Para obter mais informações, consulte CREATE FUNCTION na documentação do PostgreSQL.

ALTER FUNCTION

Assim como no Aurora PostgreSQL, é possível alterar funções, com exceção de alterar sua volatilidade.

Para obter mais informações, consulte ALTER FUNCTION na documentação do PostgreSQL.

DROP FUNCTION

Assim como no Aurora PostgreSQL, é possível descartar funções.

```
DROP FUNCTION [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]
 [, ...]
    [ CASCADE | RESTRICT ]
```

Para obter mais informações, consulte DROP FUNCTION na documentação do PostgreSQL.

Tópicos

- Distribuição de funções
- Volatilidade da função

Distribuição de funções

Quando todas as instruções de uma função são direcionadas a um único fragmento, é vantajoso enviar a função inteira para o fragmento de destino. Em seguida, o resultado é propagado de volta ao roteador, em vez de descompactar a função no próprio roteador. O recurso de envio de

funções e procedimentos armazenados é útil para clientes que desejam executar sua função ou seu procedimento armazenado mais próximo à fonte de dados, ou seja, o fragmento.

Para distribuir uma função, primeiro crie a função e, em seguida, chame o procedimento `rds_aurora.limitless_distribute_function` para distribuí-la. Esta função usa a seguinte sintaxe:

```
SELECT rds_aurora.limitless_distribute_function('function_prototype',
 ARRAY['shard_key'], 'collocating_table');
```

A função usa os seguintes parâmetros:

- *`function_prototype`*: a função a ser distribuída. Mencione apenas os argumentos de entrada e nenhum dos argumentos de saída.

  Se algum dos argumentos for definido como parâmetros `OUT`, não inclua seu tipo nos argumentos de `function_prototype`.
- `ARRAY['`*`shard_key`*`']`: a lista de argumentos da função identificados como a chave de fragmento da função.
- *`collocating_table`*: a tabela fragmentada que contém o intervalo de dados no fragmento de destino.

Para identificar o fragmento no qual inserir essa função para execução, o sistema pega o argumento `ARRAY['`*`shard_key`*`']`, faz o hash e encontra o fragmento de *`collocating_table`* que hospeda o intervalo que contém esse valor de hash.

Restrições

Ao distribuir uma função ou um procedimento, ele lida apenas com dados limitados pelo intervalo de chaves de fragmento nesse fragmento. Nos casos em que a função ou o procedimento tentar acessar dados de um fragmento diferente, os resultados retornados pela função ou pelo procedimento distribuído serão diferentes em comparação com um que não está distribuído.

Por exemplo, você cria uma função que contém consultas que têm relação com vários fragmentos, mas depois chama o procedimento `rds_aurora.limitless_distribute_function` para distribuí-la. Quando você invoca essa função fornecendo argumentos para uma chave de fragmento, é provável que os resultados dessa execução sejam limitados pelos valores presentes nesse fragmento. Esses resultados são diferentes dos que são produzidos sem a distribuição da função.

Exemplos

Considere a função a seguir, `func` em que temos a tabela fragmentada `customers` com a chave de fragmento `customer_id`.

```
postgres_limitless=> CREATE OR REPLACE FUNCTION func(c_id integer, sc integer)
 RETURNS int
    language SQL
    volatile
    AS $$
    UPDATE customers SET score = sc WHERE customer_id = c_id RETURNING score;
    $$;
```

Agora, distribuímos essa função:

```
SELECT rds_aurora.limitless_distribute_function('func(integer, integer)',
 ARRAY['c_id'], 'customers');
```

Veja a seguir exemplos de planos de consulta.

```
EXPLAIN(costs false, verbose true) SELECT func(27+1,10);

                    QUERY PLAN
 ---------------------------------------------------
  Foreign Scan
    Output: (func((27 + 1), 10))
    Remote SQL:  SELECT func((27 + 1), 10) AS func
  Single Shard Optimized
 (4 rows)
```

```
EXPLAIN(costs false, verbose true)
 SELECT * FROM customers,func(customer_id, score) WHERE customer_id=10 AND score=27;

                          QUERY PLAN
 ------------------------------------------------------------------------
  Foreign Scan
    Output: customer_id, name, score, func
    Remote SQL:  SELECT customers.customer_id,
      customers.name,
      customers.score,
      func.func
```

```
     FROM public.customers,
      LATERAL func(customers.customer_id, customers.score) func(func)
    WHERE ((customers.customer_id = 10) AND (customers.score = 27))
  Single Shard Optimized
 (10 rows)
```

O exemplo a seguir mostra um procedimento com os parâmetros `IN` e `OUT` como argumentos.

```
CREATE OR REPLACE FUNCTION get_data(OUT id INTEGER, IN arg_id INT)
    AS $$
    BEGIN
        SELECT customer_id,
        INTO id
        FROM customer
        WHERE customer_id = arg_id;
    END;
    $$ LANGUAGE plpgsql;
```

O exemplo a seguir distribui o procedimento usando somente parâmetros `IN`.

```
EXPLAIN(costs false, verbose true) SELECT * FROM get_data(1);

             QUERY PLAN
 -----------------------------------
  Foreign Scan
    Output: id
    Remote SQL:  SELECT customer_id
     FROM get_data(1) get_data(id)
  Single Shard Optimized
 (6 rows)
```

Volatilidade da função

É possível determinar se uma função é imutável, estável ou volátil verificando o valor `provolatile` na exibição pg_proc. O valor `provolatile` indica se o resultado da função depende somente de seus argumentos de entrada ou se é afetado por fatores externos.

O valor é um dos seguintes:

- `i`: funções imutáveis, que sempre fornecem o mesmo resultado para as mesmas entradas
- `s`: funções estáveis, cujos resultados (para entradas fixas) não mudam em uma varredura

- v: funções voláteis, cujos resultados podem mudar a qualquer momento. Use também v para funções com efeitos colaterais, para que as chamadas para elas não possam ser otimizadas.

Os exemplos a seguir mostram funções voláteis.

```
SELECT proname, provolatile FROM pg_proc WHERE proname='pg_sleep';

 proname  | provolatile
----------+-------------
 pg_sleep | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='uuid_generate_v4';

     proname      | provolatile
------------------+-------------
 uuid_generate_v4 | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='nextval';

 proname | provolatile
---------+-------------
 nextval | v
(1 row)
```

Não há suporte para alterar a volatilidade de uma função existente no Aurora PostgreSQL Limitless Database. Isso se aplica aos comandos `ALTER FUNCTION` e `CREATE OR REPLACE FUNCTION`, conforme mostrado nos exemplos a seguir.

```
-- Create an immutable function
CREATE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$IMMUTABLE;

-- Altering the volatility throws an error
ALTER FUNCTION immutable_func1 STABLE;

-- Replacing the function with altered volatility throws an error
```

```
CREATE OR REPLACE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$VOLATILE;
```

É altamente recomendável atribuir as volatilidades corretas às funções. Por exemplo, se sua função usa `SELECT` de várias tabelas ou faz referência a objetos de banco de dados, não a defina como `IMMUTABLE`. Se o conteúdo da tabela mudar, a imutabilidade será quebrada.

O Aurora PostgreSQL permite `SELECT` dentro de funções imutáveis, mas os resultados podem ser incorretos. O Aurora PostgreSQL Limitless Database pode retornar tanto erros quanto resultados incorretos. Para obter mais informações sobre volatilidade de funções, consulte Function volatility categories na documentação do PostgreSQL.

Sequências

As sequências nomeadas são objetos do banco de dados que geram números exclusivos em ordem crescente ou decrescente. `CREATE SEQUENCE` cria um gerador de números de sequência. Os valores da sequência são exclusivos.

Ao criar uma sequência nomeada no Aurora PostgreSQL Limitless Database, um objeto de sequência distribuída é criado. Em seguida, o Aurora PostgreSQL Limitless Database distribui partes não sobrepostas de valores de sequência em todos os Roteadores de transações distribuídas (roteadores). Essas partes são representadas como objetos de sequência local em roteadores; portanto, operações de sequência, como `nextval` e `currval`, são executadas localmente. Os roteadores operam de forma independente e solicitam novas partes da sequência distribuída quando necessário.

Para obter mais informações sobre sequências, consulte CREATE SEQUENCE na documentação do PostgreSQL.

Tópicos

- Solicitar uma nova parte
- Limitações
- Opções não compatíveis
- Exemplos
- Exibições de sequências
- Solucionar problemas de sequência

Solicitar uma nova parte

É possível configurar o tamanho das partes alocadas nos roteadores usando o parâmetro `rds_aurora.limitless_sequence_chunk_size`. O valor padrão é `250000`. Cada roteador possui inicialmente duas partes: ativa e reservada. As partes ativas são usadas para configurar objetos de sequência local (definindo `minvalue` e `maxvalue`), e as partes reservadas são armazenadas em uma tabela de catálogo interna. Quando uma parte ativa atinge o valor mínimo ou máximo, ela é substituída pela parte reservada. Para fazer isso, `ALTER SEQUENCE` é usado internamente, o que significa que `AccessExclusiveLock` é adquirido.

Os processos em segundo plano são executados a cada 10 segundos nos nós do roteador para verificar as sequências em busca de partes reservadas usadas. Se uma parte usada for encontrada,

o processo solicitará uma nova parte da sequência distribuída. Certifique-se de definir o tamanho da parte grande o suficiente para que os processos em segundo plano tenham tempo suficiente para solicitar novas partes. As solicitações remotas nunca acontecem no contexto das sessões do usuário, o que significa que não é possível solicitar uma nova sequência diretamente.

Limitações

As seguintes limitações se aplicam às sequências no Aurora PostgreSQL Limitless Database:

- O catálogo `pg_sequence`, a função `pg_sequences` e a instrução `SELECT * FROM` *`sequence_name`* mostram apenas o estado da sequência local, não o estado distribuído.
- Os valores de sequência têm garantia de serem únicos e monotônicos dentro de uma sessão. Mas eles podem estar fora de ordem com instruções `nextval` executadas em outras sessões, se essas sessões estiverem conectadas a outros roteadores.
- Verifique se o tamanho da sequência (número de valores disponíveis) é grande o suficiente para ser distribuído entre todos os roteadores. Use o parâmetro `rds_aurora.limitless_sequence_chunk_size` para configurar o `chunk_size`. (Cada roteador tem duas partes.)
- Há suporte para a opção `CACHE`, mas o cache deve ser menor que o `chunk_size`.

Opções não compatíveis

As opções a seguir não são compatíveis com sequências no Aurora PostgreSQL Limitless Database.

Funções de manipulação de sequências

A função `setval` não é compatível. Para obter mais informações, consulte Sequence Manipulation Functions na documentação do PostgreSQL.

CREATE SEQUENCE

As opções a seguir não são compatíveis.

```
CREATE [{ TEMPORARY | TEMP} | UNLOGGED] SEQUENCE
    [[ NO ] CYCLE]
```

Para ter mais informações, consulte CREATE SEQUENCE na documentação do PostgreSQL.

ALTER SEQUENCE

As opções a seguir não são compatíveis.

```
ALTER SEQUENCE
    [[ NO ] CYCLE]
```

Para obter mais informações, consulte ALTER SEQUENCE na documentação do PostgreSQL.

ALTER TABLE

O comando ALTER TABLE não é compatível com sequências.

Exemplos

CREATE/DROP SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
       1
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16960 |       20 |        1 |            1 |  10000 |      1 |        1 | f
(1 row)

% connect to another router
postgres_limitless=> SELECT nextval('s');

 nextval
---------
   10001
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;
```

```
 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16959 |       20 |    10001 |            1 |  20000 |  10001 |        1 | f
(1 row)

postgres_limitless=> DROP SEQUENCE s;
DROP SEQUENCE
```

ALTER SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> ALTER SEQUENCE s RESTART 500;
ALTER SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
     500
(1 row)

postgres_limitless=> SELECT currval('s');

 currval
---------
     500
(1 row)
```

Funções de manipulação de sequências

```
postgres=# CREATE TABLE t(a bigint primary key, b bigint);
CREATE TABLE

postgres=# CREATE SEQUENCE s minvalue 0 START 0;
CREATE SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1
```

```
postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));
INSERT 0 1

postgres=# SELECT * FROM t;

 a | b
---+---
 0 | 0
 1 | 1
(2 rows)

postgres=# ALTER SEQUENCE s RESTART 10000;
ALTER SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# SELECT * FROM t;

   a   |   b
-------+-------
     0 |     0
     1 |     1
 10000 | 10000
(3 rows)
```

Exibições de sequências

O Aurora PostgreSQL Limitless Database fornece as seguintes exibições para sequências.

rds_aurora.limitless_distributed_sequence

Esta exibição mostra o estado e a configuração de uma sequência distribuída. As colunas `minvalue`, `maxvalue`, `start`, `inc` e `cache` têm o mesmo significado que na exibição pg_sequences e mostram as opções com as quais a sequência foi criada. A coluna `lastval` mostra o último valor alocado ou reservado em um objeto de sequência distribuída. Isso não significa que o valor já tenha sido usado, pois os roteadores mantêm partes da sequência localmente.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_distributed_sequence WHERE
 sequence_name='test_serial_b_seq';

 schema_name |   sequence_name   | lastval | minvalue |  maxvalue  | start | inc |
 cache
-------------+-------------------+---------+----------+------------+-------+-----
+-------
 public      | test_serial_b_seq | 1250000 |        1 | 2147483647 |     1 |   1 |
   1
(1 row)
```

rds_aurora.limitless_sequence_metadata

Esta exibição mostra metadados de sequência distribuídos e agrega metadados de sequência dos nós do cluster. Ela usa as seguintes colunas:

- `subcluster_id`: o ID do nó do cluster que possui uma parte.
- Parte ativa: uma parte de uma sequência que está sendo usada (`active_minvalue`, `active_maxvalue`).
- Parte reservada: a parte local que será usada em seguida (`reserved_minvalue`, `reserved_maxvalue`).
- `local_last_value`: o último valor observado de uma sequência local.
- `chunk_size`: o tamanho de uma parte, conforme configurado na criação.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_sequence_metadata WHERE
 sequence_name='test_serial_b_seq' order by subcluster_id;
```

```
 subcluster_id |   sequence_name   | schema_name | active_minvalue | active_maxvalue
 | reserved_minvalue | reserved_maxvalue | chunk_size | chunk_state |
 local_last_value
---------------+-------------------+-------------+-----------------
+-----------------+-------------------+-------------------+------------
+-------------+------------------
 1             | test_serial_b_seq | public      |          500001 |          750000
 |           1000001 |           1250000 |     250000 |           1 |
 550010
 2             | test_serial_b_seq | public      |          250001 |          500000
 |            750001 |           1000000 |     250000 |           1 |

(2 rows)
```

Solucionar problemas de sequência

Os problemas a seguir podem ocorrer em sequências.

Tamanho da parte não é grande o suficiente

Se o tamanho da parte não for grande o suficiente e a taxa de transação for alta, os processos em segundo plano talvez não tenham tempo suficiente para solicitar novas partes antes que as partes ativas se esgotem. Isso pode levar a eventos de contenção e espera, como `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` e `LWlock:bufferscontent`.

Aumente o valor do parâmetro `rds_aurora.limitless_sequence_chunk_size`.

Cache de sequência definido como muito alto

No PostgreSQL, o armazenamento em cache da sequência ocorre no nível da sessão. Cada sessão aloca valores de sequência sucessivos durante um acesso ao objeto de sequência e aumenta o `last_value` do objeto de sequência adequadamente. Em seguida, os próximos usos de `nextval` dentro dessa sessão simplesmente retornam os valores pré-alocados, sem tocar no objeto de sequência.

Quaisquer números alocados, mas não usados em uma sessão, são perdidos quando essa sessão termina, resultando em "lacunas" na sequência. Isso pode consumir o sequence_chunk rapidamente e levar a eventos de contenção e espera, como `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` e `LWlock:bufferscontent`.

Reduza a configuração do cache de sequência.

A figura a seguir mostra eventos de espera causados por problemas de sequência.

Comandos SQL de Linguagem de Manipulação de Dados (DML) e processamento de consultas com e sem suporte

A tabela a seguir lista os comandos de DML com e sem suporte do Aurora PostgreSQL Limitless Database, com referências a limitações ou mais informações.

Command	Compatível?	Limitações ou mais informaçõ es
ABORT	Sim	Nenhum
ANALYZE	Sim	ANALYZE
BEGIN	Sim	Nenhum
CALL	Sim	Nenhum
CHECKPOINT	Sim	Nenhum
CLOSE	Sim	Nenhum
CLUSTER	Sim	CLUSTER
COMMIT	Sim	Nenhum
COMMIT PREPARED	Não	Não aplicável
COPY	Sim	Nenhum
DEALLOCATE	Sim	Nenhum
DECLARE	Sim	Nenhum
DELETE	Sim	Nenhum
DISCARD	Sim	Nenhum
DO	Sim	Nenhum
END	Sim	Nenhum

Command	Compatível?	Limitações ou mais informaçõ es
EXECUTE	Sim	Nenhum
EXPLAIN	Sim	EXPLAIN
FETCH	Sim	Nenhum
IMPORT FOREIGN SCHEMA	Não	Não aplicável
INSERT	Sim	INSERT
LISTEN	Não	Não aplicável
LOCK	Sim	Nenhum
MERGE	Não	Não aplicável
MOVE	Sim	Nenhum
NOTIFY	Não	Não aplicável
OPEN	Sim	Nenhum
PREPARE	Sim	Nenhum
PREPARE TRANSACTION	Não	Não aplicável
REFRESH MATERIALIZED VIEW	Não	Não aplicável
REINDEX	Não	Não aplicável
RELEASE SAVEPOINT	Sim	Nenhum
ROLLBACK	Sim	Nenhum
ROLLBACK PREPARED	Não	Não aplicável
ROLLBACK TO SAVEPOINT	Sim	Nenhum

Command	Compatível?	Limitações ou mais informaçõ es
SAVEPOINT	Sim	Nenhum
SELECT	Sim	Nenhum
SELECT INTO	Sim	Nenhum
SHOW	Sim	Nenhum
START TRANSACTION	Sim	Nenhum
UNLISTEN	Não	Nenhum
UPDATE	Sim	UPDATE
UPDATE ... WHERE CURRENT OF	Não	Não aplicável
VACUUM	Sim	VACUUM
VALUES	Sim	Nenhum

Limitações e outras informações de DML no Aurora PostgreSQL Limitless Database

Os tópicos a seguir descrevem as limitações ou fornecem mais informações sobre os comandos SQL de DML e processamento de consultas no Aurora PostgreSQL Limitless Database.

Tópicos

- ANALYZE
- CLUSTER
- EXPLAIN
- INSERT
- UPDATE
- VACUUM

ANALYZE

O comando `ANALYZE` coleta estatísticas sobre o conteúdo das tabelas no banco de dados. Posteriormente, o planejador de consultas usa essas estatísticas para ajudar a determinar os planos de execução mais eficientes para as consultas. Para obter mais informações, consulte ANALYZE na documentação do PostgreSQL.

No Aurora PostgreSQL Limitless Database, o comando `ANALYZE` coleta estatísticas da tabela em todos os roteadores e fragmentos quando é executado.

Para evitar o cálculo de estatísticas em cada roteador durante as execuções de `ANALYZE`, as estatísticas da tabela são calculadas em um dos roteadores e depois copiadas para os roteadores de mesmo nível.

CLUSTER

O comando `CLUSTER` reordena fisicamente uma tabela com base em um índice. O índice já deve ter sido definido na tabela. No Aurora PostgreSQL Limitless Database, o clustering é local para a parte do índice que está presente em cada fragmento.

Para ter mais informações, consulte CLUSTER na documentação do PostgreSQL.

EXPLAIN

Você usa o seguinte parâmetro para configurar a saída do comando EXPLAIN:

- `rds_aurora.limitless_explain_options`: o que incluir na saída de EXPLAIN. O valor padrão é `single_shard_optimization`: é mostrado se os planos são otimizados para fragmento único, mas os planos de fragmento não estão incluídos.

Neste exemplo, a saída de EXPLAIN não mostra planos de fragmentos.

```
postgres_limitless=> EXPLAIN SELECT * FROM employees where id =25;

                      QUERY PLAN
------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
 Single Shard Optimized
(2 rows)
```

Agora, configuramos `rds_aurora.limitless_explain_options` para incluir `shard_plans` e `single_shard_optimization`. Podemos ver os planos de execução das instruções em roteadores e fragmentos. Além disso, desativamos o parâmetro `enable_seqscan` para garantir que a varredura de índice seja usada na camada de fragmento.

```
postgres_limitless=> SET rds_aurora.limitless_explain_options = shard_plans,
 single_shard_optimization;
SET

postgres_limitless=> SET enable_seqscan = OFF;
SET

postgres_limitless=> EXPLAIN SELECT * FROM employees WHERE id = 25;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Remote Plans from Shard postgres_s4:
         Index Scan using employees_ts00287_id_idx on employees_ts00287
 employees_fs00003  (cost=0.14..8.16 rows=1 width=15)
           Index Cond: (id = 25)
 Single Shard Optimized
(5 rows)
```

Para obter mais informações sobre o comando EXPLAIN, consulte EXPLAIN na documentação do PostgreSQL.

INSERT

A maioria dos comandos INSERT é compatível com o Aurora PostgreSQL Limitless Database.

O PostgreSQL não tem um comando UPSERT explícito, mas é compatível com as instruções INSERT ... ON CONFLICT.

INSERT ... ON CONFLICT não é compatível se a ação de conflito tiver uma subconsulta ou uma função mutável:

```
-- RANDOM is a mutable function.
INSERT INTO sharded_table VALUES (1, 100) ON CONFLICT (id) DO UPDATE SET other_id =
 RANDOM();

ERROR: Aurora Limitless Tables doesn't support pushdown-unsafe functions with DO UPDATE
 clauses.
```

Para obter mais informações sobre o comando INSERT, consulte INSERT na documentação do PostgreSQL.

UPDATE

A atualização da chave de fragmento não é compatível. Por exemplo, você tem uma tabela fragmentada chamada customers, com uma chave de fragmento customer_id. As seguintes instruções DML causam erros:

```
postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_id =1;
ERROR:  Shard key column update is not supported

postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_name='abc';
ERROR:  Shard key column update is not supported
```

Para atualizar uma chave de fragmento, primeiro aplique DELETE à linha com a chave de fragmento e, em seguida, aplique INSERT a uma nova linha com o valor atualizado da chave de fragmento.

Para obter mais informações sobre o comando UPDATE, consulte Updating data na documentação do PostgreSQL.

VACUUM

É possível realizar a limpeza em tabelas fragmentadas e de referência. As seguintes funções `VACUUM` são totalmente compatíveis com o Aurora PostgreSQL Limitless Database:

- VACUUM
- ANALYZE
- DISABLE_PAGE_SKIPPING
- FREEZE
- FULL
- INDEX_CLEANUP
- PARALLEL
- PROCESS_TOAST
- TRUNCATE
- VERBOSE

`VACUUM` no Aurora PostgreSQL Limitless Database tem as seguintes limitações:

- A extensão pg_visibility_map não é compatível.
- A verificação de índices não utilizados com a visualização pg_stat_all_indexes não é compatível.
- As visualizações consolidadas para pg_stat_user_indexes, pg_class e pg_stats não foram implementadas.

Para obter mais informações sobre o comando `VACUUM`, consulte VACUUM na documentação do PostgreSQL. Para obter mais informações sobre como a limpeza funciona no Aurora PostgreSQL Limitless Database, consulte Recuperar espaço de armazenamento por meio de limpeza.

Variáveis no Aurora PostgreSQL Limitless Database

Você pode usar as seguintes variáveis para configurar o Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_active_shard_key

Define uma única chave de fragmento ao consultar o banco de dados, fazendo com que todas as consultas `SELECT` e DML sejam anexadas com a chave de fragmento como um predicado constante. Para ter mais informações, consulte Configurar uma chave de fragmento ativa.

rds_aurora.limitless_create_table_collocate_with

Defina essa variável com um nome de tabela específico para colocar tabelas recém-criadas com essa tabela. Para ter mais informações, consulte Criação de tabelas ilimitadas usando variáveis.

rds_aurora.limitless_create_table_mode

Define o modo de criação da tabela. Para ter mais informações, consulte Criação de tabelas ilimitadas usando variáveis.

rds_aurora.limitless_create_table_shard_key

Defina essa variável como uma matriz de nomes de colunas para usar como chaves de fragmentação. Para ter mais informações, consulte Criação de tabelas ilimitadas usando variáveis.

rds_aurora.limitless_explain_options

O que incluir na saída de `EXPLAIN`. Para ter mais informações, consulte EXPLAIN.

Parâmetros do cluster de banco de dados no Aurora PostgreSQL Limitless Database

Você pode usar os seguintes parâmetros de cluster de banco de dados para configurar o Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_adaptive_fetch_size

Melhora a pré-busca em lote. Quando definido como `true`, esse parâmetro permite um tamanho de busca autoajustável (adaptável) para pré-busca. Quando definido como `false`, o tamanho da busca é constante.

rds_aurora.limitless_auto_scale_options

Define as opções disponíveis para adicionar roteadores ou dividir fragmentos em um grupo de fragmentos de banco de dados. O valor pode ser `add_router`, `split_shard` ou os dois.

Para ter mais informações, consulte Adicionar um roteador a um grupo de fragmentos de banco de dados e Dividir um fragmento em um grupo de fragmentos de banco de dados.

rds_aurora.limitless_distributed_deadlock_timeout

O tempo de espera em um bloqueio antes de verificar se há uma condição de deadlock distribuído, em milissegundos. O padrão é `1000` (1 segundo).

Para ter mais informações, consulte Deadlocks distribuídos no Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_enable_auto_scale

Permite a adição de roteadores e a divisão de fragmentos em um grupo de fragmentos de banco de dados.

Para ter mais informações, consulte Adicionar um roteador a um grupo de fragmentos de banco de dados e Dividir um fragmento em um grupo de fragmentos de banco de dados.

rds_aurora.limitless_finalize_split_shard_mode

Determina como as divisões de fragmentos iniciadas pelo sistema são finalizadas. Para ter mais informações, consulte Dividir um fragmento em um grupo de fragmentos de banco de dados.

rds_aurora.limitless_maximum_adaptive_fetch_size

Define o limite máximo para o tamanho de busca adaptável. O intervalo é de `1–INT_MAX`. O padrão é `1000`.

Usar o Amazon Aurora Global Database

Com o recurso Amazon Aurora Global Database, você configura vários clusters de banco de dados do Aurora que abrangem várias Regiões da AWS. O Aurora sincroniza automaticamente todas as alterações feitas no cluster de banco de dados primário com um ou mais clusters secundários. Um Aurora Global Database tem um cluster de banco de dados primário em uma região e até dez clusters de banco de dados secundários em diferentes regiões. Essa configuração de várias regiões oferece recuperação rápida de qualquer interrupção incomum que possa afetar uma Região da AWS inteira. Ter uma cópia completa de todos os dados em várias localizações geográficas também viabiliza operações de leitura de baixa latência em aplicações que se conectam de locais extremamente separados em todo o mundo.

Tópicos

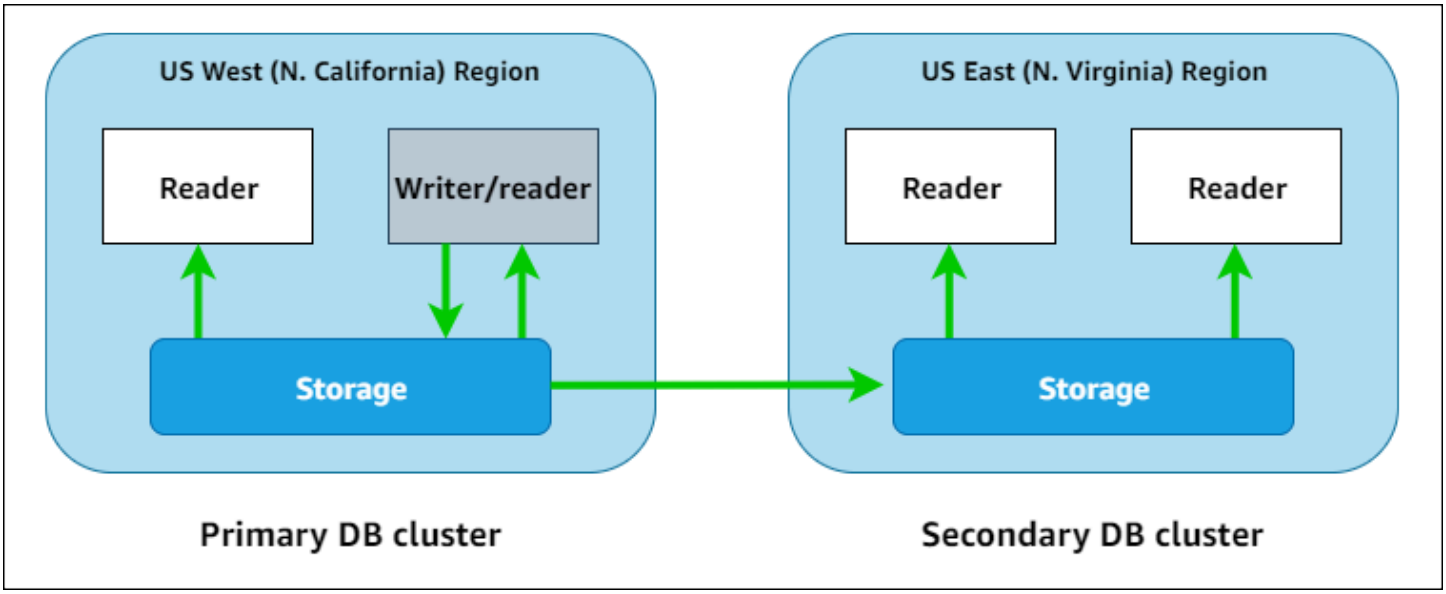
- Visão geral do Amazon Aurora Global Database
- Vantagens do Amazon Aurora Global Database
- Disponibilidade de regiões e versões
- Limitações do Amazon Aurora Global Database
- Conceitos básicos do Amazon Aurora Global Database
- Gerenciar um banco de dados global do Amazon Aurora
- Conectar-se ao Amazon Aurora Global Database
- Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora
- Usar a transição ou o failover no Amazon Aurora Global Database
- Monitorar um banco de dados global do Amazon Aurora
- Usar implantações azul/verde para o Amazon Aurora Global Database
- Usar bancos de dados globais do Amazon Aurora com outros produtos da AWS
- Atualizar um Amazon Aurora Global Database

Visão geral do Amazon Aurora Global Database

Ao usar o recurso Amazon Aurora Global Database, é possível executar as aplicações distribuídas globalmente usando um único banco de dados do Aurora que abrange várias Regiões da AWS.

Um Aurora Global Database consiste em uma Região da AWS principal, onde seus dados são utilizados, e até dez Regiões da AWS secundárias apenas para leitura. Envie operações de gravação ao cluster de banco de dados primário na Região da AWS primária. A maneira mais conveniente de fazer isso é conectar-se ao endpoint do gravador do Aurora Global Database, que sempre aponta para o cluster de banco de dados primário, mesmo após uma transição ou um failover para outra Região da AWS. Após qualquer operação, o Aurora replica dados para as Regiões da AWS secundárias que usam infraestrutura exclusiva, com latência normalmente inferior a um segundo.

O diagrama a seguir mostra um exemplo de Aurora Global Database que abrange duas Regiões da AWS.



É possível aumentar a escala verticalmente de cada cluster secundário de maneira independente, adicionando uma ou mais instâncias do leitor do Aurora para atender a workloads somente leitura. É possível usar o Aurora Serverless v2 para as instâncias do leitor a fim de ter uma escalabilidade ainda mais granular e flexível.

Somente o cluster primário realiza operações de gravação. Os clientes que realizam operações de gravação se conectam ao endpoint do gravador do Aurora Global Database, que sempre aponta para a instância de banco de dados do gravador do cluster primário. Como mostrado no diagrama, o Aurora usa o volume de armazenamento de cluster e não o mecanismo de banco de dados para replicação rápida e de baixo custo. Para saber mais, consulte Visão geral do armazenamento do Amazon Aurora.

O Aurora Global Database é projetado para aplicações com uma presença mundial. Os clusters de banco de dados secundários somente leitura em várias Regiões da AWS ajudam a otimizar operações de leitura mais próximas dos usuários da aplicação. Ao usar o recurso de

encaminhamento de gravação, você também pode configurar um banco de dados global para que os clusters secundários enviem solicitações de gravação ao primário. Para obter mais informações, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

O Aurora Global Database oferece suporte a duas operações diferentes para alterar a região do cluster de banco de dados primário, dependendo do cenário: transição do Aurora Global Database e failover do Aurora Global Database.

- Para procedimentos operacionais planejados, como troca regional, use o mecanismo de transição (anteriormente chamado de “failover planejado gerenciado”). Com esse recurso, é possível realocar o cluster primário de um Aurora Global Database íntegro para uma de suas regiões secundárias sem perda de dados. Para saber mais, consulte Realizar transições para o Amazon Aurora Global Database.
- Para recuperar o Aurora Global Database após uma interrupção na região primária, use o mecanismo de failover. Com esse recurso, você realiza um failover do cluster de banco de dados primário para outra região (failover entre regiões). Para saber mais, consulte Failovers planejados gerenciados para o Aurora Global Database.

Vantagens do Amazon Aurora Global Database

Ao usar o Aurora Global Database, você pode ter as seguintes vantagens:

- Leituras globais com latência local: se você tem escritórios em todo o mundo, pode usar o Aurora Global Database para manter suas principais fontes de informações atualizadas na Região da AWS primária. Escritórios em outras regiões podem acessar as informações em sua própria região com latência local.
- Clusters de banco de dados secundários do Aurora escaláveis: é possível escalar os clusters secundários adicionando mais instâncias somente leitura a uma Região da AWS secundária. O cluster secundário é somente leitura, portanto, é compatível com até 16 instâncias de banco de dados somente leitura em vez do limite normal de 15 para um único cluster do Aurora.
- Replicação rápida de clusters de banco de dados do Aurora primários para secundários: a replicação realizada pelo Aurora Global Database tem pouco impacto no desempenho do cluster de banco de dados primário. Os recursos das instâncias de banco de dados são totalmente dedicados para atender as workloads de leitura e gravação.
- Recuperação de interrupções em toda a região: os clusters secundários permitem que você disponibilize um Aurora Global Database em uma nova Região da AWS primária mais depressa

(menor RTO) e com menor perda de dados (menor RPO) do que as soluções tradicionais de replicação.

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e de regiões com o Aurora Global Database, consulte Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.

Limitações do Amazon Aurora Global Database

No momento, as seguintes limitações se aplicam ao Aurora Global Database:

- O Aurora Global Database está disponível em determinadas Regiões da AWS e para versões específicas do Aurora MySQL e do Aurora PostgreSQL. Para obter mais informações, consulte Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.
- O Aurora Global Database tem requisitos específicos de configuração para classes de instância de banco de dados do Aurora aceitas, número máximo de Regiões da AWS etc. Para obter mais informações, consulte Requisitos de configuração do banco de dados do Amazon Aurora global.
- Para compatibilidade do Aurora MySQL com MySQL 5.7, as transições do Aurora Global Database exigem a versão 2.09.1 ou uma versão secundária posterior.
- Só é possível realizar transições ou failovers gerenciados entre regiões com o Aurora Global Database caso os clusters de banco de dados primário e secundário tenham as mesmas versões principal e secundária. Dependendo do mecanismo e de suas respectivas versões, os níveis de patch podem precisar ser idênticos ou podem ser diferentes. Para conferir uma lista de mecanismos e suas respectivas versões que permitem essas operações entre clusters primários e secundários com diferentes níveis de patch, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões. Se as versões do mecanismo exigirem níveis de patch idênticos, será possível realizar o failover manualmente seguindo as etapas em Realizar failovers manuais para bancos de dados globais do Aurora.
- O Aurora Global Database ainda não oferece suporte aos seguintes recursos do Aurora:
  - Aurora Serverless v1
  - Retroceder no Aurora

- Para saber as limitações ao uso do recurso RDS Proxy com o Aurora Global Database, consulte Limitações do RDS Proxy com bancos de dados globais.
- A atualização automática da versão secundária não se aplica a clusters do Aurora MySQL e do Aurora PostgreSQL que fazem parte de um banco de dados global. Observe que você pode especificar essa configuração para uma instância de banco de dados que faz parte de um cluster de banco de dados global, mas a configuração não tem qualquer efeito.
- O Aurora Global Database ainda não oferece suporte ao Aurora Auto Scaling para clusters de banco de dados secundários.
- Para usar o Database Activity Streams (DAS) no Aurora Global Database executando o Aurora MySQL 5.7, a versão do mecanismo deve ser 2.08 ou posterior. Para ter mais informações sobre DAS, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.
- No momento, as seguintes limitações se aplicam à atualização do Aurora Global Database:
  - Você não pode aplicar um grupo de parâmetros personalizado ao cluster de banco de dados global enquanto estiver executando uma atualização de versão principal desse banco de dados global do Aurora. Crie seus grupos de parâmetros personalizados em cada região do cluster global e, depois, aplique-os manualmente aos clusters regionais após a atualização.
  - Com um banco de dados Aurora global baseado no Aurora MySQL, você não poderá executar uma atualização no local do Aurora MySQL versão 2 para a versão 3 se o parâmetro `lower_case_table_names` estiver ativado. Para ter mais informações sobre os métodos que você pode usar, consulte Atualizações de versão principal.
  - Com o Aurora Global Database, não será possível realizar uma atualização de versão principal do mecanismo de banco de dados do Aurora PostgreSQL se o recurso de objetivo de ponto de recuperação (RPO) estiver ativado. Para ter mais informações sobre o recurso RPO, consulte Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–.
  - Com um Aurora Global Database, não é possível realizar uma atualização de versão secundária do Aurora MySQL 3.01 ou 3.02 para 3.03 ou posterior utilizando o processo padrão. Para obter detalhes sobre processo que deve ser utilizado, consulte Atualizar o Aurora MySQL modificando a versão do mecanismo.

  Para ter informações sobre como atualizar o Aurora Global Database, consulte Atualizar um Amazon Aurora Global Database.
- Não é possível interromper nem iniciar os clusters de banco de dados do Aurora em seu banco de dados global individualmente. Para saber mais, consulte Interromper e iniciar um cluster de banco de dados do Amazon Aurora.

- Instâncias de banco de dados do leitor do Aurora anexadas ao cluster de banco de dados do Aurora secundário podem ser reiniciadas em determinadas circunstâncias. Se a instância de banco de dados do gravador da Região da AWS primária for reiniciada ou sofrer failover, as instâncias de banco de dados do leitor em regiões secundárias também serão reiniciadas. O cluster secundário fica indisponível até que todas as instâncias de banco de dados do leitor estejam novamente sincronizadas com a instância do gravador do cluster de banco de dados primário. O comportamento do cluster primário durante a reinicialização ou o failover é igual ao de um cluster de banco de dados único e não global. Para obter mais informações, consulte Replicação com o Amazon Aurora.

  Certifique-se de entender os impactos no seu banco de dados global antes de fazer alterações no cluster de banco de dados primário. Para saber mais, consulte Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada.
- O Aurora Global Database ainda não oferece suporte ao status `inaccessible-encryption-credentials-recoverable` quando o Amazon Aurora perde o acesso à chave do AWS KMS do cluster de banco de dados. Nesses casos, o cluster de banco de dados criptografado entra diretamente no estado terminal `inaccessible-encryption-credentials`. Para mais informações sobre esses estados, consulte Visualizar o status do cluster do banco de dados.
- O Secrets Manager não é compatível com o Aurora Global Database. Ao adicionar uma região a um banco de dados global, você deve primeiro desativar a integração do Secrets Manager para a instância de banco de dados.
- Os clusters de banco de dados baseados em Aurora PostgreSQL que usam o Aurora Global Database têm as seguintes limitações:
  - Não há suporte ao gerenciamento de cache de cluster para clusters de banco de dados secundários do Aurora PostgreSQL que fazem parte de bancos de dados globais do Aurora.
  - Se o cluster primário do banco de dados global for baseado em uma réplica de uma instância do Amazon RDS PostgreSQL, não será possível criar um cluster secundário. Não tente criar um secundário a partir desse cluster usando a operação Console de gerenciamento da AWS, AWS CLI ou API da `CreateDBCluster`. As tentativas de fazer isso expiram e o cluster secundário não é criado.

Recomendamos criar clusters de banco de dados secundários para os bancos de dados globais usando a mesma versão do mecanismo de banco de dados do Aurora como primário. Para obter mais informações, consulte Criar um banco de dados global do Amazon Aurora.

Conceitos básicos do Amazon Aurora Global Database

Para começar a usar o Aurora Global Database, primeiro decida qual mecanismo de banco de dados do Aurora você deseja usar e em quais Regiões da AWS. Somente versões específicas dos mecanismos de banco de dados do Aurora MySQL e do Aurora PostgreSQL em determinadas Regiões da AWS são compatíveis com o Aurora Global Database. Para obter a lista completa, consulte Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.

É possível criar um Aurora Global Database de uma das seguintes maneiras:

- Crie um Aurora Global Database com novos clusters e instâncias de banco de dados do Aurora: é possível fazer isso seguindo as etapas em Criar um banco de dados global do Amazon Aurora. Depois de criar o cluster de banco de dados primário do Aurora, adicione a Região da AWS secundária seguindo as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.
- Use um cluster de banco de dados do Aurora existente que ofereça suporte ao recurso Aurora Global Database e adicione uma Região da AWS a ele: será possível fazer isso apenas se o cluster de banco de dados do Aurora existente usar uma versão do mecanismo de banco de dados compatível globalmente.

  Verifique se você pode escolher Add region (Adicionar região) para Action (Ação) no Console de gerenciamento da AWS quando seu cluster de bancos de dados Aurora for selecionado. Se você puder, você pode usar esse Aurora cluster de banco de dados para seu Aurora cluster global. Para obter mais informações, consulte Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Antes de criar um Aurora Global Database, recomendamos que você entenda todos os requisitos de configuração.

Tópicos

- Requisitos de configuração do banco de dados do Amazon Aurora global
- Criar um banco de dados global do Amazon Aurora
- Adicionar uma Região da AWS a um Amazon Aurora Global Database
- Criando um cluster de banco de dados Aurora dedicado em uma região secundária

- Criar um banco de dados global do Amazon Aurora com base em um snapshot do Aurora ou do Amazon RDS

Requisitos de configuração do banco de dados do Amazon Aurora global

Antes de começar a trabalhar com o banco de dados global, planeje os nomes do cluster e da instância, as regiões da AWS, o número de instâncias e as classes de instância que você pretende usar. Suas escolhas devem estar de acordo com os requisitos de configuração a seguir.

Um Aurora Global Database abrange pelo menos duas Regiões da AWS. A Região da AWS principal oferece suporte a um cluster de banco de dados Aurora com uma instância de banco de dados Aurora de gravador. Uma Região da AWS secundária executa um cluster de banco de dados Aurora apenas para leitura composto inteiramente de réplicas do Aurora. Pelo menos uma Região da AWS secundária é necessária, mas um Aurora Global Database pode ter até dez Regiões da AWS secundárias. A tabela lista o máximo de clusters de bancos de dados Aurora, instâncias de bancos de dados Aurora e réplicas do Aurora permitidos em um banco de dados do Aurora global.

Descrição	Região da AWS principal	Regiões da AWS secundárias
clusters de bancos de dados Aurora	1	10 (máximo)
Instâncias do gravador	1	0
Instâncias somente leitura (réplicas do Aurora), por cluster de bancos de dados Aurora	15 (máximo)	16 (total)
Instâncias somente leitura (máximo permitido, considerando o número real de regiões secundárias)	15 - s	s = número total de Regiões da AWS secundárias

Os clusters de bancos de dados Aurora que compõem um banco de dados do Aurora global têm os seguintes requisitos específicos:

- Requisitos de classe de instância de banco de dados – Um banco de dados do Aurora global requer classes de instância de banco de dados otimizadas para aplicativos com uso intensivo de memória Para ter mais informações sobre as classes de instância de banco de dados otimizadas, consulte Tipos de classe de instância de banco de dados. Recomendamos utilizar uma classe de instância db.r5 ou superior.
- Requisitos de Região da AWS: um Aurora Global Database precisa de um cluster de banco de dados Aurora principal em uma Região da AWS e pelo menos um cluster de banco de dados Aurora secundário em uma região diferente. Você pode criar até dez clusters de banco de dados secundários do Aurora (somente leitura) e cada um deve estar em uma região diferente. Em outras palavras, dois clusters de banco de dados Aurora em um Aurora Global Database não podem estar na mesma Região da AWS.

  Consulte informações sobre quais combinações do mecanismo de banco de dados do Aurora, versão do mecanismo e Região da AWS você pode usar com o banco de dados global do Aurora em Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.
- Requisitos de nomenclatura: os nomes escolhidos para cada um de seus clusters de banco de dados Aurora devem ser exclusivos em todas as Regiões da AWS. Você não pode usar o mesmo nome para clusters de bancos de dados Aurora diferentes, mesmo que eles estejam em regiões diferentes.
- Requisitos de capacidade para o Aurora Serverless v2: para um banco de dados global com o Aurora Serverless v2, a capacidade mínima recomendada para o cluster de banco de dados na Região da AWS primária é oito ACUs.

Antes de seguir os procedimentos nesta seção, você precisa de uma Conta da AWS. Conclua as tarefas de configuração para trabalhar com Amazon Aurora. Para obter mais informações, consulte Configuração de seu ambiente para Amazon Aurora. Também é necessário concluir outras etapas preliminares para criar qualquer cluster de bancos de dados Aurora. Para saber mais, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Quando estiver tudo pronto para configurar o banco de dados global, consulte Criar um banco de dados global do Amazon Aurora para conhecer o procedimento de criação de todos os recursos necessários. Também é possível seguir o procedimento em Adicionar uma Região da AWS a um Amazon Aurora Global Database para criar um banco de dados global usando um cluster existente do Aurora como cluster primário.

Criar um banco de dados global do Amazon Aurora

Para criar um banco de dados global do Aurora e os recursos associados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS, siga as etapas abaixo.

> **Note**
>
> Se você tiver um cluster de banco de dados do Aurora que esteja executando um mecanismo de banco de dados do Aurora compatível globalmente, poderá usar uma forma abreviada desse procedimento. Se for o caso, você poderá adicionar outra Região da AWS ao cluster de banco de dados existente para criar o banco de dados global do Aurora. Para fazer isto, consulte Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Console

As etapas para criar um Aurora Global Database começam fazendo login em uma Região da AWS compatível com o recurso Aurora Global Database. Para obter uma lista completa, consulte Regiões e mecanismos de banco de dados compatíveis com bancos de dados globais do Aurora.

Uma das etapas a seguir é escolher uma nuvem privada virtual (VPC) com base na Amazon VPC para o cluster de bancos de dados Aurora. Para usar sua própria VPC, recomendamos que você a crie com antecedência para que esteja disponível para escolher. Ao mesmo tempo, crie sub-redes relacionadas e, conforme necessário, um grupo de sub-rede e um grupo de segurança. Para saber como, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

Para obter informações gerais sobre como criar um cluster de bancos de dados Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Para criar um banco de dados do Aurora global

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Create database (Criar banco de dados). Na página Criar banco de dados (Criar banco de dados) , faça o seguinte:
   - Como método de criação do banco de dados, escolha Standard Create (Criação padrão). (Não escolha Easy Create (Criação fácil).

- Em `Engine type` na seção Opções de mecanismo, escolha o tipo de mecanismo aplicável, Aurora (compatível com MySQL) ou Aurora (compatível com PostgreSQL).

3. Continue criando seu banco de dados Aurora global usando as etapas dos procedimentos a seguir.

Criar um banco de dados global do Aurora MySQL

As etapas a seguir se aplicam a todas as versões do Aurora MySQL.

Como criar um banco de dados Aurora global usando o Aurora MySQL

Preencha a página Create database (Criar banco de dados).

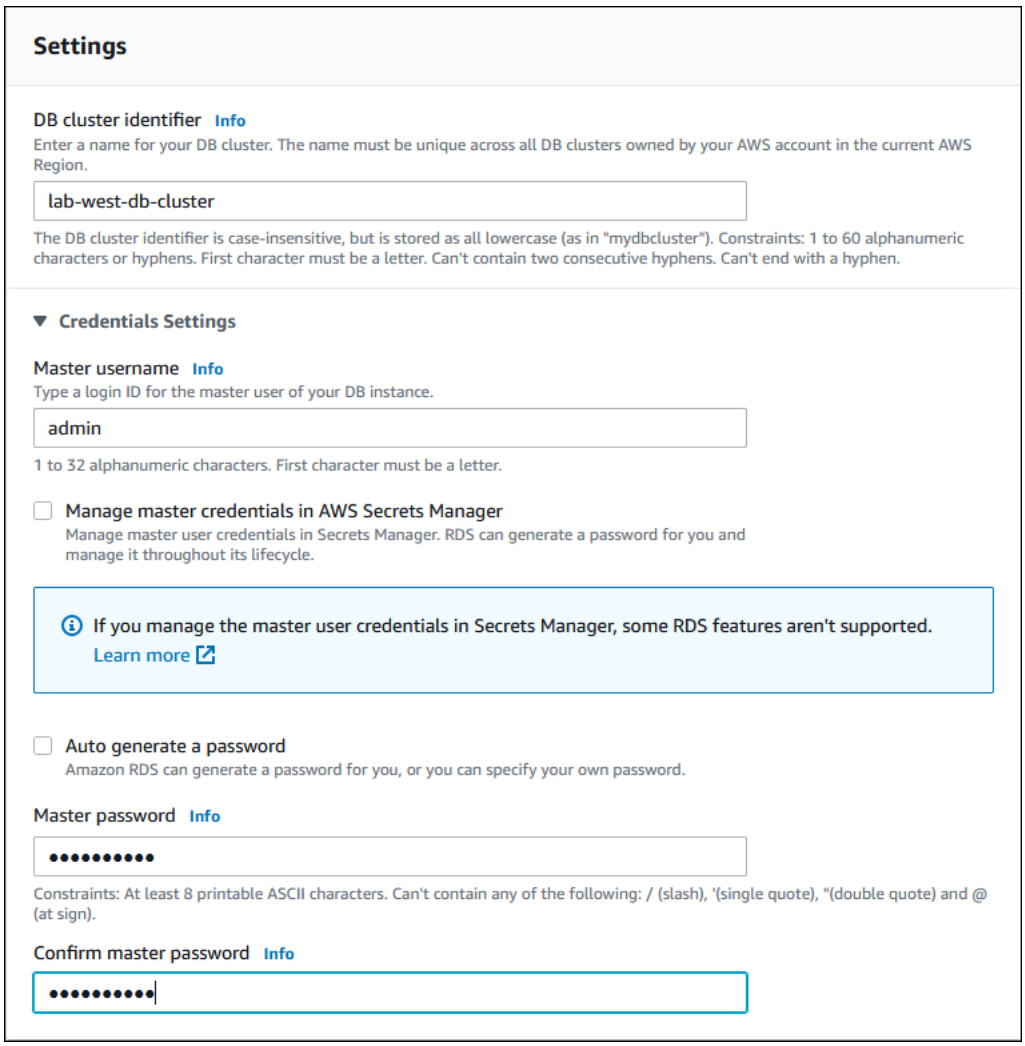
1. Para Engine options (Opções de mecanismo), escolha o seguinte:

   - Em Engine version (Versão do mecanismo), escolha a versão de Aurora MySQL que deseja usar para seu banco de dados do Aurora global.

2. Para Templates (Modelos), escolha Production (Produção). Ou, você pode escolher Dev/Teste se apropriado para o seu caso de uso. Não use Dev/Teste em ambientes de produção.
3. Em Settings (Configurações), faça o seguinte:

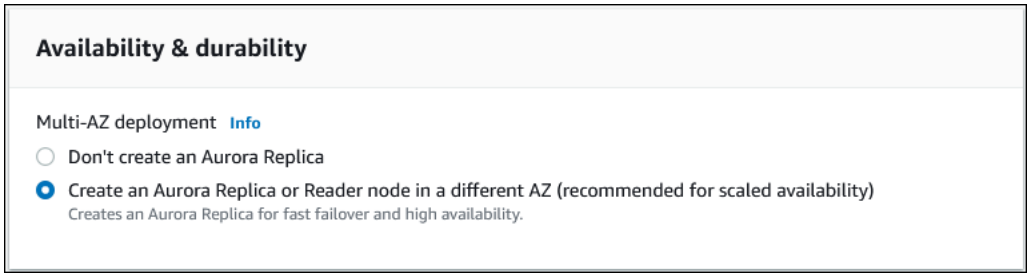
   a. Insira um nome significativo para o identificador do cluster de banco de dados. Quando você terminar de criar o banco de dados Aurora global, esse nome identifica o cluster de banco de dados primário.

   b. Insira sua própria senha para a conta de usuário de `admin` para a instância de banco de dados ou deixe Aurora gerar uma para você. Se você escolher Auto generate a password (Gerar uma senha automática), você terá uma opção para copiar a senha.



4. Em DB instance class (Classe da instância do banco de dados), escolha `db.r5.large` ou qualquer outra classe de instância de banco de dados otimizada para a memória. Recomendamos utilizar uma classe de instância db.r5 ou superior.

5. Para Availability & durability (Disponibilidade e durabilidade), recomendamos que você escolha Aurora criar uma Aurora réplica de um zona de disponibilidade (AZ) diferente para você. Se você não criar uma réplica Aurora agora, precisará mais tarde.



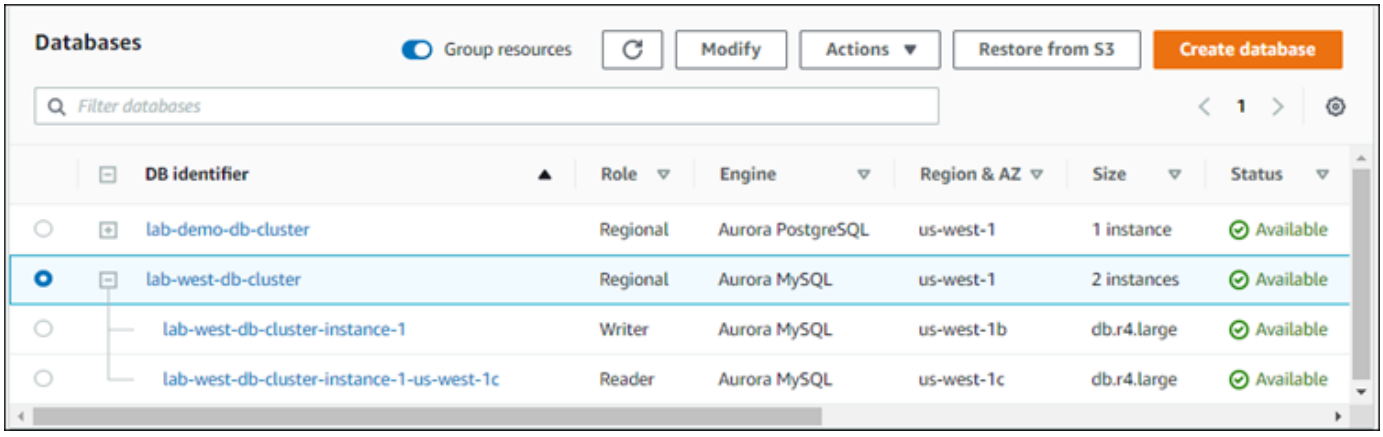
6. Para Connectivity (Conectividade), escolha a virtual private cloud (VPC) com base na Amazon VPC qual define o ambiente de rede virtual para essa instância de banco de dados. Você pode escolher os padrões para simplificar essa tarefa.
7. Conclua as configurações Database authentication (Autenticação de banco de dados). Para simplificar o processo, você pode escolher a Password authentication (Autenticação de senha) agora e configurar AWS Identity and Access Management (IAM) mais tarde.
8. Em Additional configuration (Configuração adicional), faça o seguinte:

   a. Insira um nome para Initial database name (Banco de dados inicial) para criar a instância de banco de dados primário do Aurora para esse cluster. Este é o nó do gravador para o cluster de banco de dados primário do Aurora.

      Deixe os padrões selecionados para o grupo de parâmetros do cluster de banco de dados e o grupo de parâmetros de banco de dados, a menos que você tenha seus próprios grupos de parâmetros personalizados que deseja usar.

   b. Limpe a caixa de seleção Enable backtrack (Habilitar retrocesso), se estiver selecionada. Os bancos de dados globais do Aurora não são compatíveis com retrocesso. Caso contrário, você pode aceitar todas as outras configurações padrão para Additional configuration (Configuração adicional).
9. Escolha Create database (Criar banco de dados).

   Pode levar alguns minutos para Aurora concluir o processo de criação da instância de bancos de dados Aurora, sua réplica do Aurora e o cluster de bancos de dados Aurora. Você pode dizer quando o cluster de banco de dados Aurora está pronto para usar como o cluster de banco de dados primário em um banco de dados global Aurora por seu status. Quando isso acontece, o status e o do nó do gravador e da réplica estão Disponíveis, conforme mostrado a seguir.



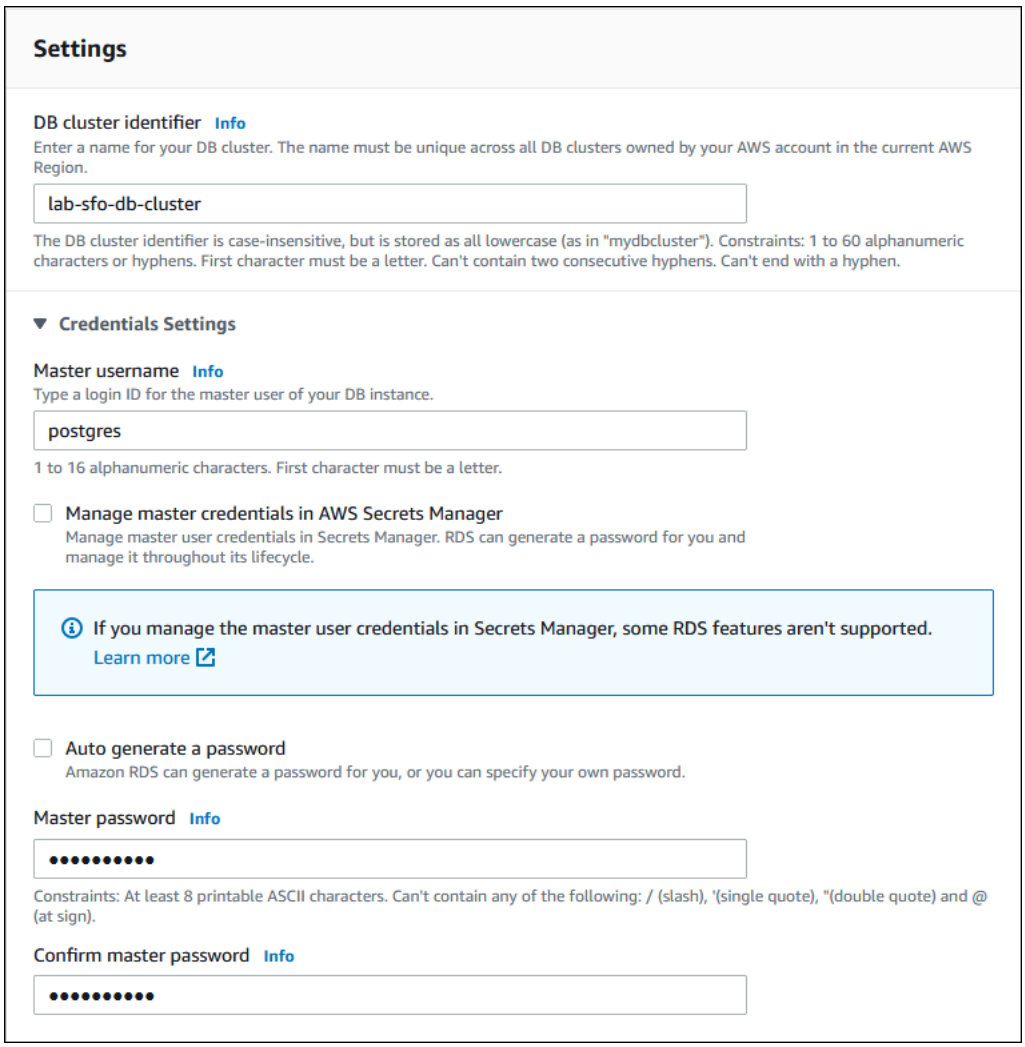
Quando seu cluster de banco de dados primário estiver disponível, crie o banco de dados do global Aurora adicionando um cluster secundário. Para isso, siga as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Criar um banco de dados global usando Aurora PostgreSQL

Como criar um banco de dados Aurora global usando o Aurora PostgreSQL

Preencha a página Create database (Criar banco de dados).

1. Para Engine options (Opções de mecanismo), escolha o seguinte:
   - Em Engine version (Versão do mecanismo), escolha a versão do Aurora PostgreSQL que você deseja usar para seu banco de dados Aurora global.
2. Para Templates (Modelos), escolha Production (Produção). Ou você pode escolher Dev/Test, se apropriado. Não use Dev/Teste em ambientes de produção.
3. Em Settings (Configurações), faça o seguinte:
   a. Insira um nome significativo para o identificador do cluster de banco de dados. Quando você terminar de criar o banco de dados Aurora global, esse nome identifica o cluster de banco de dados primário.
   b. Insira sua própria senha para a conta de administrador padrão para o cluster de banco de dados ou deixe Aurora gerar uma para você. Se você escolher Auto generate a password (Gerar uma senha automática), você terá uma opção para copiar a senha.



4. Em DB instance class (Classe da instância do banco de dados), escolha `db.r5.large` ou qualquer outra classe de instância de banco de dados otimizada para a memória. Recomendamos utilizar uma classe de instância db.r5 ou superior.
5. Para disponibilidade e durabilidade, recomendamos que você escolha Aurora criar uma réplica Aurora de um AZ diferente para você. Se você não criar uma réplica Aurora agora, precisará mais tarde.
6. Para Connectivity (Conectividade), escolha a virtual private cloud (VPC) com base na Amazon VPC qual define o ambiente de rede virtual para essa instância de banco de dados. Você pode escolher os padrões para simplificar essa tarefa.
7. (Opcional) Conclua as configurações Database authentication (Autenticação de banco de dados). A autenticação por senha está sempre habilitada. Para simplificar o processo, você pode pular esta seção e configurar IAM ou senha e autenticação Kerberos posteriormente.
8. Em Additional configuration (Configuração adicional), faça o seguinte:

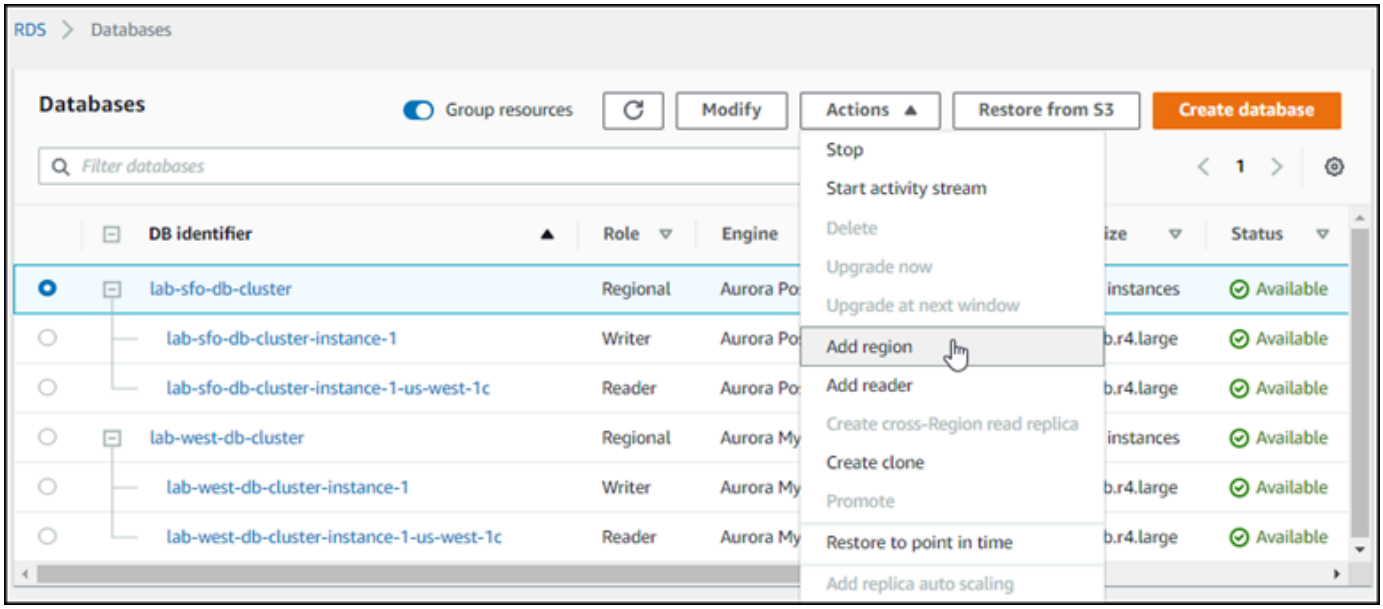
a. Insira um nome para Initial database name (Banco de dados inicial) para criar a instância de banco de dados primário do Aurora para esse cluster. Este é o nó do gravador para o cluster de banco de dados primário do Aurora.

   Deixe os padrões selecionados para o grupo de parâmetros do cluster de banco de dados e o grupo de parâmetros de banco de dados, a menos que você tenha seus próprios grupos de parâmetros personalizados que deseja usar.

b. Aceite todas as outras configurações padrão para Additional configuration (Configuração adicional), como Criptografia, Exportações de log e outros.

9. Escolha Create database (Criar banco de dados).

   Pode levar alguns minutos para Aurora concluir o processo de criação da instância de bancos de dados Aurora, sua réplica do Aurora e o cluster de bancos de dados Aurora. Quando o cluster estiver pronto para uso, o cluster de bancos de dados Aurora e seus nós de gravador e réplica exibirão o status Available (Disponível). Isso se torna o cluster de banco de dados primário do seu banco de dados Aurora global, depois de adicionar um secundário.



Quando seu cluster de banco de dados primário estiver disponível, crie um ou mais clusters secundários seguindo as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

AWS CLI

Os comandos AWS CLI nos procedimentos a seguir realizam as seguintes tarefas:

1. Crie um banco de dados global Aurora, dando nome a ele e especificando o tipo de mecanismo de banco de dados de Aurora que você planeja usar.
2. Crie um cluster de bancos de dados Aurora para o banco de dados do Aurora global.
3. Crie a instância de bancos de dados Aurora para o cluster. Este é o cluster de banco de dados primário do Aurora para o banco de dados global.
4. Crie uma segunda instância de banco de dados para cluster de bancos de dados Aurora. Este é um leitor para concluir o cluster de bancos de dados Aurora.
5. Crie um segundo cluster de bancos de dados Aurora em outra região e adicione-o ao banco de dados Aurora global, seguindo as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Siga o procedimento para o mecanismo de banco de dados do Aurora.

Criar um banco de dados global do Aurora MySQL

Para criar um banco de dados do Aurora global usando o Aurora MySQL

1. Use o comando `create-global-cluster` da CLI, transferindo o nome da Região da AWS, o mecanismo de banco de dados Aurora e a versão.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-mysql \
       --engine-version version # optional
   ```

   Para Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-mysql ^
       --engine-version version # optional
   ```

   Isso cria um banco de dados Aurora global “vazio”, com apenas um nome (identificador) e mecanismo de banco de dados do Aurora. Pode levar alguns minutos para que o banco de dados Aurora global esteja disponível. Antes de ir à próxima etapa, use o comando de CLI `describe-global-clusters` para ver se está disponível.

```
aws rds describe-global-clusters --region primary_region --global-cluster-
identifier global_database_id
```

Quando o banco de dados do Aurora global estiver disponível, você pode criar seu cluster de banco de dados primário do Aurora.

2. Para criar um cluster de banco de dados primário do Aurora, use o comando de CLI do `create-db-cluster`. Inclua o nome do Aurora Global Database usando o parâmetro `--global-cluster-identifier`.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-mysql \
     --engine-version version \
     --global-cluster-identifier global_database_id
   ```

   Para Windows:

   ```
   aws rds create-db-cluster ^
     --region primary_region ^
     --db-cluster-identifier primary_db_cluster_id ^
     --master-username userid ^
     --master-user-password password ^
     --engine aurora-mysql ^
     --engine-version version ^
     --global-cluster-identifier global_database_id
   ```

   Use o comando `describe-db-clusters` da AWS CLI para confirmar se o cluster de bancos de dados Aurora está pronto. Para destacar um cluster de bancos de dados Aurora específico, use o parâmetro `--db-cluster-identifier`. Ou você pode deixar de fora o nome do cluster de bancos de dados Aurora no comando para obter detalhes sobre todos os seus clusters de bancos de dados Aurora na região determinada.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

Quando a resposta é exibida `"Status": "available"` para o cluster, estará pronta para uso.

3. Crie a instância de banco de dados para o cluster de banco de dados primário do Aurora. Para isso, use o comando de CLI de `create-db-instance`. Dê o comando o nome do cluster de bancos de dados Aurora e especifique os detalhes de configuração para a instância. Você não precisa transferir os parâmetros `--master-username` e `--master-user-password` no comando, porque obtém aqueles do cluster de bancos de dados Aurora.

   Para o `--db-instance-class`, você pode usar apenas aqueles de classes otimizadas para a memória, como `db.r5.large`. Recomendamos utilizar uma classe de instância db.r5 ou superior. Para ter mais informações sobre essas classes, consulte Tipos de classe de instância de banco de dados.

   Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-mysql \
  --engine-version version \
  --region primary_region
```

   Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-mysql ^
  --engine-version version ^
  --region primary_region
```

   A operação `create-db-instance` pode demorar um pouco para ser concluída. Verifique o status para ver se a instância de bancos de dados Aurora está disponível antes de continuar.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Quando o comando exibe o status `available`, é possível criar outra instância de bancos de dados do Aurora para o cluster de banco de dados primário. Esta é a instância do leitor (a réplica Aurora) para o cluster de bancos de dados Aurora.

4. Para criar outra instância de bancos de dados Aurora para o cluster, use o comando de CLI do `create-db-instance`.

   Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-mysql
```

   Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-mysql
```

Quando a instância de banco de dados está disponível, a replicação começa do nó do gravador para a Réplica. Antes de continuar, verifique se a instância de banco de dados está disponível com o comando de CLI do `describe-db-instances`.

Neste ponto, você tem um banco de dados do Aurora global com seu cluster de banco de dados primário do Aurora contendo uma instância de banco de dados de gravador e uma réplica de Aurora. Agora você pode adicionar um cluster de bancos de dados Aurora somente leitura em uma região diferente para concluir seu banco de dados do Aurora global. Para isso, siga as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Criar um banco de dados global usando Aurora PostgreSQL

Quando você cria objetos Aurora para um banco de dados global Aurora usando os comandos a seguir, pode levar alguns minutos para que cada um fique disponível. Recomendamos que, após

concluir qualquer comando, verifique o status do objeto Aurora específico para garantir que o status esteja disponível.

Para isso, use o comando de CLI de `describe-global-clusters`.

```
aws rds describe-global-clusters --region primary_region
    --global-cluster-identifier global_database_id
```

Para criar um banco de dados do Aurora global usando o Aurora PostgreSQL

1. Use o comando de CLI de `create-global-cluster`.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-postgresql \
       --engine-version version # optional
   ```

   Para Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-postgresql ^
       --engine-version version # optional
   ```

   Quando o Aurora Global Database estiver disponível, você pode criar seu cluster de banco de dados primário do Aurora.

2. Para criar um cluster de banco de dados primário do Aurora, use o comando de CLI do `create-db-cluster`. Inclua o nome do Aurora Global Database usando o parâmetro `--global-cluster-identifier`.

   Para Linux, macOS ou Unix:

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-postgresql \
   ```

```
  --engine-version version \
  --global-cluster-identifier global_database_id
```

Para Windows:

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

Verifique se o cluster de bancos de dados Aurora está pronto. Quando a resposta do comando a seguir mostra `"Status": "available"` para o cluster de bancos de dados Aurora, você pode continuar.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

3. Crie a instância de banco de dados para o cluster de banco de dados primário do Aurora. Para isso, use o comando de CLI de `create-db-instance`.

   Transfira o nome do seu cluster de bancos de dados Aurora para o parâmetro `--db-cluster-identifier`.

   Você não precisa transferir os parâmetros `--master-username` e `--master-user-password` no comando, porque obtém aqueles do cluster de bancos de dados Aurora.

   Para o `--db-instance-class`, você pode usar apenas aqueles de classes otimizadas para a memória, como `db.r5.large`. Recomendamos utilizar uma classe de instância db.r5 ou superior. Para ter mais informações sobre essas classes, consulte Tipos de classe de instância de banco de dados.

   Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
```

```
  --db-instance-identifier db_instance_id \
  --engine aurora-postgresql \
  --engine-version version \
  --region primary_region
```

Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --region primary_region
```

4. Verifique o status da instância de bancos de dados Aurora antes de continuar.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Se a resposta mostrar que o status da instância de bancos de dados do Aurora é `available`, será possível criar outra instância de bancos de dados do Aurora para o cluster de banco de dados primário.

5. Para criar uma réplica de Aurora para cluster de bancos de dados Aurora, use o comando de CLI do`create-db-instance`.

Para Linux, macOS ou Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-postgresql
```

Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-postgresql
```

Quando a instância de banco de dados está disponível, a replicação começa do nó do gravador para a Réplica. Antes de continuar, verifique se a instância de banco de dados está disponível com o comando de CLI do `describe-db-instances`.

Seu banco de dados Aurora global existe, mas tem apenas sua região primária com um cluster de bancos de dados Aurora composto por uma instância de banco de dados de gravador e uma réplica do Aurora. Agora você pode adicionar um cluster de bancos de dados Aurora somente leitura em uma região diferente para concluir seu banco de dados do Aurora global. Para isso, siga as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

API do RDS

Para criar um banco de dados global Aurora com a API do RDS, execute a operação CreateGlobalCluster.

Adicionar uma Região da AWS a um Amazon Aurora Global Database

É possível usar o procedimento a seguir para adicionar um cluster secundário adicional a um banco de dados global existente. Também é possível criar um banco de dados global por meio de um cluster de banco de dados independente do Aurora usando esse procedimento para adicionar a primeira região da AWS secundária.

Um Aurora Global Database precisa de pelo menos um cluster de banco de dados Aurora secundário em uma Região da AWS diferente do cluster de banco de dados Aurora principal. Você pode anexar até dez clusters de banco de dados secundários ao Aurora Global Database. Repita o procedimento a seguir para cada novo cluster de banco de dados secundário. Para cada cluster de banco de dados secundário que você adiciona ao banco de dados Aurora global, reduza o número de réplicas Aurora permitidas ao cluster de banco de dados primário para um.

Por exemplo, se seu Aurora Global Database tiver dez regiões secundárias, seu cluster de banco de dados primário poderá ter apenas dez (em vez de quinze) réplicas. Para obter mais informações, consulte Requisitos de configuração do banco de dados do Amazon Aurora global.

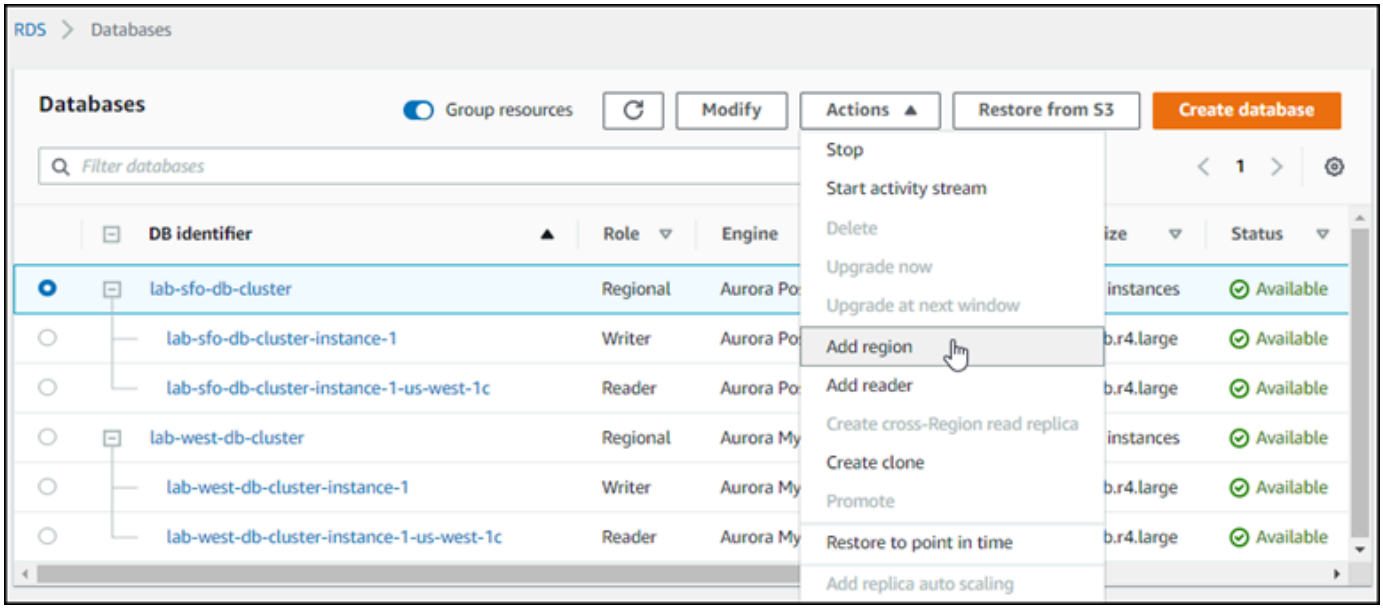
O número de réplicas do Aurora (instâncias de leitor) no cluster de banco de dados primário determina o número de clusters de banco de dados secundários que você pode adicionar. Em alguns casos, o número de instâncias de leitor no cluster de banco de dados primário mais o número de clusters secundários pode totalizar 15. Por exemplo, se você tiver 14 instâncias de leitor no cluster de banco de dados primário e 1 cluster secundário, não será possível adicionar um cluster secundário a um banco de dados global.

> **Note**
>
> No Aurora MySQL versão 3, ao criar um cluster secundário, certifique-se de que o valor de `lower_case_table_names` corresponda ao valor no cluster principal. Essa configuração é um parâmetro de banco de dados que afeta a forma como o servidor lida com diferenciação entre minúsculas e maiúsculas. Para obter mais informações sobre parâmetros de banco de dados, consulte Grupos de parâmetros para Amazon Aurora.
> Recomendamos que, ao criar um cluster secundário, use a mesma versão do mecanismo de banco de dados para o primário e o secundário. Se necessário, atualize o cluster primário para que fique com a mesma versão do secundário. Para obter mais informações, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões.

Console

Para adicionar uma Região da AWS a um Aurora Global Database

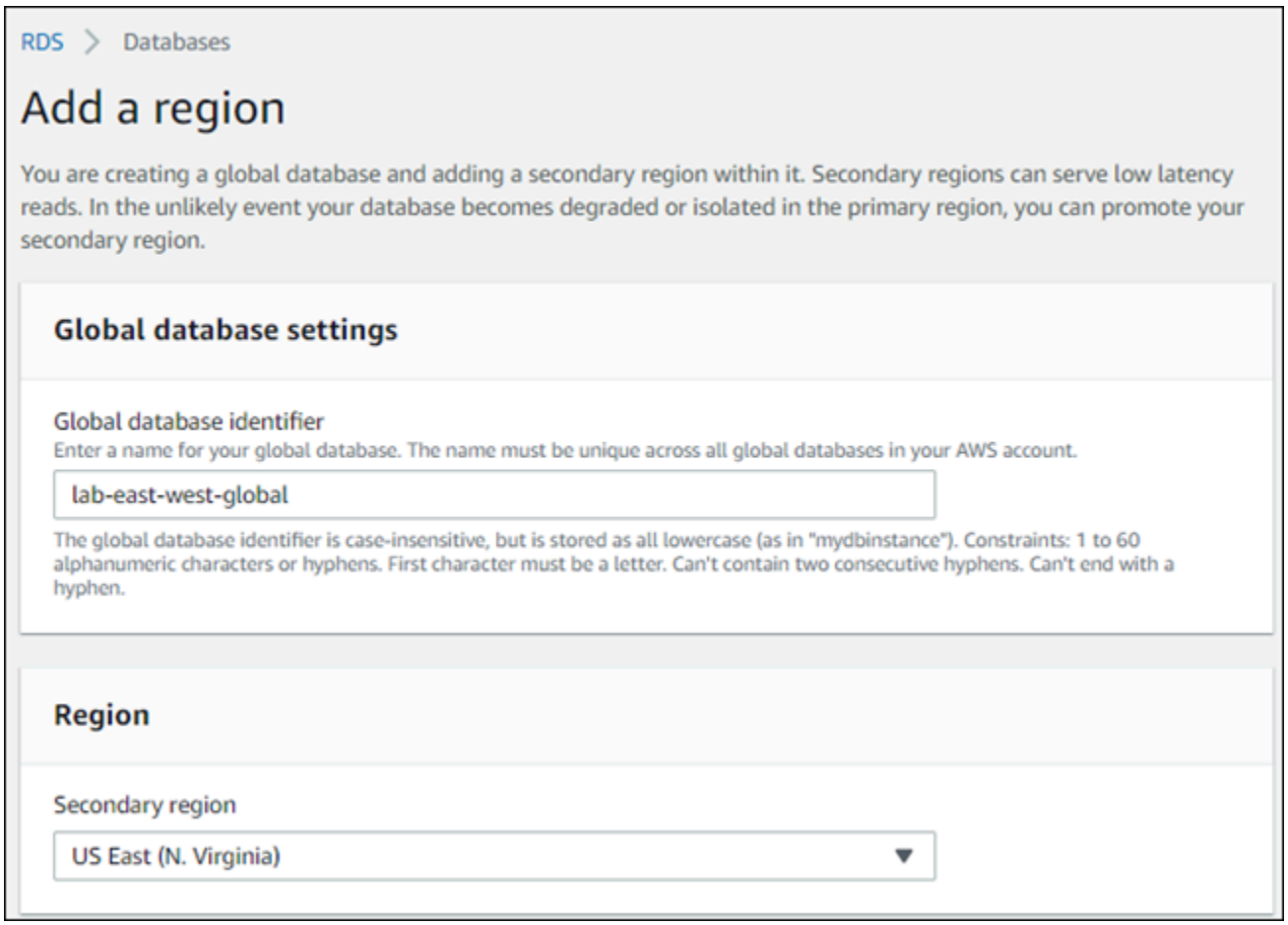
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação do Console de gerenciamento da AWS, escolha Databases (Bancos de dados).
3. Escolha o banco de dados Aurora global que precisa de um cluster de bancos de dados Aurora secundário. Certifique-se de que o cluster de bancos de dados Aurora primário seja `Available`.
4. Em Ações, selecione Adicionar região da AWS.



5. Na página Add a region (Adicionar uma região), escolha a Região da AWS secundária.

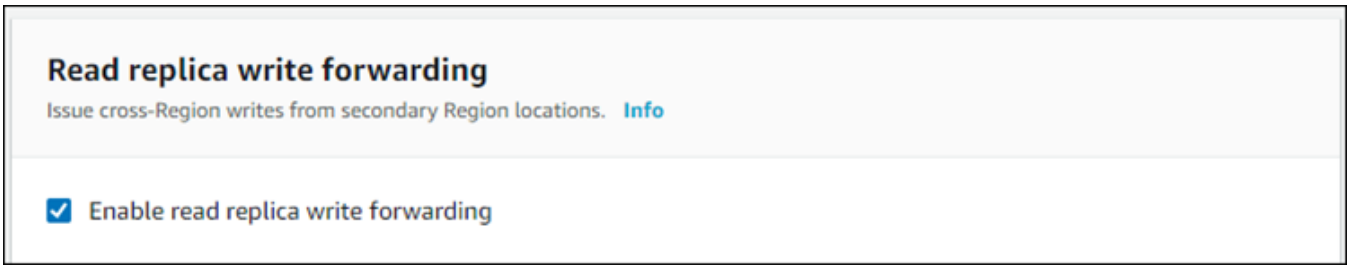
   Não é possível escolher uma Região da AWS que já tenha um cluster de banco de dados Aurora secundário para o mesmo Aurora Global Database. Além disso, não pode ser a mesma região que o cluster de banco de dados primário do Aurora.

> **Note**
>
> Os bancos de dados globais do Babelfish para Aurora PostgreSQL funcionarão em regiões secundárias somente se os parâmetros que controlam as preferências do Babelfish estiverem ativados nessas regiões. Para obter mais informações, consulte Configurações de grupo de parâmetros de cluster de banco de dados para o Babelfish.



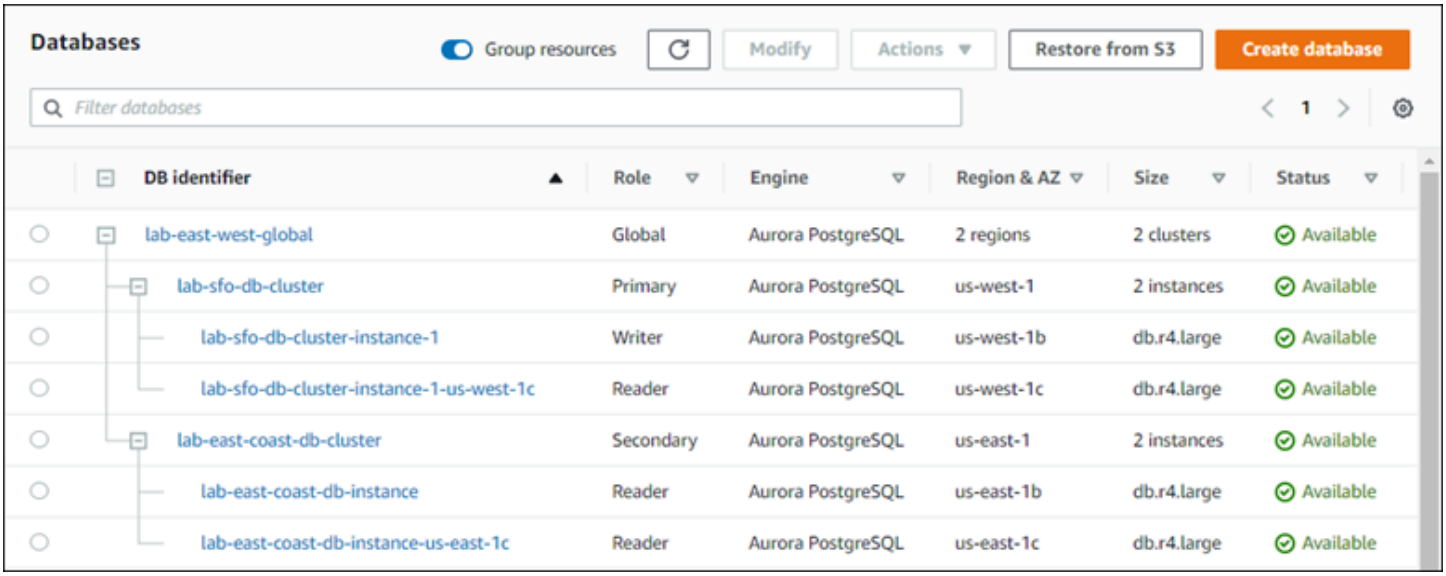
6. Preencha os campos restantes do cluster secundário do Aurora na nova região da AWS. Estas são as mesmas opções de configuração que para qualquer instância de cluster de banco de dados Aurora, exceto para a seguinte opção apenas para bancos de dados globais Aurora baseados no Aurora MySQL:

   - Ativar encaminhamento de gravação de réplica de leitura – Essa configuração opcional permite que os clusters de banco de dados secundários do banco de dados global Aurora encaminhe as operações de gravação para o cluster primário. Para obter mais informações, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.



7. Escolha Adicionar região da AWS.

Ao terminar de adicionar a região ao seu banco de dados global do Aurora, você poderá visualizá-la na lista Databases (Bancos de dados) no Console de gerenciamento da AWS, como mostrado na captura de tela.



AWS CLI

Para adicionar uma Região da AWS secundária a um Aurora Global Database

Para adicionar um cluster secundário ao banco de dados global usando a CLI, você já deve ter o objeto de contêiner do cluster global. Se você ainda não executou o comando `create-global-cluster`, consulte o procedimento da CLI em Criar um banco de dados global do Amazon Aurora.

1. Use o comando de CLI de `create-db-cluster` com o nome (`--global-cluster-identifier`) do seu banco de dados do Aurora global. Para outros parâmetros:
2. Para `--region`, escolha uma Região da AWS diferente de sua região principal do Aurora.
3. Escolha valores específicos para os parâmetros `--engine` e `--engine-version`. Esses valores são os mesmos que os do cluster de banco de dados primário Aurora em seu banco de dados globalAurora.
4. Para um cluster criptografado, especifique sua Região da AWS principal como a `--source-region` da criptografia.

O exemplo a seguir cria um novo cluster de banco de dados de Aurora e o anexa a um banco de dados do Aurora global como um cluster de banco de dados Aurora secundário somente leitura. Na última etapa, uma instância de bancos de dados Aurora é adicionada ao novo cluster de bancos de dados Aurora.

Para Linux, macOS ou Unix:

```
aws rds --region secondary_region \
  create-db-cluster \
    --db-cluster-identifier secondary_cluster_id \
    --global-cluster-identifier global_database_id \
    --engine aurora-mysql | aurora-postgresql \
    --engine-version version

aws rds --region secondary_region \
  create-db-instance \
    --db-instance-class instance_class \
    --db-cluster-identifier secondary_cluster_id \
    --db-instance-identifier db_instance_id \
    --engine aurora-mysql | aurora-postgresql
```

Para Windows:

```
aws rds --region secondary_region ^
  create-db-cluster ^
    --db-cluster-identifier secondary_cluster_id ^
    --global-cluster-identifier global_database_id_id ^
    --engine aurora-mysql | aurora-postgresql ^
    --engine-version version

aws rds --region secondary_region ^
  create-db-instance ^
    --db-instance-class instance_class ^
    --db-cluster-identifier secondary_cluster_id ^
    --db-instance-identifier db_instance_id ^
    --engine aurora-mysql | aurora-postgresql
```

API do RDS

Para adicionar uma nova Região da AWS a um Aurora Global Database com a API do RDS, execute a operação CreateDBCluster. Especifique o identificador do banco de dados global existente usando o parâmetro `GlobalClusterIdentifier`.

Criando um cluster de banco de dados Aurora dedicado em uma região secundária

Embora um Aurora Global Database exija pelo menos um cluster de banco de dados Aurora secundário em uma Região da AWS diferente da principal, você pode usar uma configuração dedicada para o cluster secundário. Um cluster de banco de dados secundário dedicado do Aurora é aquele que não tem uma instância de banco de dados. Esse tipo de configuração pode reduzir as despesas para um banco de dados globalAurora. Em um cluster de bancos de dados Aurora, a computação e o armazenamento são desacoplados. Sem a instância de banco de dados, você não será cobrado pela computação, apenas pelo armazenamento. Se estiver configurado corretamente, o volume de armazenamento de um secundário dedicado será mantido em sincronia com o cluster de banco de dados primário.Aurora.

Você adiciona o cluster secundário como normalmente faz ao criar um banco de dados globalAurora. Se você estiver criando todos os clusters no banco de dados global, siga o procedimento em Criar um banco de dados global do Amazon Aurora. Se você já tiver um cluster de banco de dados a ser usado como cluster primário, siga o procedimento em Adicionar uma Região da AWS a um Amazon Aurora Global Database.

Depois que o cluster de banco de dados primário do Aurora iniciar a replicação para o secundário, você deve excluir a instância de banco de dados somente leitura do Aurora do cluster de banco de dados secundário do Aurora. Esse cluster secundário agora é considerado “dedicado” porque não tem mais uma instância de banco de dados. Mesmo sem nenhuma instância de banco de dados no cluster primário, o Aurora mantém o volume de armazenamento em sincronia com o cluster de banco de dados primário do Aurora.

> **Warning**
>
> Com o Aurora PostgreSQL, para criar um cluster dedicado em uma Região da AWS secundária, use a AWS CLI ou a API do RDS para adicionar a Região da AWS secundária. Ignore a etapa para criar a instância de banco de dados do leitor para o cluster secundário. Atualmente, a criação de um cluster dedicado não é compatível com o console do RDS. Para obter os procedimentos da CLI e da API a serem usados, consulte Adicionar uma Região da AWS a um Amazon Aurora Global Database.
>
> Se o banco de dados global estiver usando uma versão de mecanismo do Aurora PostgreSQL inferior a 13.4, 12.8 ou 11.13, criar uma instância de banco de dados de leitor em uma região secundária e, posteriormente, excluí-la pode causar um problema de vácuo do Aurora PostgreSQL na instância de banco de dados de gravador da região primária. Se

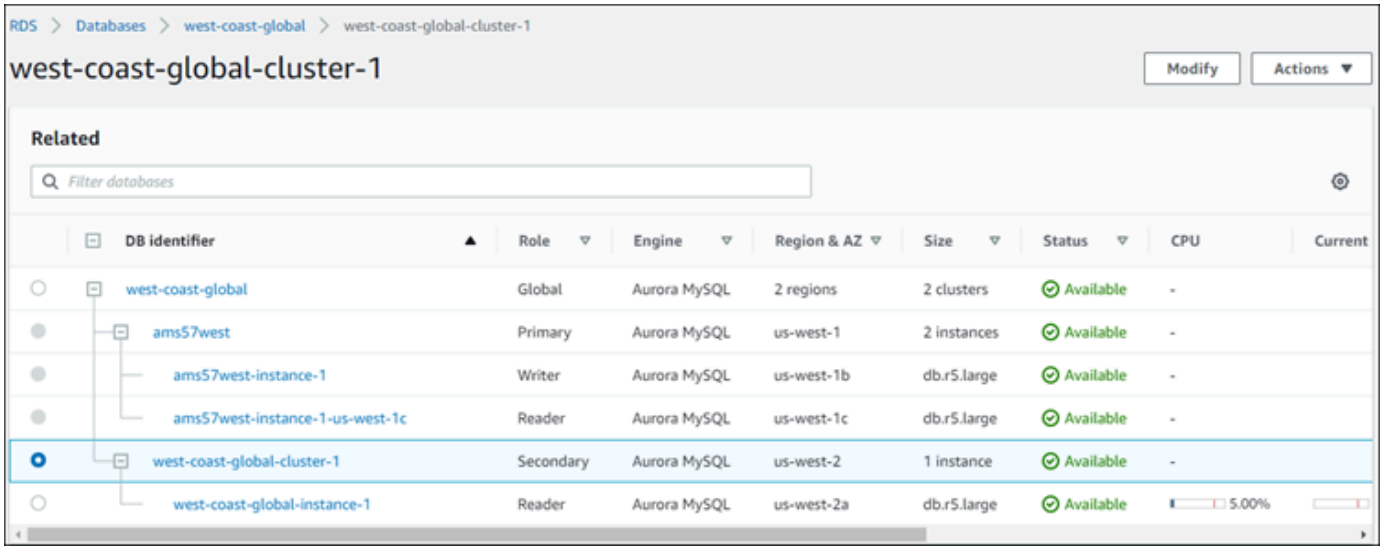
você encontrar esse problema, reinicie a instância de banco de dados do gravador da região primária depois de excluir a instância de banco de dados do leitor da região secundária.

Para adicionar um cluster de banco de dados Aurora secundário dedicado ao seu banco de dados global Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação do Console de gerenciamento da AWS, escolha Databases (Bancos de dados).
3. Escolha o banco de dados Aurora global que precisa de um cluster de bancos de dados Aurora secundário. Certifique-se de que o cluster de bancos de dados Aurora primário seja `Available`.
4. Em Ações, selecione Adicionar região da AWS.
5. Na página Add a region (Adicionar uma região), escolha a Região da AWS secundária.

   Não é possível escolher uma Região da AWS que já tenha um cluster de banco de dados Aurora secundário para o mesmo Aurora Global Database. Além disso, não pode ser a mesma região que o cluster de banco de dados primário do Aurora.
6. Preencha os campos restantes do cluster secundário do Aurora na nova Região da AWS. Essas são as mesmas opções de configuração que para qualquer instância de cluster de banco de dados deAurora.

   Para um banco de dados global Aurora baseado no Aurora MySQL, desconsidere a opção Enable read replica write forwarding (Ativar encaminhamento de gravação de réplica de leitura). Esta opção não terá nenhuma função depois que você excluir a instância do leitor.
7. Escolha Adicionar região da AWS. Ao terminar de adicionar a região ao seu banco de dados global do Aurora, você poderá visualizá-la na lista Databases (Bancos de dados) no Console de gerenciamento da AWS, como mostrado na captura de tela.

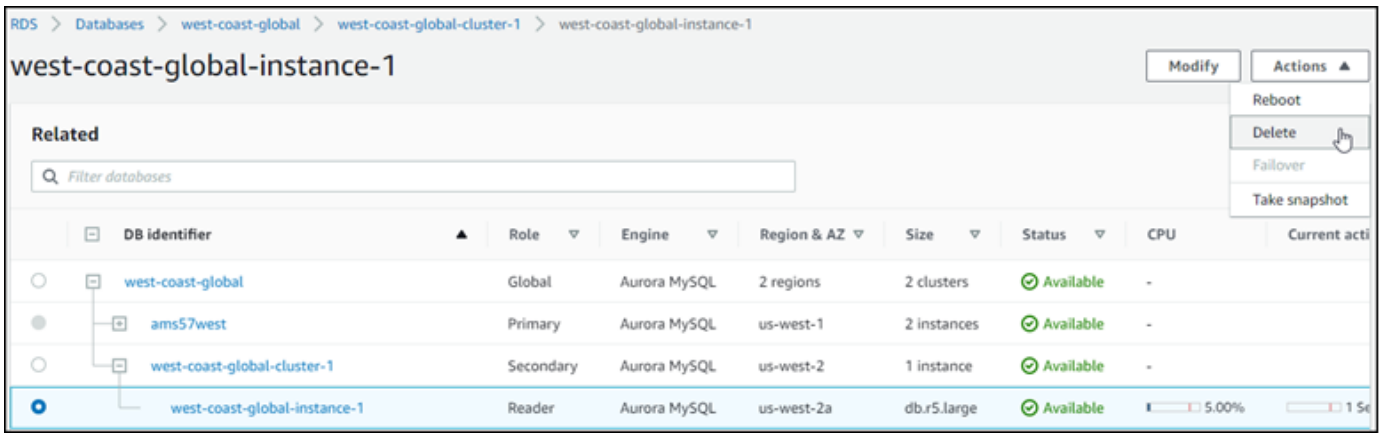


8. Verifique o status do cluster de banco de dados secundário do Aurora e sua instância do leitor antes de continuar, usando o Console de gerenciamento da AWS ou a AWS CLI. Por exemplo:

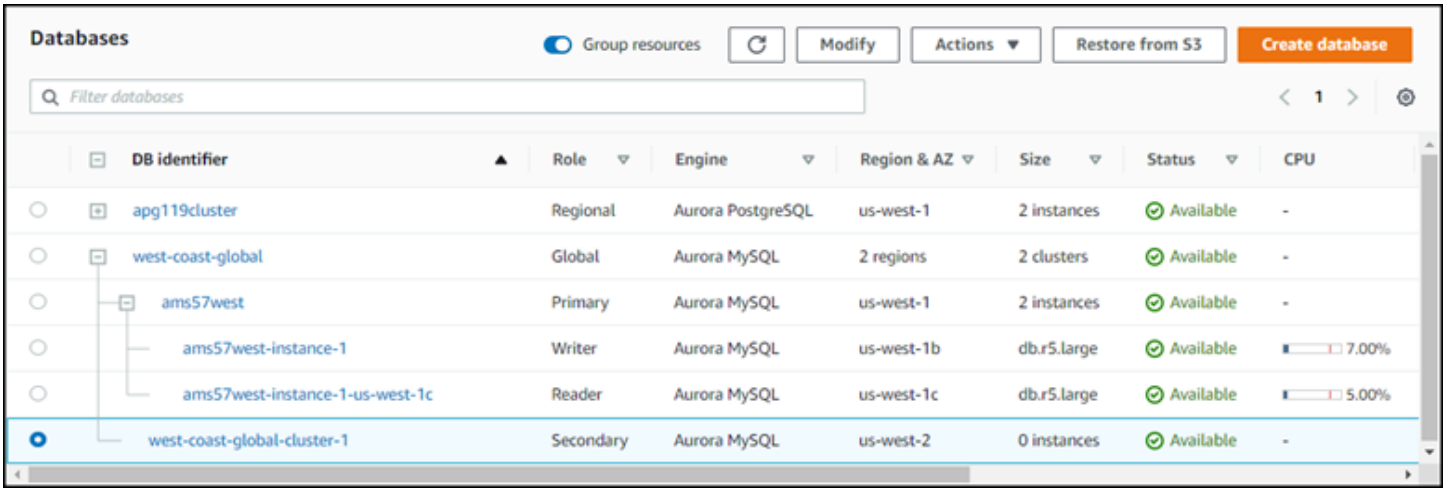
```
$ aws rds describe-db-clusters --db-cluster-identifier secondary-cluster-id --query
 '*[].[Status]' --output text
```

   Pode levar vários minutos para que o status de um cluster de banco de dados secundário do Aurora recém-adicionado mude de `creating` para `available`. Quando o cluster de bancos de dados Aurora estiver disponível, você pode excluir a instância do leitor.

9. Selecione a instância do leitor no cluster de banco de dados secundário do Aurora e escolha Delete (Excluir).



Depois de excluir a instância do leitor, o cluster secundário permanece parte do banco de dados global Aurora. Não tem nenhuma instância associada a ele, como mostrado a seguir.



Você pode usar esse cluster de banco de dados Aurora secundário dedicado para recuperar manualmente seu Amazon Aurora Global Database de uma interrupção não planejada na Região da AWS principal, se tal interrupção ocorrer.

Criar um banco de dados global do Amazon Aurora com base em um snapshot do Aurora ou do Amazon RDS

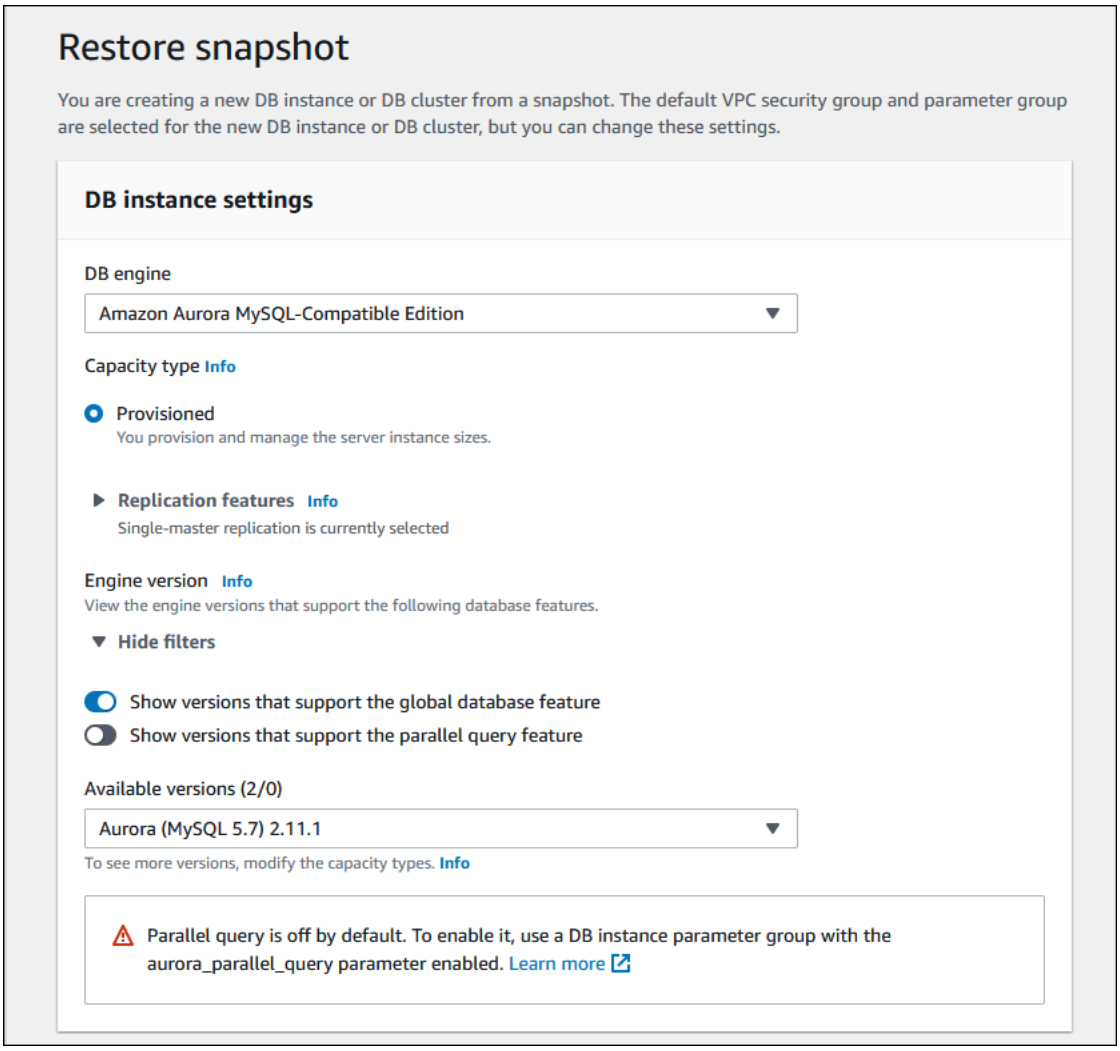
Você pode restaurar um snapshot de um cluster de bancos de dados Aurora ou de uma instância de banco de dados do Amazon RDS para usar como ponto de partida para seu banco de dados Aurora global. Você recupera o snapshot e cria um novo cluster de bancos de dados Aurora provisionado ao mesmo tempo. Em seguida, adicione outra Região da AWS ao cluster de banco de dados restaurado, transformando-o em um Aurora Global Database. Qualquer Aurora cluster de banco de dados criado usando um snapshot dessa maneira se torna o Aurora cluster primário do seu banco de dados global .

O snapshot que você usa pode ser de um `provisioned` ou de um cluster de banco de dados do `serverless` Aurora.

Durante o processo de restauração, escolha o mesmo tipo de mecanismo de banco de dados que o snapshot. Por exemplo, suponha que você deseje restaurar um snapshot que foi feito a partir de um cluster de banco de dados do Aurora Serverless que está executando o Aurora PostgreSQL. Nesse caso, você cria um cluster de banco de dados Aurora PostgreSQL usando esse mesmo mecanismo de Aurora banco de dados e versão.

O cluster de banco de dados restaurado assume a função de cluster principal para banco de dados global Aurora quando você adiciona uma Região da AWS a ele. Todos os dados contidos neste

cluster primário são replicados para todos os clusters secundários adicionados ao banco de dados Aurora global.



Gerenciar um banco de dados global do Amazon Aurora

Realize a maioria das operações de gerenciamento nos clusters individuais que constituem um banco de dados global do Aurora. Ao selecionar Agrupar recursos relacionados na página Bancos de dados do console, você verá o cluster primário e os clusters secundários agrupados no banco de dados global associado. Para localizar as Regiões da AWS em que os clusters do banco de dados global estão sendo executados, o mecanismo e a versão do banco de dados Aurora, bem como seu identificador, use a guia Configuration (Configuração).

Os processos de failover entre regiões está disponível apenas para objetos do banco de dados global do Aurora, não para um único cluster de banco de dados do Aurora. Para saber mais, consulte Usar a transição ou o failover no Amazon Aurora Global Database.

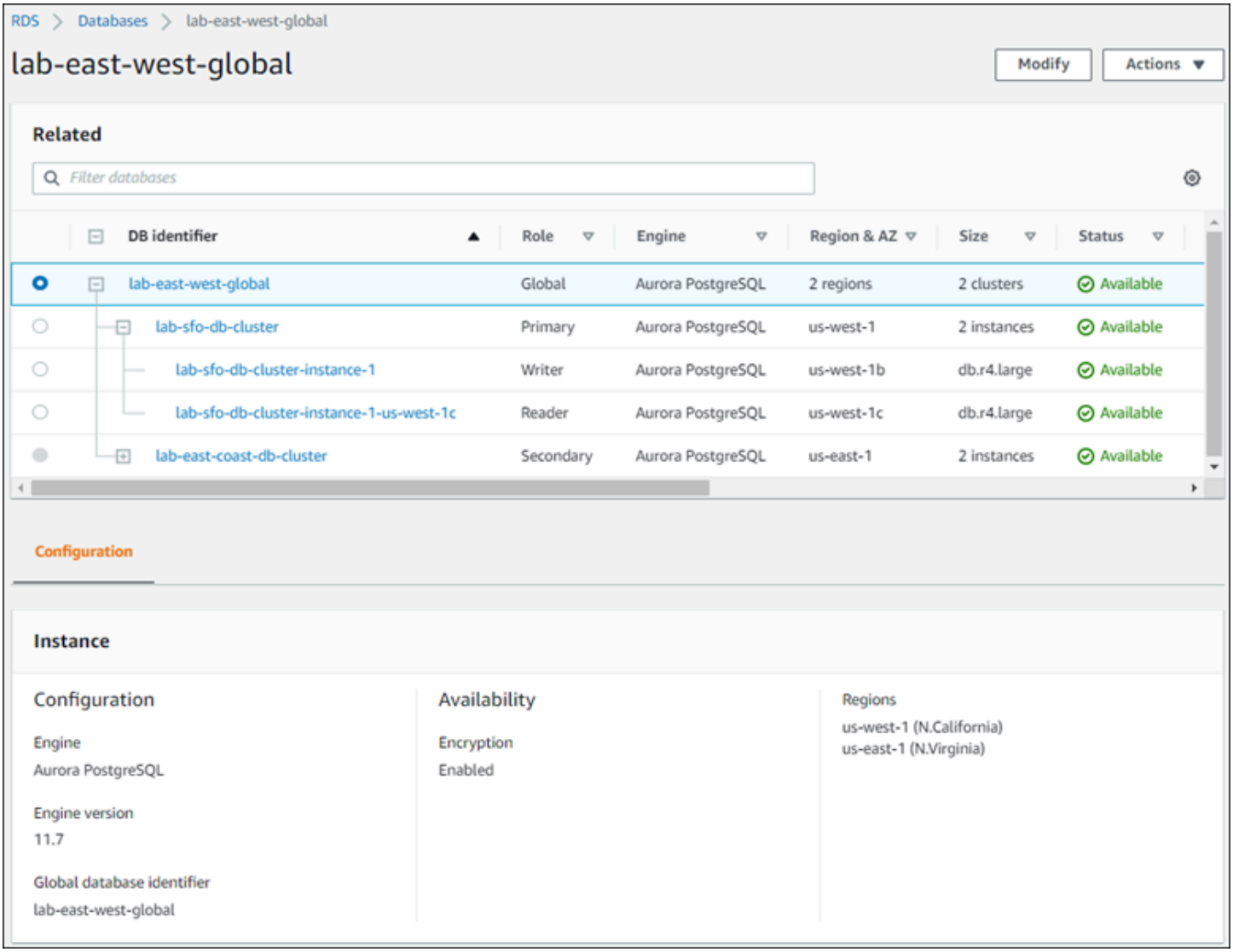
Para recuperar um banco de dados global Aurora de uma interrupção não planejada em sua região principal, consulte Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada.

Tópicos

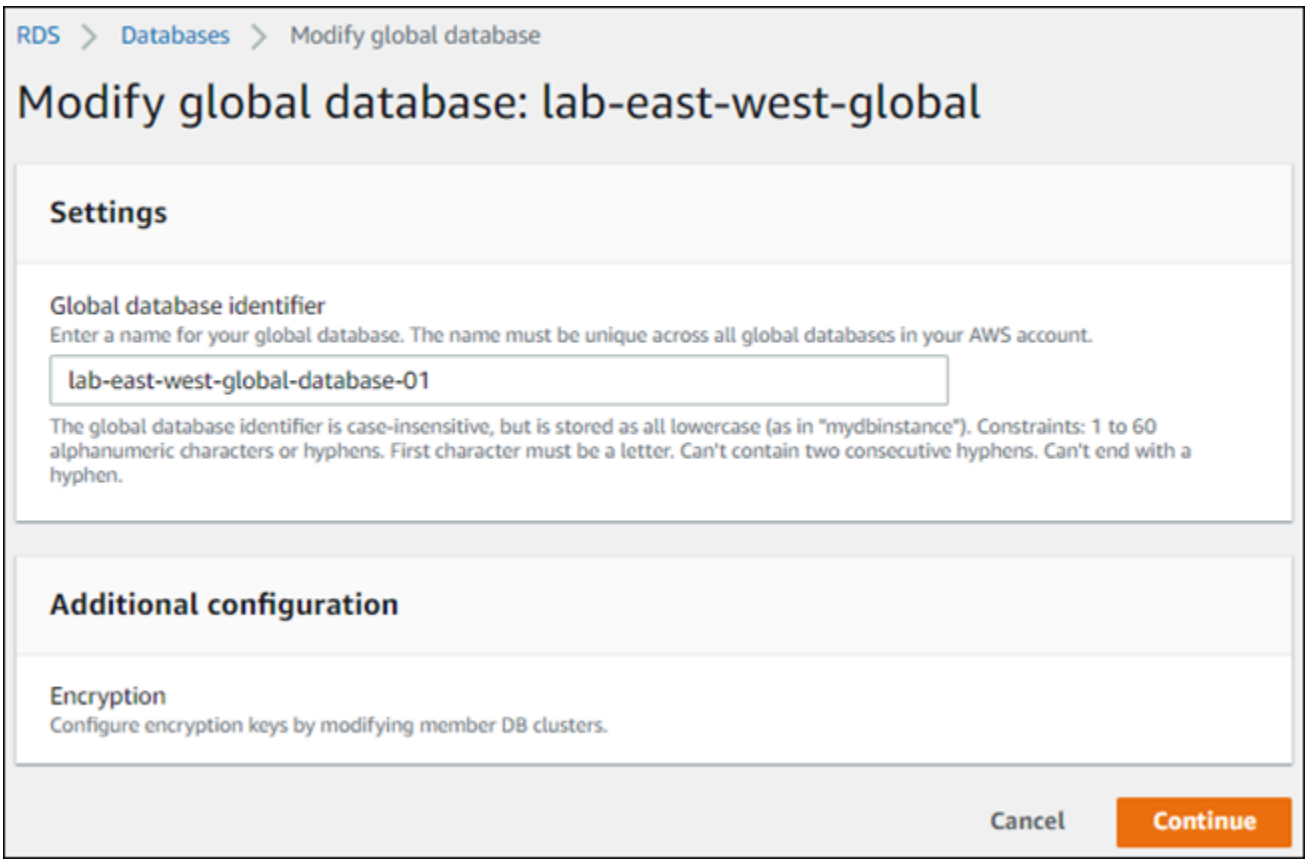
- Modificar um banco de dados global do Amazon Aurora
- Modificando parâmetros para um banco de dados do Aurora global
- Remover um cluster de um banco de dados global do Amazon Aurora
- Excluir um banco de dados global do Amazon Aurora
- Marcar os recursos do Amazon Aurora Global Database

Modificar um banco de dados global do Amazon Aurora

A página Databases (Bancos de dados) no Console de gerenciamento da AWS lista todos os bancos de dados globais do Aurora, exibindo os clusters primário e secundário de cada um. O banco de dados global Aurora tem suas próprias configurações. Especificamente, tem Regiões da AWS associadas a seus clusters principais e secundários, como mostrado na captura de tela a seguir.



Quando você faz alterações no banco de dados Aurora global, você tem a chance de cancelar as alterações, como mostrado na captura de tela a seguir.



Quando você escolhe Continue (Continuar), você confirma as alterações.

Modificando parâmetros para um banco de dados do Aurora global

Configure os grupos de parâmetros de banco de dados do Aurora de maneira independente para cada cluster do Aurora dentro do banco de dados global Aurora. A maioria dos parâmetros funciona da mesma maneira de outros tipos de clusters do Aurora. Recomendamos que você mantenha as configurações consistentes entre todos os clusters em um banco de dados global. Isso ajuda a evitar mudanças inesperadas de comportamento se você promover um cluster secundário para ser o primário.

Por exemplo, use as mesmas configurações para os fusos horários e os conjuntos de caracteres a fim de evitar um comportamento inconsistente caso um cluster diferente assuma como o cluster primário.

As definições de configuração `aurora_enable_repl_bin_log_filtering` e `aurora_enable_replica_log_compression` não entram em vigor.

Remover um cluster de um banco de dados global do Amazon Aurora

Você pode remover clusters de bancos de dados Aurora de seu banco de dados do Aurora global por vários motivos diferentes. Por exemplo, você pode querer remover um cluster de bancos de dados Aurora de um banco de dados do Aurora global se o cluster primário se tornar degradado ou isolado. Em seguida, ele se torna um cluster de banco de dados Aurora provisionado autônomo que pode ser usado para criar um novo banco de dados global Aurora. Para saber mais, consulte Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada.

Você também pode remover clusters do banco de dados do Aurora porque deseja excluir um banco de dados do Aurora global que você não precisa mais. Você não pode excluir o banco de dados global Aurora até depois de remover (desanexar) todos os clusters de banco de dados Aurora associados, deixando o primário por último. Para obter mais informações, consulte Excluir um banco de dados global do Amazon Aurora.

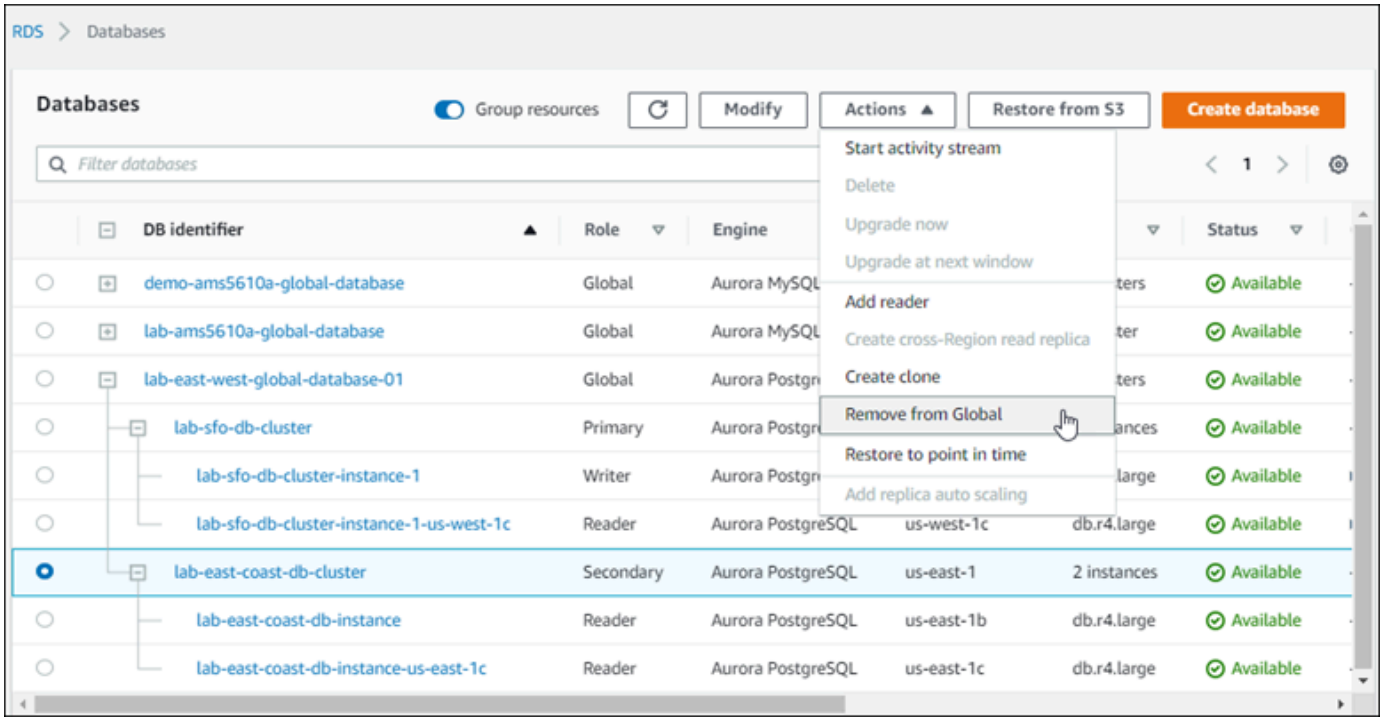
Quando um cluster de banco de dados Aurora é separado do banco de dados global Aurora, ele não é mais sincronizado com o primário. Ele se torna um cluster de banco de dados Aurora provisionado autônomo com recursos completos de leitura/gravação.

Você pode remover clusters de bancos de dados Aurora de seu banco de dados global usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

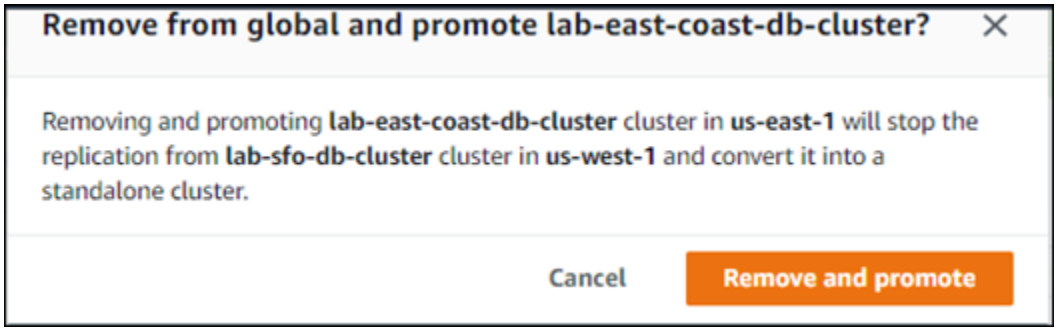
Console

Como remover um cluster do Aurora de um banco de dados global Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha o cluster na página Databases (Bancos de dados) .
3. Para Actions (Ações), selecione Remove from Global (Remover do global).

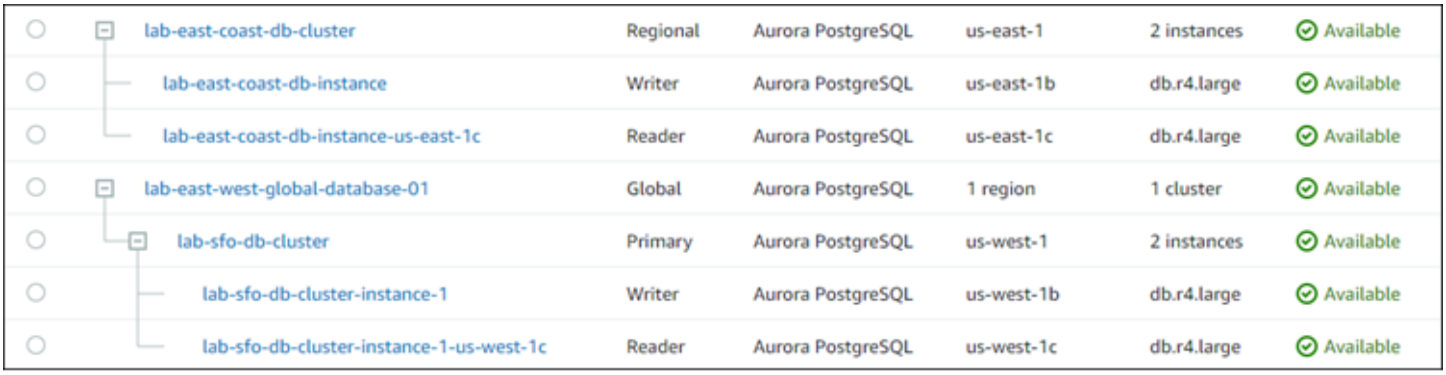


Você vê um prompt para confirmar que deseja desanexar o secundário do banco de dados Aurora global.



4. Escolha Remove and promote (Remover e promover) para remover o cluster do banco de dados global.

O cluster de bancos de dados Aurora não está mais servindo como um secundário no banco de dados do Aurora global e não é mais sincronizado com o cluster de banco de dados primário. É um cluster de bancos de dados Aurora autônomo com capacidade de leitura/gravação completa.



Depois de remover ou excluir todos os clusters secundários, remova o cluster primário da mesma maneira. Você não pode desanexar (remover) o cluster de banco de dados Aurora primário de um banco de dados Aurora global até depois de remover todos os clusters secundários.

O banco de dados global Aurora pode permanecer na lista Databases (Bancos de dados) com Regiões 0 e AZs. Você pode excluir se não quiser mais usar esse banco de dados do Aurora global. Para obter mais informações, consulte Excluir um banco de dados global do Amazon Aurora.

AWS CLI

Para remover um cluster Aurora do banco de dados global Aurora, execute o comando de CLI remove-from-global-cluster com os seguintes parâmetros:

- `--global-cluster-identifier` – O nome (identificador) do seu banco de dados Aurora global.
- `--db-cluster-identifier` – O nome de cada cluster de Aurora banco de dados a ser removido do banco de dados Aurora global. Remova todos os clusters de bancos de dados Aurora secundários antes de remover o primário.

Os exemplos a seguir primeiro removem um cluster secundário e o cluster primário de um banco de dados global Aurora.

Para Linux, macOS ou Unix:

```
aws rds --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_database_id

aws rds --region primary_region \
```

```
  remove-from-global-cluster \
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

Repita o comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* para cada Região da AWS secundária em seu Aurora Global Database.

Para Windows:

```
aws rds --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_database_id

aws rds --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_database_id
```

Repita o comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* para cada Região da AWS secundária em seu Aurora Global Database.

API do RDS

Para remover um cluster do Aurora de um banco de dados global Aurora com a API do RDS, execute a ação RemoveFromGlobalCluster.

Excluir um banco de dados global do Amazon Aurora

Como um banco de dados Aurora global normalmente mantém dados críticos de negócios, não é possível excluir o banco de dados global e os clusters associados em uma única etapa. Para excluir um banco de dados globalAurora, faça o seguinte:

- Remova todos os clusters de banco de dados secundários do banco de dados do Aurora global. Cada cluster se torna um cluster de banco de dados de Aurora autônomo. Para saber como, consulte Remover um cluster de um banco de dados global do Amazon Aurora.
- Em cada cluster de banco de dados Aurora autônomo, exclua todas as réplicas de Aurora.
- Remova o cluster do banco de dados primário do banco de dados Aurora global. Isso se torna um cluster de banco de dados Aurora autônomo.

- No cluster de banco de dados Aurora primário, primeiro exclua todas as réplicas do Aurora e, em seguida, exclua a instância de banco de dados do gravador.

A exclusão da instância do gravador do cluster de banco de dados Aurora autônomo recentemente também remove normalmente o cluster de bancos de dados Aurora e o banco de dados do Aurora global.

Para obter mais informações gerais, consulte Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.

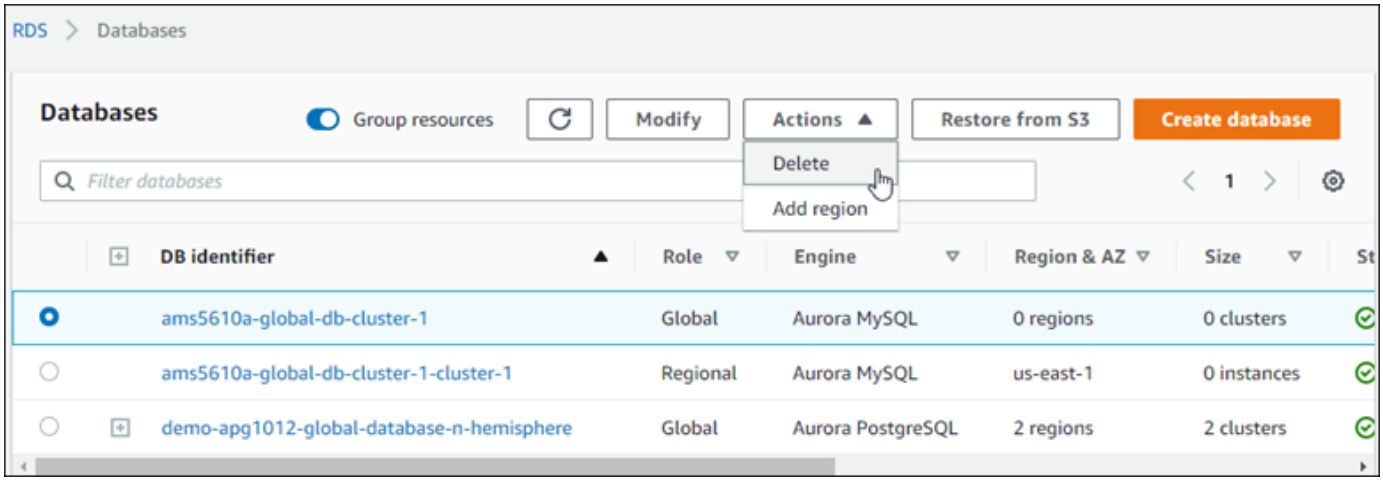
Para excluir um banco de dados global Aurora, você pode usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para excluir um banco de dados do Aurora global

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados) e localize o banco de dados do Aurora global que deseja excluir na listagem.
3. Confirme se todos os clusters foram removidos do banco de dados do Aurora global. O banco de dados global Aurora deve mostrar 0 regiões e AZs e um tamanho de 0 clusters.

   Se o banco de dados Aurora global contiver clusters de bancos de dados Aurora, você não poderá excluí-lo. Se necessário, desconecte os clusters de bancos de dados Aurora primário e secundário do banco de dados Aurora global. Para obter mais informações, consulte Remover um cluster de um banco de dados global do Amazon Aurora.
4. Escolha o banco de dados global do Aurora na lista e, em seguida, selecione Excluir no menu Ações.



AWS CLI

Para excluir um Aurora Global Database, execute o comando delete-global-cluster da CLI com o nome da Região da AWS e o identificador do Aurora Global Database, como mostra o exemplo a seguir.

Para Linux, macOS ou Unix:

```
aws rds --region primary_region delete-global-cluster \
   --global-cluster-identifier global_database_id
```

Para Windows:

```
aws rds --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_database_id
```

API do RDS

Para excluir um cluster que faça parte de um banco de dados do Aurora global, execute a operação de API DeleteGlobalCluster.

Marcar os recursos do Amazon Aurora Global Database

Com o recurso Aurora Global Database, é possível aplicar etiquetas do RDS a recursos em diferentes níveis em um banco de dados global. Se você não souber a forma como as etiquetas são usadas com a AWS ou os recursos do Aurora, consulte Marcar recursos do Amazon Aurora e do Amazon RDS antes de aplicar as etiquetas ao banco de dados global.

> **Note**
>
> Como os processos da AWS marcam dados como parte de seus mecanismos de relatório de custos, não inclua dados confidenciais nem informações de identificação pessoal (PII) nos nomes e nos valores das etiquetas.

É possível aplicar etiquetas aos seguintes tipos de recurso em um banco de dados global:

- O objeto de contêiner de todo o banco de dados global. Esse recurso é conhecido como cluster global.

  Depois de criar o cluster global realizando uma operação Adicionar região da AWS no console, é possível adicionar etiquetas na página de detalhes do cluster global. Na guia Etiquetas, na página de detalhes do cluster global, é possível adicionar, remover ou modificar etiquetas e seus valores associados escolhendo Gerenciar etiquetas.

  Com a AWS CLI e a API do RDS, é possível adicionar etiquetas ao cluster global no momento da criação. Também é possível adicionar, remover ou modificar etiquetas para um cluster global existente.
- O cluster primário. Aqui, use os mesmos procedimentos utilizados para trabalhar com etiquetas em clusters autônomos do Aurora. É possível configurar as etiquetas antes de transformar o cluster do Aurora original em um banco de dados global. Também é possível adicionar, remover ou modificar etiquetas e seus valores associados escolhendo Gerenciar etiquetas na guia Etiquetas da página de detalhes do cluster de banco de dados.
- Qualquer cluster secundário. Aqui, use os mesmos procedimentos utilizados para trabalhar com etiquetas em clusters autônomos do Aurora. É possível configurar as etiquetas ao mesmo tempo em que um cluster secundário do Aurora é criado com a ação Adicionar região da AWS no console. Também é possível adicionar, remover ou modificar etiquetas e seus valores associados escolhendo Gerenciar etiquetas na guia Etiquetas da página de detalhes do cluster de banco de dados.
- Instâncias de banco de dados individuais nos clusters primário ou secundário. Aqui, use os mesmos procedimentos utilizados para trabalhar com etiquetas em instâncias de banco de dados do Aurora ou do RDS. É possível configurar as etiquetas ao mesmo tempo em que uma nova instância de banco de dados é adicionada ao cluster do Aurora com a ação Adicionar leitor no console. Também é possível adicionar, remover ou modificar etiquetas e seus valores associados

escolhendo Gerenciar etiquetas na guia Etiquetas da página de detalhes da instância de banco de dados.

Apresentamos a seguir alguns exemplos dos tipos de etiqueta que podem ser atribuídos em um banco de dados global:

- É possível adicionar etiquetas ao cluster global para registrar informações gerais sobre a aplicação, como identificadores anônimos que representam proprietários e contatos na organização. É possível usar etiquetas para representar propriedades da aplicação que usa o banco de dados global.
- É possível adicionar etiquetas ao cluster primário e aos clusters secundários para monitorar os custos da aplicação ao nível da região da AWS. Para saber detalhes sobre esse procedimento, consulte Como o faturamento da AWS funciona com tags no Amazon RDS.
- É possível adicionar etiquetas a instâncias de banco de dados específicas com os clusters do Aurora para indicar sua finalidade especial. Por exemplo, no cluster primário, é possível ter uma instância do leitor com baixa prioridade de failover usada exclusivamente para geração de relatórios. Uma etiqueta pode distinguir essa instância de banco de dados para fins especiais de outras instâncias dedicadas a alta disponibilidade no cluster primário.
- É possível usar etiquetas em todos os níveis dos recursos do banco de dados global para controlar o acesso por meio de políticas do IAM. Para obter mais informações, consulte Controlar acesso aos recursos da AWS no Guia do usuário do Gerenciamento de Identidade e Acesso da AWS.

  > **Tip**
  >
  > No Console de gerenciamento da AWS, adicione etiquetas ao contêiner do cluster global como uma etapa separada depois de criá-lo. Se você quiser evitar intervalos de tempo em que o cluster global exista sem etiquetas de controle de acesso, poderá aplicar as etiquetas durante a operação `CreateGlobalCluster`, criando esse recurso por meio da AWS CLI, da API do RDS ou de um modelo do CloudFormation.

- É possível usar etiquetas ao nível do cluster, ou para o cluster global, a fim de registrar informações sobre garantia de qualidade e testes da aplicação. Por exemplo, é possível especificar uma etiqueta em um cluster de banco de dados para registrar a última vez que você realizou uma transição para esse cluster. É possível especificar uma etiqueta no cluster global para registrar a hora da última simulação de recuperação de desastres para toda a aplicação.

Exemplos de marcação para bancos de dados globais na AWS CLI

Os exemplos da AWS CLI a seguir mostram como especificar e examinar etiquetas para todos os tipos de recurso do Aurora no banco de dados global.

É possível especificar etiquetas para o contêiner do cluster global com o comando `create-global-cluster`. O exemplo a seguir cria um cluster global e atribui duas etiquetas a ele. As etiquetas têm as chaves `tag1` e `tag2`.

```
$  aws rds create-global-cluster --global-cluster-identifier my_global_cluster_id \
  --engine aurora-mysql --tags Key=tag1,Value=val1 Key=tag2,Value=val2
```

É possível listar as etiquetas no contêiner do cluster global com o comando `describe-global-clusters`. Ao trabalhar com etiquetas, você normalmente executa esse comando primeiro para recuperar o nome do recurso da Amazon (ARN) do cluster global. Você usará o ARN como parâmetro nos comandos subsequentes para trabalhar com etiquetas. O comando a seguir exibe as informações de etiquetas no atributo `TagList`. Também mostra o ARN, que é usado como parâmetro nos exemplos posteriores.

```
$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "GlobalClusterArn": "my_global_cluster_arn",
            ...
            "TagList": [
                {
                    "Value": "val1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                }
            ]
        }
    ]
}
```

É possível adicionar novas etiquetas com o comando `add-tags-to-resource`. Com esse comando, você especifica o nome do recurso da Amazon (ARN) do cluster global em vez do identificador. Adicionar uma etiqueta com o mesmo nome que uma etiqueta existente substitui o valor dessa etiqueta. Se você incluir espaços ou caracteres especiais nos valores da etiqueta, cite os valores conforme apropriado para o sistema operacional ou shell de comando. O exemplo a seguir modifica as etiquetas do cluster global do exemplo anterior. Originalmente, o cluster tinha etiquetas com as chaves `tag1` e `tag2`. Após a conclusão do comando, o cluster global terá uma nova etiqueta com a chave `tag3`, e a etiqueta com a chave `tag1` terá um valor diferente.

```
$  aws rds add-tags-to-resource --resource-name my_global_cluster_arn \
  --tags Key=tag1,Value="new value for tag1" Key=tag3,Value="entirely new tag"

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                },
                {
                    "Value": "entirely new tag",
                    "Key": "tag3"
                }
            ]
        }
    ]
}
```

É possível excluir uma etiqueta do cluster global com o comando `remove-tags-from-resource`. Com esse comando, você especifica somente um conjunto de chaves de etiqueta, sem nenhum valor de etiqueta. O exemplo a seguir modifica as etiquetas do cluster global do exemplo anterior.

Originalmente, o cluster tinha etiquetas com as chaves `tag1`, `tag2` e `tag3`. Depois que o comando terminar, somente a etiqueta com a chave `tag1` permanecerá.

```
$  aws rds remove-tags-from-resource --resource-name my_global_cluster_arn --tag-keys
 tag2 tag3

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                }
            ]
        }
    ]
}
```

Conectar-se ao Amazon Aurora Global Database

Cada Aurora Global Database vem com um endpoint do gravador que é atualizado automaticamente pelo Aurora para direcionar solicitações para a instância atual do gravador do cluster de banco de dados primário. Com o endpoint do gravador, não é necessário modificar a string de conexão depois de alterar a localização da região primária usando os recursos gerenciados de transição e de failover do Aurora Global Database. Para saber mais sobre como usar o endpoint do gravador com a transição e o failover do Aurora Global Database, consulte Usar a transição ou o failover no Amazon Aurora Global Database. Consulte informações sobre como se conectar a um Aurora Global Database com o RDS Proxy em Usa o RDS Proxy com bancos de dados globais do Aurora.

Tópicos

- Escolher o endpoint que atenda às necessidades da aplicação
- Visualizar os endpoints de um Amazon Aurora Global Database
- Considerações sobre o uso de endpoints do gravador globais

Escolher o endpoint que atenda às necessidades da aplicação

A conexão com um Aurora Global Database depende da necessidade de ler ou gravar no banco de dados e da região da AWS para a qual você deseja encaminhar as solicitações. Apresentamos a seguir alguns casos de uso típicos:

- Encaminhar solicitações para a instância do gravador: conecte-se ao banco de dados global Aurora se você precisar executar instruções de linguagem de manipulação de dados (DML) e linguagem de definição de dados (DDL) ou precisar de uma forte consistência entre leituras e gravações, conecte-se ao endpoint do gravador do Aurora Global Database. Esse endpoint encaminha as solicitações para a instância do gravador no cluster primário do banco de dados global. Esse endpoint é atualizado automaticamente para encaminhar solicitações para a instância do gravador, eliminando a necessidade de atualizar a aplicação sempre que você alterar a localização do gravador no cluster global. Também é possível usar o endpoint global para enviar solicitações de leitura/gravação entre regiões ao gravador.

  > **Note**
  >
  > Se você configurar o banco de dados global antes que o endpoint do gravador do Aurora Global Database esteja disponível, a aplicação poderá se conectar ao endpoint do cluster primário. Nesse caso, recomendamos trocar as configurações de conexão para usar o endpoint do gravador global. Isso evita a necessidade de alterar as configurações de conexão após cada transição ou failover do Aurora Global Database.
  > A primeira parte do nome do endpoint do gravador é o nome do Aurora Global Database. Portanto, se você renomear o Aurora Global Database, o nome do endpoint do gravador será alterado e todos os códigos que o usarem deverão ser atualizados com o novo nome.

- Escalar as leituras próximas da região da aplicação: para escalar solicitações somente leitura na mesma região da AWS que sua aplicação, ou em regiões próximas, conecte-se ao endpoint do leitor dos clusters primários ou secundários do Aurora.
- Escalar leituras com gravações ocasionais entre regiões: com relação a instruções DML ocasionais, como para manutenção e limpeza de dados, conecte-se ao endpoint do leitor de um cluster secundário que tenha o encaminhamento de gravação habilitado. Com o encaminhamento de gravação, o Aurora encaminha automaticamente as instruções de gravação para o gravador na região primária do Aurora Global Database. O encaminhamento de gravação oferece os seguintes benefícios:

- Não é necessário fazer o trabalho árduo de estabelecer conectividade entre os clusters secundário e primário para enviar gravações entre regiões.
- Não é necessário dividir solicitações de leitura e de gravação na aplicação.
- Não é necessário desenvolver uma lógica complexa para gerenciar a consistência das solicitações de leitura após gravação.

No entanto, com o encaminhamento de gravação, é necessário atualizar o código ou a configuração da aplicação para se conectar ao endpoint do leitor da região primária recém-promovida depois de realizar um failover ou uma transição entre regiões. Recomendamos monitorar a latência das operações realizadas por meio do encaminhamento de gravação com o objetivo de conferir as despesas do processamento das solicitações de gravação. Por fim, o encaminhamento de gravação não aceita determinadas operações do MySQL nem do PostgreSQL, como fazer alterações na linguagem de definição de dados (DDL) ou instruções `SELECT FOR UPDATE`.

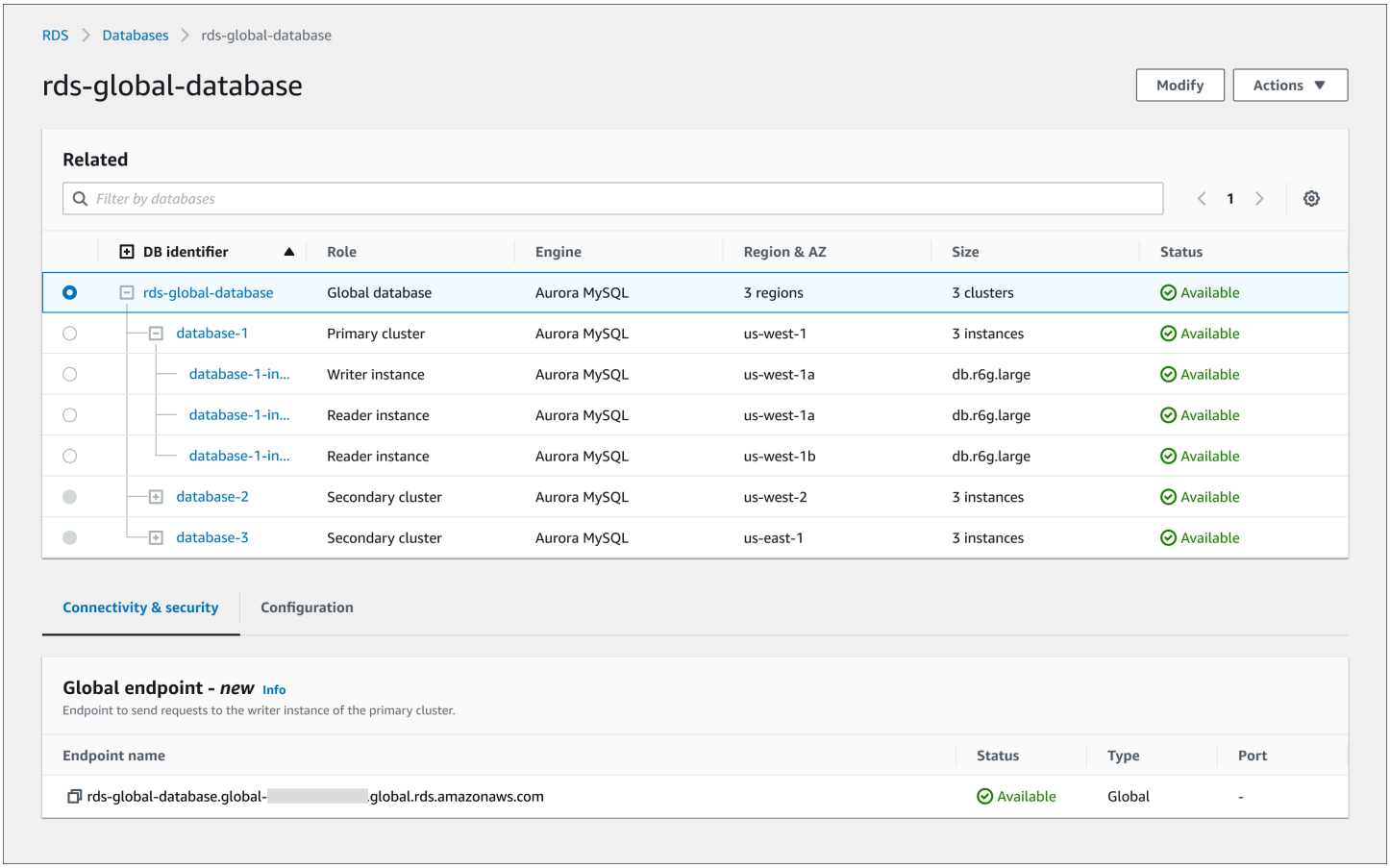
Para saber mais sobre como usar o encaminhamento de gravação entre regiões da AWS, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Para saber os detalhes sobre os diferentes tipos de endpoint do Aurora, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Visualizar os endpoints de um Amazon Aurora Global Database

Ao visualizar um Aurora Global Database no console, é possível ver todos os endpoints associados a todos os clusters. A figura a seguir mostra um exemplo dos tipos de endpoint exibidos quando você visualiza os detalhes do cluster de banco de dados primário:

- Gravador global: o único endpoint de leitura/gravação que sempre aponta para a instância de banco de dados do gravador atual do cluster de banco de dados global.
- Gravador: o endpoint de conexão para solicitações de leitura/gravação para o cluster de banco de dados primário no cluster de banco de dados global.
- Leitor: o endpoint de conexão para solicitações somente leitura para um cluster de banco de dados primário ou secundário no cluster de banco de dados global. Para minimizar a latência, escolha o endpoint leitor que está na sua Região da AWS ou na Região da AWS mais próxima de você.



Console

Como visualizar os endpoints de um banco de dados global

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Na lista, escolha o banco de dados global ou o cluster de banco de dados primário ou secundário cujos endpoints você deseja exibir.
4. Escolha a guia Segurança e conexão para ver os detalhes do endpoint. Os endpoints exibidos dependem do tipo de cluster que você selecionou, da seguinte forma:

   - Banco de dados global: o endpoint do gravador global.
   - Cluster de banco de dados primário: o endpoint do gravador global, o endpoint do cluster e o endpoint do leitor do cluster primário.
   - Cluster de banco de dados secundário: o endpoint do cluster e o endpoint do leitor para o cluster secundário. Em um cluster secundário, o endpoint do cluster exibe o status inativo

porque não processa solicitações de gravação. Ainda é possível se conectar ao endpoint do cluster, mas somente para consultas de leitura.

AWS CLI

Para visualizar o endpoint do gravador do cluster global, use o comando describe-global-clusters da AWS CLI, como no exemplo a seguir.

```
aws rds describe-global-clusters --region aws_region
{
    "GlobalClusters": [
        {
            "GlobalClusterIdentifier": "global_cluster_id",
            "GlobalClusterResourceId": "cluster-unique_string",
            "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:global_cluster_id",
            "Status": "available",
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",
            "GlobalClusterMembers": [

              ...
            ],
            "Endpoint":
 "global_cluster_id.global-unique_string.global.rds.amazonaws.com"
        }
    ]
}
```

Para visualizar os endpoints de cluster e do leitor dos clusters de banco de dados membros do cluster global, use o comando describe-db-clusters da AWS CLI, como no exemplo a seguir. Os valores exibidos para `Endpoint` e `ReaderEndpoint` são os endpoints do cluster e do leitor, respectivamente.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier db_cluster_id
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
```

```
                "az_1",
                "az_2",
                "az_3"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "db_cluster_id",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-08-01T18:21:11.301Z",
            "Endpoint":
 "db_cluster_id.cluster-unique_string.primary_region.rds.amazonaws.com",
            "ReaderEndpoint": "db_cluster_id.cluster-
ro-unique_string.primary_region.rds.amazonaws.com",
            "MultiAZ": false,
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",

            "ReadReplicaIdentifiers": [
                "arn:aws:rds:secondary_region:123456789012:cluster:db_cluster_id"
            ],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "db_instance_id",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                }
            ],

            ...
            "TagList": [],
            "GlobalWriteForwardingRequested": false
        }
    ]
}
```

API do RDS

Para visualizar o endpoint do gravador do cluster global, use a operação DescribeGlobalClusters da API do RDS. Para visualizar os endpoints de cluster e do leitor dos clusters de banco de dados membros do cluster global, use a operação DescribeDBClusters da API do RDS.

Considerações sobre o uso de endpoints do gravador globais

É possível fazer uso eficaz dos endpoints do gravador do Aurora Global Database seguindo as diretrizes e as práticas recomendadas abaixo:

- Para minimizar a interrupção após um failover ou uma transição entre regiões, é possível configurar a conectividade de VPC entre a computação da aplicação e as regiões primária e secundária da AWS. Por exemplo, suponha que você tenha aplicações ou sistemas clientes em execução na mesma VPC que o cluster primário. Se o cluster secundário for promovido, o endpoint do gravador global mudará automaticamente para apontar para esse cluster. Embora o endpoint do gravador global permita que você evite alterar as configurações de conexão da aplicação, as aplicações só poderão acessar os endereços IP na VPC da região da AWS recém-promovida a primária depois de configurar a rede entre as duas VPCs. Consulte Amazon VPC-to-Amazon VPC connectivity options a fim de avaliar diferentes opções para configurar essa conectividade.
- A atualização do endpoint do gravador global após uma transição ou um failover de banco de dados global pode levar muito tempo, dependendo da duração do armazenamento em cache do Serviço de Nomes de Domínio (DNS). Consulte o Manual do administrador do banco de dados Amazon Aurora MySQL para saber mais. O Aurora Global Database emite um evento do RDS quando percebe uma alteração de DNS no endpoint do gravador global. É possível usar o evento para criar estratégias e garantir que o cache do DNS não se estenda além do tempo após a geração do evento. Para obter mais informações, consulte Eventos de cluster de banco de dados.
- O Aurora Global Database replica os dados de forma assíncrona. Os métodos de failover entre regiões podem ocasionar a gravação dos dados da transação que não foram replicados para a secundária escolhida antes do início do failover. Embora o Aurora tente, da melhor forma possível, bloquear as gravações na região da AWS primária original, o failover pode ser suscetível a problemas de cérebro dividido. As considerações para minimizar a perda de dados e o risco de cérebro dividido também se aplicam aos endpoints do gravador do Aurora Global Database. Para obter mais informações, consulte Failovers planejados gerenciados para o Aurora Global Database.

Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora

Você pode reduzir o número de endpoints que precisa gerenciar para aplicativos executados em seu banco de dados global Aurora usando o encaminhamento de gravação. Com o encaminhamento

de gravação ativado, os clusters secundários em um banco de dados global Aurora encaminham instruções SQL que executam operações de gravação no cluster primário. O cluster primário atualiza a origem e, em seguida, propaga as alterações resultantes de volta a todas as regiões AWS secundárias.

A configuração de encaminhamento de gravação ajuda a impedir que você implemente seu próprio mecanismo para enviar operações de gravação de uma região secundária da AWS para a região primária. O Aurora lida com a configuração de rede entre regiões. O Aurora também transmite toda sessão necessária e o contexto transacional para cada instrução. Os dados são sempre alterados primeiro no cluster primário e, depois, replicados nos clusters secundários no banco de dados global Aurora. Desta forma, o cluster primário é sempre a fonte da revelação com a cópia mais atualizada de todos os dados.

Tópicos

- Como usar o encaminhamento de gravação em um banco de dados global do Aurora MySQL
- Usar o encaminhamento de gravação em um banco de dados global do Aurora PostgreSQL

Como usar o encaminhamento de gravação em um banco de dados global do Aurora MySQL

Tópicos

- Disponibilidade de região e versão do encaminhamento de gravação no Aurora MySQL
- Ativar o encaminhamento de gravação no Aurora MySQL
- Verificar se um cluster secundário está com o encaminhamento de gravação ativado no Aurora MySQL
- Compatibilidade do SQL e de aplicações com o encaminhamento de gravação no Aurora MySQL
- Isolamento e consistência para o encaminhamento de gravação no Aurora MySQL
- Executar instruções de várias partes com o encaminhamento de gravação no Aurora MySQL
- Transações com o encaminhamento de gravação no Aurora MySQL
- Parâmetros de configuração para o encaminhamento de gravação no Aurora MySQL
- Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora MySQL
- Variáveis de status do Aurora MySQL para encaminhamento de gravação

Disponibilidade de região e versão do encaminhamento de gravação no Aurora MySQL

O encaminhamento de gravação é compatível com o Aurora MySQL 2.08.1 e versões posteriores, em todas as regiões em que bancos de dados globais baseados no Aurora MySQL estão disponíveis.

Para ter informações sobre a disponibilidade de versões e regiões dos bancos de dados globais do Aurora, consulte Bancos de dados globais do Aurora com o Aurora MySQL.

Ativar o encaminhamento de gravação no Aurora MySQL

Por padrão, o encaminhamento de gravação não está habilitado quando você adiciona um cluster secundário a um banco de dados global Aurora.

Para habilitar o encaminhamento de gravação usando o Console de gerenciamento da AWS, marque a caixa de seleção Ativar o encaminhamento de gravação global em Encaminhamento de gravação de réplica de leitura ao adicionar uma região para um banco de dados global. Quanto a um cluster secundário existente, modifique o cluster para Ativar o encaminhamento de gravação global. Para desativar o encaminhamento de gravação, desmarque a caixa de seleção Ativar o encaminhamento de gravação global ao adicionar a região ou modificar o cluster secundário.

Para habilitar o encaminhamento de gravação usando a AWS CLI, use a opção `--enable-global-write-forwarding`. Essa opção funciona quando você cria um cluster secundário usando o comando `create-db-cluster`. Ela também funciona quando você modifica um cluster secundário existente usando o comando `modify-db-cluster`. Para isso, é necessário que o banco de dados global use uma versão do Aurora que ofereça suporte ao encaminhamento de gravação. É possível desabilitar o encaminhamento de gravação usando a opção `--no-enable-global-write-forwarding` com esses mesmos comandos da CLI.

Para habilitar o encaminhamento de gravação usando a API do Amazon RDS, defina o parâmetro `EnableGlobalWriteForwarding` como `true`. Esse parâmetro funciona quando você cria um cluster secundário usando a operação `CreateDBCluster`. Ele também funciona quando você modifica um cluster secundário existente usando a operação `ModifyDBCluster`. Para isso, é necessário que o banco de dados global use uma versão do Aurora que ofereça suporte ao encaminhamento de gravação. É possível desativar o encaminhamento de gravação definindo o parâmetro `EnableGlobalWriteForwarding` como `false`.

> **Note**
>
> Para que uma sessão de banco de dados use o encaminhamento de gravação, especifique uma configuração para o parâmetro de configuração `aurora_replica_read_consistency`. Faça isso em todas as sessões que usam o recurso de encaminhamento de gravação. Para obter informações sobre esse parâmetro, consulte Isolamento e consistência para o encaminhamento de gravação no Aurora MySQL. O recurso RDS Proxy não é compatível com o valor `SESSION` da variável `aurora_replica_read_consistency`. Definir esse valor pode causar um comportamento inesperado.

Os exemplos de CLI a seguir mostram como é possível configurar um banco de dados global Aurora com o encaminhamento de gravação habilitado ou desabilitado. Os itens destacados representam os comandos e as opções que são importantes especificar e manter consistentes ao configurar a infraestrutura para um banco de dados global Aurora.

O exemplo a seguir cria um banco de dados global Aurora, um cluster primário e um cluster secundário com o encaminhamento de gravação habilitado. Faça as substituições necessárias usando suas próprias opções de nome de usuário, senha e regiões principais e secundárias da AWS.

```
# Create overall global database.
aws rds create-global-cluster --global-cluster-identifier write-forwarding-test \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create primary cluster, in the same AWS Region as the global database.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-1 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --master-username user_name --master-user-password password \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1
```

```
aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create secondary cluster, in a different AWS Region than the global database,
# with write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2 \
  --enable-global-write-forwarding

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2
```

O exemplo a seguir continua a partir do anterior. Ele cria um cluster secundário sem o encaminhamento de gravação habilitado e, depois, habilita o esse recurso. Após a conclusão desse exemplo, todos os clusters secundários no banco de dados global estarão com o encaminhamento de gravação habilitado.

```
# Create secondary cluster, in a different AWS Region than the global database,
# without write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1
```

```
aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds modify-db-cluster --db-cluster-identifier write-forwarding-test-cluster-2 \
  --region us-east-2 \
  --enable-global-write-forwarding
```

Verificar se um cluster secundário está com o encaminhamento de gravação ativado no Aurora MySQL

Para determinar se é possível usar o encaminhamento de gravação em um cluster secundário, verifique se esse cluster tem o atributo `"GlobalWriteForwardingStatus": "enabled"`.

No Console de gerenciamento da AWS, na guia Configuração da página de detalhes do cluster, você vê o status Habilitado para Encaminhamento de gravação de réplica de leitura global.

Para ver o status da configuração de encaminhamento de gravação global para todos os clusters, execute o seguinte comando da AWS CLI.

Um cluster secundário mostra o valor `"enabled"` ou `"disabled"` para indicar se o encaminhamento de gravação está ativado ou desativado. Um valor de `null` indica que o encaminhamento de gravação não está disponível para esse cluster. Nesse caso, o cluster não faz parte de um banco de dados global ou é o cluster primário em vez de um cluster secundário. O valor também poderá ser `"enabling"` ou `"disabling"` se o encaminhamento de gravação estiver no processo de ser ativado ou desativado.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu

[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
```

```
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Para localizar apenas os clusters secundários que têm o encaminhamento de gravação global habilitado, execute o comando a seguir. Este comando também retorna o endpoint do leitor do cluster. Use o endpoint do leitor do cluster secundário ao usar o encaminhamento de gravação do secundário para o primário no banco de dados global Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilidade do SQL e de aplicações com o encaminhamento de gravação no Aurora MySQL

É possível usar os seguintes tipos de instruções SQL com o encaminhamento de gravação:

- Instruções de linguagem de manipulação de dados (DML), como `INSERT`, `DELETE` e `UPDATE`. Existem algumas restrições com relação às propriedades dessas instruções que podem ser usadas com o encaminhamento de gravação, conforme descrito a seguir.
- Instruções `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Instruções `PREPARE` e `EXECUTE`.

Certas instruções não são permitidas ou podem gerar resultados obsoletos ao serem usadas em um banco de dados global com o encaminhamento de gravação. Assim, a configuração `EnableGlobalWriteForwarding` é desativada por padrão para clusters secundários. Antes de ativá-la, verifique se o código do aplicativo não é afetado por nenhuma dessas restrições.

As restrições a seguir se aplicam às instruções SQL usadas com o encaminhamento de gravação. Em alguns casos, é possível usar as instruções em clusters secundários com o encaminhamento de gravação habilitado no nível do cluster. Essa abordagem funcionará se o encaminhamento de gravação não estiver ativado na sessão pelo parâmetro de configuração `aurora_replica_read_consistency`. Tentar usar uma instrução quando não é permitido devido ao encaminhamento de gravação exibe uma mensagem de erro com o seguinte formato.

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Linguagem de definição de dados (DDL)

Conecte-se ao cluster primário para executar essas instruções. Não é possível executá-las em instâncias de banco de dados do leitor.

Atualizar uma tabela permanente usando dados de uma tabela temporária

Você pode usar tabelas temporárias em clusters secundários com o encaminhamento de gravação habilitado. No entanto, não é possível usar uma instrução DML para modificar uma tabela permanente se a instrução se referir a uma tabela temporária. Por exemplo, não é possível usar uma instrução `INSERT ... SELECT` que usa os dados de uma tabela temporária. A tabela temporária existe no cluster secundário e não está disponível quando a instrução é executada no cluster primário.

Transações XA

Não é possível usar as instruções a seguir em um cluster secundário quando o encaminhamento de gravação está ativado na sessão. Essas instruções podem ser usadas em clusters secundários que não têm o encaminhamento de gravação habilitado ou em sessões em que a configuração `aurora_replica_read_consistency` está vazia. Antes de ativar o encaminhamento de gravação em uma sessão, verifique se o código usa essas instruções.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
```

```
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instruções LOAD para tabelas permanentes

Não é possível usar as instruções a seguir em um cluster secundário com o encaminhamento de gravação habilitado.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
        LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Você pode carregar dados em uma tabela temporária em um cluster secundário. No entanto, execute toda as instruções LOAD que se referem a tabelas permanentes apenas no cluster primário.

Instruções de plugin

Não é possível usar as instruções a seguir em um cluster secundário com o encaminhamento de gravação habilitado.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Instruções SAVEPOINT

Não é possível usar as instruções a seguir em um cluster secundário quando o encaminhamento de gravação está ativado na sessão. Você pode usar essas instruções em clusters secundários que não têm o encaminhamento de gravação habilitado ou em sessões em que a configuração `aurora_replica_read_consistency` está em branco. Verifique se o código usa essas instruções antes de ativar o encaminhamento de gravação em uma sessão.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Isolamento e consistência para o encaminhamento de gravação no Aurora MySQL

Em sessões que usam o encaminhamento de gravação, só é possível usar o nível de isolamento `REPEATABLE READ`. Embora também seja possível usar o nível de isolamento `READ COMMITTED`

com clusters somente leitura em regiões secundárias da AWS, esse nível de isolamento não funciona com o encaminhamento de gravação. Para obter informações sobre os níveis de isolamento `REPEATABLE READ` e `READ COMMITTED`, consulte Níveis de isolamento do Aurora MySQL.

É possível controlar o grau de consistência de leitura em um cluster secundário. O nível de consistência de leitura determina quanto o cluster secundário espera antes de cada operação de leitura para garantir que algumas ou todas as alterações sejam replicadas do cluster primário. Você pode ajustar o nível de consistência de leitura para garantir que todas as operações de gravação encaminhadas da sessão estejam visíveis no cluster secundário antes de qualquer consulta subsequente. Você também pode usar essa configuração para garantir que as consultas no cluster secundário sempre vejam as atualizações mais atuais do cluster primário. Isso acontece mesmo para aquelas submetidas por outras sessões ou outros clusters. Para especificar esse tipo de comportamento para o aplicativo, escolha um valor para o parâmetro de nível de sessão `aurora_replica_read_consistency`.

> **Important**
>
> Sempre defina o parâmetro `aurora_replica_read_consistency` para qualquer sessão para a qual você deseja encaminhar gravações. Caso contrário, o Aurora não habilita o encaminhamento de gravação para essa sessão. Esse parâmetro tem um valor vazio por padrão, então escolha um valor específico quando você usar esse parâmetro. O parâmetro `aurora_replica_read_consistency` tem efeito somente em clusters secundários nos quais o encaminhamento de gravação está habilitado.
> Para o Aurora MySQL versão 2 e versão 3 anterior à 3.04, use `aurora_replica_read_consistency` como uma variável de sessão. Para o Aurora MySQL versão 3.04 e posterior, você pode usar `aurora_replica_read_consistency` como uma variável de sessão ou como um parâmetro de cluster de banco de dados.

Para o parâmetro `aurora_replica_read_consistency`, é possível especificar os valores `EVENTUAL`, `SESSION` e `GLOBAL`.

À medida que você aumenta o nível de consistência, a aplicação passa mais tempo aguardando as alterações serem propagadas entre regiões da AWS. Você pode escolher o equilíbrio entre o tempo de resposta rápido e a garantia de que as alterações feitas em outros locais estejam totalmente disponíveis antes da execução das consultas.

Com a consistência de leitura definida como `EVENTUAL`, as consultas em uma região secundária da AWS que usa o encaminhamento de gravação podem ver dados ligeiramente obsoletos devido ao atraso de replicação. Os resultados das operações de gravação na mesma sessão não ficam visíveis até que a operação de gravação seja executada na região primária e replicada para a região atual. A consulta não espera que os resultados atualizados estejam disponíveis. Assim, ela pode recuperar os dados mais antigos ou os dados atualizados, dependendo do tempo das declarações e da quantidade de atraso da replicação.

Com a consistência de leitura definida como `SESSION`, todas as consultas em uma região secundária da AWS que usa o encaminhamento de gravação veem os resultados de todas as alterações feitas nessa sessão. As alterações são visíveis independentemente de a transação ser confirmada. Se necessário, a consulta aguardará que os resultados das operações de gravação encaminhadas sejam replicados para a região atual. Ele não aguarda resultados atualizados de operações de gravação realizadas em outras regiões ou em outras sessões da região atual.

Com a consistência de leitura definida como `GLOBAL`, uma sessão em uma região secundária da AWS vê as alterações feitas por essa sessão. Também vê todas as alterações confirmadas da região principais da AWS e das outras regiões secundárias da AWS. Cada consulta pode aguardar por um período que varia de acordo com a quantidade de atraso da sessão. A consulta prossegue quando o cluster secundário está atualizado com todos os dados confirmados do cluster primário, a partir do momento em que a consulta foi iniciada.

Para obter mais informações sobre todos os parâmetros envolvidos no encaminhamento de gravação, consulte Parâmetros de configuração para o encaminhamento de gravação no Aurora MySQL.

Exemplos de uso do encaminhamento de gravação

Esses exemplos usam `aurora_replica_read_consistency` como uma variável de sessão. Para o Aurora MySQL versão 3.04 e posterior, você pode usar `aurora_replica_read_consistency` como uma variável de sessão ou como um parâmetro de cluster de banco de dados.

No exemplo a seguir, o cluster primário está na Região da US East (N. Virginia). O cluster secundário está na região da Leste dos EUA (Ohio). O exemplo mostra os efeitos da execução de instruções `INSERT` seguidas por instruções `SELECT`. Dependendo do valor da configuração `aurora_replica_read_consistency`, os resultados podem diferir dependendo do tempo das instruções. Para obter maior consistência, você pode esperar brevemente antes de emitir a instrução

SELECT. Ou Aurora pode esperar automaticamente até que os resultados terminem a replicação antes de prosseguir SELECT.

Neste exemplo, há uma configuração de consistência de leitura de `eventual`. Executar uma instrução INSERT imediatamente seguida por uma instrução SELECT ainda retorna o valor de `COUNT(*)`. Esse valor reflete o número de linhas antes que a nova linha seja inserida. Executar SELECT novamente, pouco tempo depois, retornará a contagem de linhas atualizada. As declarações SELECT não aguardam.

```
mysql> set aurora_replica_read_consistency = 'eventual';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Com uma configuração de consistência de leitura de `session`, uma declaração SELECT imediatamente após INSERT aguarda até que as alterações da declaração INSERT sejam visíveis. Declarações SELECT subsequentes não aguardam.

```
mysql> set aurora_replica_read_consistency = 'session';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
```

```
+----------+
1 row in set (0.01 sec)
mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Com a configuração de consistência de leitura ainda definida como `session`, a introdução de uma breve espera após a execução de uma instrução `INSERT` torna a contagem de linhas atualizada disponível no momento em que a próxima instrução `SELECT` é executada.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Com uma configuração de consistência de leitura de `global`, cada instrução `SELECT` aguarda para garantir que todas as alterações de dados a partir da hora de início da instrução estejam visíveis antes de executar a consulta. A quantidade de espera por cada instrução `SELECT` varia de acordo com a quantidade de atraso de replicação entre os clusters primário e secundário.

```
mysql> set aurora_replica_read_consistency = 'global';
```

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Executar instruções de várias partes com o encaminhamento de gravação no Aurora MySQL

Uma instrução DML pode consistir em várias partes, como uma instrução `INSERT ... SELECT` ou uma instrução `DELETE ... WHERE`. Nesse caso, a instrução inteira é encaminhada para o cluster primário e executada nele.

Transações com o encaminhamento de gravação no Aurora MySQL

Se a transação é encaminhada para o cluster primário depende do modo de acesso da transação. É possível especificar o modo de acesso da transação usando a instrução `SET TRANSACTION` ou a instrução `START TRANSACTION`. Você também pode especificar o modo de acesso da transação alterando o valor da sessão transaction_read_only. Você só pode alterar esse valor de sessão enquanto estiver conectado a um cluster de banco de dados que tenha o encaminhamento de gravação habilitado.

Se uma transação de longa duração não emitir nenhuma instrução por um período substancial, ela poderá exceder o tempo limite ocioso. Este período tem um padrão de um minuto. Você pode aumentá-lo para até um dia. Uma transação que excede o tempo limite ocioso é cancelada pelo

cluster primário. A instrução subsequente enviada recebe um erro de tempo limite. Depois, o Aurora reverte a transação.

Esse tipo de erro pode ocorrer em outros casos quando o encaminhamento de gravação fica indisponível. Por exemplo, o Aurora cancelará todas as transações que usam o encaminhamento de gravação se você reiniciar o cluster primário ou se desativar a configuração de encaminhamento de gravação.

Parâmetros de configuração para o encaminhamento de gravação no Aurora MySQL

Os parameter groups de cluster do Aurora incluem configurações para o recurso de encaminhamento de gravação. Como são parâmetros de cluster, todas as instâncias de banco de dados em cada cluster têm os mesmos valores para essas variáveis. Detalhes sobre esses parâmetros são resumidos na tabela a seguir, com notas de uso após a tabela.

Nome	Escopo	Type	Valor padrão	Valores válidos
`aurora_fwd_master_idle_timeout` (Aurora MySQL versão 2)	Global	inteiro não assinado	60	1–86.400
`aurora_fwd_master_max_connections_pct` (Aurora MySQL versão 2)	Global	inteiro longo não assinado	10	0–90
`aurora_fwd_writer_idle_timeout` (Aurora MySQL versão 3)	Global	inteiro não assinado	60	1–86.400
`aurora_fwd_writer_max_connections_pct` (Aurora MySQL versão 3)	Global	inteiro longo não assinado	10	0–90
`aurora_replica_read_consistency`	Sessão para versão 2 e versão 3 inferior a 3.04, Global	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Nome	Escopo	Type	Valor padrão	Valores válidos
	para versão 3.04 e posterior			

Para controlar as solicitações de gravação recebidas de clusters secundários, use estas configurações no cluster primário:

- `aurora_fwd_master_idle_timeout`, `aurora_fwd_writer_idle_timeout`: o número de segundos que o cluster primário aguarda atividades em uma conexão encaminhada de um cluster secundário antes de a encerrar. Se a sessão permanecer ociosa além desse período, o Aurora cancelará a sessão.
- `aurora_fwd_master_max_connections_pct`, `aurora_fwd_writer_max_connections_pct`: o limite máximo em conexões de banco de dados que podem ser utilizadas em uma instância de banco de dados de gravador para lidar com consultas encaminhadas de leitores. Ele é expresso como uma porcentagem da configuração `max_connections` para a instância de banco de dados de gravador no cluster primário. Por exemplo, se `max_connections` for 800 e `aurora_fwd_master_max_connections_pct` ou `aurora_fwd_writer_max_connections_pct` for 10, o gravador permitirá um máximo de 80 sessões encaminhadas simultâneas. Essas conexões vêm do mesmo grupo de conexões gerenciado pela configuração `max_connections`.

  Essa configuração se aplica somente ao cluster primário, quando um ou mais clusters secundários têm o encaminhamento de gravação habilitado. Se você diminuir o valor, as conexões existentes não serão afetadas. O Aurora leva o novo valor da configuração em conta ao tentar criar uma nova conexão a partir de um cluster secundário. O valor padrão é 10, representando 10% do valor `max_connections`. Se você habilitar o encaminhamento de consulta em qualquer um dos clusters secundários, essa configuração deverá ter um valor diferente de zero para que as operações de gravação de clusters secundários sejam bem-sucedidas. Se o valor for zero, as operações de gravação receberão o código de erro `ER_CON_COUNT_ERROR` com a mensagem `Not enough connections on writer to handle your request`.

O parâmetro `aurora_replica_read_consistency` habilita o encaminhamento de gravação. Você o usa em cada sessão. Você pode especificar `EVENTUAL`, `SESSION` ou `GLOBAL` no nível de consistência de leitura. Para saber mais sobre os níveis de consistência, consulte Isolamento e consistência para o encaminhamento de gravação no Aurora MySQL. As seguintes regras se aplicam a esse parâmetro:

- O valor padrão é '' (vazio).
- O encaminhamento de gravação só estará disponível em uma sessão se `aurora_replica_read_consistency` estiver definido como `EVENTUAL`, `SESSION` ou `GLOBAL`. Esse parâmetro é relevante somente em instâncias de leitor de clusters secundários que têm o encaminhamento de gravação habilitado e que estão em um banco de dados global Aurora.
- Você não pode definir essa variável (quando vazia) ou não definida (quando já estiver definida) dentro de uma transação de várias instruções. No entanto, você pode alterá-lo de um valor válido (`EVENTUAL`,`SESSION` ou `GLOBAL`) para outro valor válido (`EVENTUAL`, `SESSION` ou `GLOBAL`) durante essa transação.
- A variável não pode ser `SET` quando o encaminhamento de gravação não está habilitado no cluster secundário.

Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora MySQL

As métricas do Amazon CloudWatch a seguir se aplicam ao cluster primário quando você usa o encaminhamento de gravação em um ou mais clusters secundários. Essas métricas são todas medidas na instância de banco de dados de gravador no cluster primário.

Métrica do CloudWatch	Unidade	Descrição
`AuroraDMLRejectedMasterFull`	Contagem	O número de consultas encaminhadas que são rejeitadas porque a sessão está cheia na instância de banco de dados do gravador. Para o Aurora MySQL versão 2.

Métrica do CloudWatch	Unidade	Descrição
`AuroraDMLRejectedWriterFull`	Contagem	O número de consultas encaminhadas que são rejeitadas porque a sessão está cheia na instância de banco de dados do gravador. Para o Aurora MySQL versão 3.
`ForwardingMasterDMLLatency`	Milissegundos	Tempo médio para processar cada declaração DML encaminhada na instância de banco de dados de gravador. Não inclui o tempo para o cluster secundário encaminhar a solicitação de gravação nem o tempo para replicar as alterações de volta no cluster secundário. Para o Aurora MySQL versão 2.
`ForwardingMasterDMLThroughput`	Contagem por segundo	Número de instruções DML encaminhadas processadas a cada segundo por essa instância de banco de dados de gravador. Para o Aurora MySQL versão 2.

Métrica do CloudWatch	Unidade	Descrição
`ForwardingMasterOpenSessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador. Para o Aurora MySQL versão 2.
`ForwardingWriterDMLLatency`	Milissegundos	Tempo médio para processar cada declaração DML encaminhada na instância de banco de dados de gravador. Não inclui o tempo para o cluster secundário encaminhar a solicitação de gravação nem o tempo para replicar as alterações de volta no cluster secundário. Para o Aurora MySQL versão 3.
`ForwardingWriterDMLThroughput`	Contagem por segundo	Número de instruções DML encaminhadas processadas a cada segundo por essa instância de banco de dados de gravador. Para o Aurora MySQL versão 3.
`ForwardingWriterOpenSessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador. Para o Aurora MySQL versão 3.

As métricas do CloudWatch a seguir se aplicam a cada cluster secundário. Essas métricas são medidas em cada instância de banco de dados de leitor em um cluster secundário com o encaminhamento de gravação habilitado.

Métrica do CloudWatch	Unidade	Descrição
`ForwardingReplicaDMLLatency`	Milissegundos	Tempo médio de resposta de DMLs encaminhadas na réplica.
`ForwardingReplicaDMLThroughput`	Contagem por segundo	Número de instruções DML encaminhadas processadas por segundo.
`ForwardingReplicaOpenSessions`	Contagem	O número de sessões que estão usando o encaminhamento de gravação em uma instância de banco de dados do leitor.
`ForwardingReplicaReadWaitLatency`	Milissegundos	Tempo médio de espera que uma declaração `SELECT` em uma instância de banco de dados do leitor aguarda para alcançar o cluster primário. O grau em que a instância de banco de dados de leitor aguarda antes de processar uma consulta depende da configuração `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Contagem por segundo	Número total de instruções `SELECT` processadas a cada segundo em todas as sessões

Métrica do CloudWatch	Unidade	Descrição
		que estão encaminhando gravações.
`ForwardingReplicaSelectLatency`	Milissegundos	Encaminhamento de latência de `SELECT`, média sobre todas as instruções `SELECT` encaminhadas dentro do período de monitoramento.
`ForwardingReplicaSelectThroughput`	Contagem por segundo	Encaminhamento de throughput de `SELECT` por média de segundos no período de monitoramento.

Variáveis de status do Aurora MySQL para encaminhamento de gravação

As variáveis de status do Aurora MySQL a seguir se aplicam ao cluster primário quando você usa o encaminhamento de gravação em um ou mais clusters secundários. Essas métricas são todas medidas na instância de banco de dados de gravador no cluster primário.

Variável de status do Aurora MySQL	Unidade	Descrição
`Aurora_fwd_master_dml_stmt_count`	Contagem	Número total de instruções DML encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 2.
`Aurora_fwd_master_dml_stmt_duration`	Microssegundos	Duração total das instruções DML encaminhadas para essa instância de banco de dados de gravador.

Variável de status do Aurora MySQL	Unidade	Descrição
		Para o Aurora MySQL versão 2.
`Aurora_fwd_master_open_sessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador. Para o Aurora MySQL versão 2.
`Aurora_fwd_master_select_stmt_count`	Contagem	Número total de instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 2.
`Aurora_fwd_master_select_stmt_duration`	Microssegundos	Duração total das instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 2.
`Aurora_fwd_writer_dml_stmt_count`	Contagem	Número total de instruções DML encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 3.

Variável de status do Aurora MySQL	Unidade	Descrição
`Aurora_fwd_writer_dml_stmt_duration`	Microssegundos	Duração total das instruções DML encaminhadas para essa instância de banco de dados de gravador.
`Aurora_fwd_writer_open_sessions`	Contagem	Número de sessões encaminhadas na instância de banco de dados de gravador. Para o Aurora MySQL versão 3.
`Aurora_fwd_writer_select_stmt_count`	Contagem	Número total de instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 3.
`Aurora_fwd_writer_select_stmt_duration`	Microssegundos	Duração total das instruções `SELECT` encaminhadas para essa instância de banco de dados de gravador. Para o Aurora MySQL versão 3.

As variáveis de status do Aurora MySQL a seguir se aplicam a cada cluster secundário. Essas métricas são medidas em cada instância de banco de dados de leitor em um cluster secundário com o encaminhamento de gravação habilitado.

Variável de status do Aurora MySQL	Unidade	Descrição
`Aurora_fwd_replica_dml_stmt_count`	Contagem	Número total de instruções DML encaminhadas dessa instância de banco de dados de leitor.
`Aurora_fwd_replica_dml_stmt_duration`	Microssegundos	Duração total de todas as instruções DML encaminhadas dessa instância de banco de dados de leitor.
`Aurora_fwd_replica_errors_session_limit`	Contagem	Número de sessões rejeitadas pelo cluster primário devido a uma das seguintes condições de erro: • gravador completo • Muitas declarações encaminhadas em andamento.
`Aurora_fwd_replica_open_sessions`	Contagem	O número de sessões que estão usando o encaminhamento de gravação em uma instância de banco de dados do leitor.
`Aurora_fwd_replica_read_wait_count`	Contagem	Número total de esperas de leitura-após-gravação nessa instância de banco de dados de leitor.
`Aurora_fwd_replica_read_wait_duration`	Microssegundos	Duração total das esperas devido à configuração de consistência de leitura nessa

Variável de status do Aurora MySQL	Unidade	Descrição
		instância de banco de dados de leitor.
`Aurora_fwd_replica_select_stmt_count`	Contagem	Número total de instruções `SELECT` encaminhadas dessa instância de banco de dados de leitor.
`Aurora_fwd_replica_select_stmt_duration`	Microssegundos	Duração total das instruções `SELECT` encaminhadas dessa instância de banco de dados de leitor.

Usar o encaminhamento de gravação em um banco de dados global do Aurora PostgreSQL

Tópicos

- Disponibilidade de região e versão do encaminhamento de gravação no Aurora PostgreSQL
- Ativar o encaminhamento de gravação no Aurora PostgreSQL
- Verificar se um cluster secundário está com o encaminhamento de gravação ativado no Aurora PostgreSQL
- Compatibilidade do SQL e de aplicações com o encaminhamento de gravação no Aurora PostgreSQL
- Isolamento e consistência para o encaminhamento de gravação no Aurora PostgreSQL
- Modos de acesso a transações com encaminhamento de gravação
- Executar instruções de várias partes com o encaminhamento de gravação no Aurora PostgreSQL
- Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL
- Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora PostgreSQL
- Eventos de espera para encaminhamento de gravação no Aurora PostgreSQL

Disponibilidade de região e versão do encaminhamento de gravação no Aurora PostgreSQL

No Aurora PostgreSQL versão 16 e versões principais posteriores, o encaminhamento global de gravação é possível em todas as versões secundárias. Para versões anteriores do Aurora PostgreSQL, o encaminhamento global de gravação é possível na versão 15.4 e em versões secundárias posteriores e na versão 14.9 e em versões secundárias posteriores. O encaminhamento de gravação está disponível em todas as regiões da AWS em que os bancos de dados globais baseados no Aurora PostgreSQL estão disponíveis.

Para ter mais informações sobre a disponibilidade de versões e regiões dos bancos de dados globais Aurora, consulte Bancos de dados globais do Aurora com o Aurora PostgreSQL.

Ativar o encaminhamento de gravação no Aurora PostgreSQL

Por padrão, o encaminhamento de gravação não está habilitado quando você adiciona um cluster secundário a um banco de dados global Aurora. Você pode ativar o encaminhamento de gravação para seu cluster de banco de dados secundário enquanto o cria ou a qualquer momento depois de criá-lo. Se necessário, você pode desativá-lo mais tarde. Ativar ou desativar o encaminhamento de gravação não causa tempo de inatividade ou reinicialização.

> **Note**
>
> É possível usar o encaminhamento de gravação local para as aplicações que têm gravações ocasionais e exigem consistência de leitura após gravação, que é a capacidade de ler a gravação mais recente em uma transação.

Console

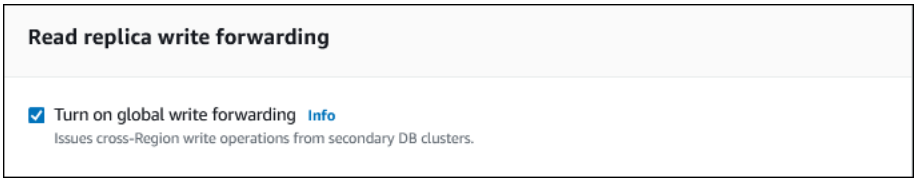
No console, você pode ativar ou desativar o encaminhamento de gravação ao criar ou modificar um cluster de banco de dados secundário.

Ativar ou desativar o encaminhamento de gravação ao criar um cluster de banco de dados secundário

Ao criar um cluster de banco de dados secundário, você ativa o encaminhamento de gravação marcando a caixa de seleção Ativar encaminhamento de gravação global em Ler o encaminhamento de gravação da réplica. Para desativá-lo, desmarque a caixa de seleção. Para criar um cluster de

banco de dados secundário, siga as instruções relacionadas ao seu mecanismo de banco de dados em Criar um cluster de bancos de dados do Amazon Aurora.

A captura de tela a seguir mostra a seção Ler o encaminhamento de gravação da réplica com a caixa de seleção Ativar encaminhamento de gravação global marcada.



Ativar ou desativar o encaminhamento de gravação ao modificar um cluster de banco de dados secundário

No console, você pode modificar um cluster de banco de dados secundário ou desativar o encaminhamento de gravação.

No console, para ativar ou desativar o encaminhamento de gravação para um cluster de banco de dados secundário

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados secundário e selecione Modificar.
4. Na seção Ler o encaminhamento de gravação da réplica, marque ou desmarque a caixa de seleção Ativar encaminhamento de gravação global.
5. Escolha Continuar.
6. Em Programar modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção agendada, o Aurora ignorará essa configuração e ativará o encaminhamento de gravação imediatamente.
7. Escolha Modificar Cluster.

AWS CLI

Para ativar o encaminhamento de gravação pela AWS CLI, use a opção `--enable-global-write-forwarding`. Essa opção funciona quando você cria um cluster secundário usando o comando create-db-cluster. Ela também funciona quando você modifica um cluster secundário existente usando o comando modify-db-cluster. Para isso, é necessário que o banco de dados

global use uma versão do Aurora que ofereça suporte ao encaminhamento de gravação. É possível desabilitar o encaminhamento de gravação usando a opção `--no-enable-global-write-forwarding` com esses mesmos comandos da CLI.

Os procedimentos a seguir descrevem como ativar ou desativar o encaminhamento de gravação para um cluster de banco de dados secundário no cluster global usando a AWS CLI.

Para ativar ou desativar o encaminhamento de gravação para um cluster de banco de dados secundário existente

- Chame o comando modify-db-cluster AWS CLI e forneça os seguintes valores:
  - `--db-cluster-identifier`: o nome do cluster de banco de dados.
  - `--enable-global-write-forwarding` para ativar ou `--no-enable-global-write-forwarding` para desativar.

  O exemplo a seguir permite o encaminhamento de gravação para o cluster `sample-secondary-db-cluster` de banco de dados.

  Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-secondary-db-cluster \
    --enable-global-write-forwarding
```

  Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-secondary-db-cluster ^
    --enable-global-write-forwarding
```

API do RDS

Para habilitar o encaminhamento de gravação usando a API do Amazon RDS, defina o parâmetro `EnableGlobalWriteForwarding` como `true`. Esse parâmetro funciona quando você cria um cluster secundário usando a operação CreateDBCluster. Ele também funciona quando você modifica um cluster secundário existente usando a operação ModifyDBCluster. Para isso, é necessário que o banco de dados global use uma versão do Aurora que ofereça suporte ao encaminhamento

de gravação. É possível desabilitar o encaminhamento de gravação definindo o parâmetro `EnableGlobalWriteForwarding` como `false`.

Verificar se um cluster secundário está com o encaminhamento de gravação ativado no Aurora PostgreSQL

Para determinar se é possível usar o encaminhamento de gravação em um cluster secundário, verifique se esse cluster tem o atributo `"GlobalWriteForwardingStatus": "enabled"`.

No Console de gerenciamento da AWS, é exibido `Read replica write forwarding` na guia Configuração da página de detalhes do cluster. Para ver o status da configuração de encaminhamento de gravação global para todos os clusters, execute o seguinte comando da AWS CLI.

Um cluster secundário mostra o valor `"enabled"` ou `"disabled"` para indicar se o encaminhamento de gravação está ativado ou desativado. Um valor de `null` indica que o encaminhamento de gravação não está disponível para esse cluster. Nesse caso, o cluster não faz parte de um banco de dados global ou é o cluster primário em vez de um cluster secundário. O valor também poderá ser `"enabling"` ou `"disabling"` se o encaminhamento de gravação estiver no processo de ser ativado ou desativado.

Example

```
aws rds describe-db-clusters --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Para localizar apenas os clusters secundários que têm o encaminhamento de gravação global habilitado, execute o comando a seguir. Este comando também retorna o endpoint do leitor do cluster. Use o endpoint do leitor do cluster secundário ao usar o encaminhamento de gravação do secundário para o primário no banco de dados global Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilidade do SQL e de aplicações com o encaminhamento de gravação no Aurora PostgreSQL

Certas instruções não são permitidas ou podem gerar resultados obsoletos ao serem usadas em um banco de dados global com o encaminhamento de gravação. Além disso, funções definidas pelo usuário e procedimentos definidos pelo usuário não são compatíveis. Assim, a configuração `EnableGlobalWriteForwarding` é desativada por padrão para clusters secundários. Antes de ativá-la, verifique se o código do aplicativo não é afetado por nenhuma dessas restrições.

É possível usar os seguintes tipos de instruções SQL com o encaminhamento de gravação:

- Instruções de linguagem de manipulação de dados (DML), como `INSERT`, `DELETE` e `UPDATE`.
- `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`Instruções
- Instruções `PREPARE` e `EXECUTE`.
- Instruções `EXPLAIN` com as instruções desta lista

Os seguintes tipos de instruções SQL não são compatíveis com o encaminhamento de gravação:

- Instruções Data Definition Language (DDL)

- ANALYZE
- CLUSTER
- COPY
- Cursores: os cursores não são compatíveis, então feche-os antes de usar o encaminhamento de gravação.
- GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL
- LOCK
- SAVEPOINTInstruções
- SELECT INTO
- SET CONSTRAINTS
- TRUNCATE
- VACUUM

Isolamento e consistência para o encaminhamento de gravação no Aurora PostgreSQL

Em sessões que usam o encaminhamento de gravação, é possível usar os níveis de isolamento REPEATABLE READ e READ COMMITTED. No entanto, o nível de isolamento SERIALIZABLE não é compatível.

É possível controlar o grau de consistência de leitura em um cluster secundário. O nível de consistência de leitura determina quanto o cluster secundário espera antes de cada operação de leitura para garantir que algumas ou todas as alterações sejam replicadas do cluster primário. Você pode ajustar o nível de consistência de leitura para garantir que todas as operações de gravação encaminhadas da sessão estejam visíveis no cluster secundário antes de qualquer consulta subsequente. Você também pode usar essa configuração para garantir que as consultas no cluster secundário sempre vejam as atualizações mais atuais do cluster primário. Isso acontece mesmo para aquelas submetidas por outras sessões ou outros clusters. Para especificar esse tipo de comportamento para a aplicação, escolha o valor apropriado para o parâmetro apg_write_forward.consistency_mode. O parâmetro apg_write_forward.consistency_mode tem efeito somente em clusters secundários nos quais o encaminhamento de gravação está habilitado.

> **Note**
>
> Para o parâmetro `apg_write_forward.consistency_mode`, é possível especificar o valor `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Por padrão, o valor é definido como `SESSION`. Definir o valor como `OFF` desativa o encaminhamento de gravação na sessão.

À medida que você aumenta o nível de consistência, a aplicação passa mais tempo aguardando as alterações serem propagadas entre regiões da AWS. Você pode escolher o equilíbrio entre o tempo de resposta rápido e a garantia de que as alterações feitas em outros locais estejam totalmente disponíveis antes da execução das consultas.

Com cada configuração de modo de consistência disponível, o efeito é o seguinte:

- `SESSION`: todas as consultas em uma região secundária da AWS que usa o encaminhamento de gravação veem os resultados de todas as alterações feitas nessa sessão. As alterações são visíveis independentemente de a transação ser confirmada. Se necessário, a consulta aguardará que os resultados das operações de gravação encaminhadas sejam replicados para a região atual. Ele não aguarda resultados atualizados de operações de gravação realizadas em outras regiões ou em outras sessões da região atual.
- `EVENTUAL`: as consultas em uma região secundária da AWS que usa o encaminhamento de gravação podem ver dados ligeiramente obsoletos devido ao atraso de replicação. Os resultados das operações de gravação na mesma sessão não ficam visíveis até que a operação de gravação seja executada na região primária e replicada para a região atual. A consulta não espera que os resultados atualizados estejam disponíveis. Assim, ela pode recuperar os dados mais antigos ou os dados atualizados, dependendo do tempo das declarações e da quantidade de atraso da replicação.
- `GLOBAL`: uma sessão em uma região secundária da AWS vê as alterações feitas por essa sessão. Também vê todas as alterações confirmadas da região principais da AWS e das outras regiões secundárias da AWS. Cada consulta pode aguardar por um período que varia de acordo com a quantidade de atraso da sessão. A consulta prossegue quando o cluster secundário está atualizado com todos os dados confirmados do cluster primário, a partir do momento em que a consulta foi iniciada.
- `OFF`: o encaminhamento de gravação está desativado na sessão.

Para obter mais informações sobre todos os parâmetros envolvidos no encaminhamento de gravação, consulte Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL.

Modos de acesso a transações com encaminhamento de gravação

Se o modo de acesso à transação estiver definido como somente leitura, o encaminhamento de gravação não será usado. Você poderá definir o modo de acesso como somente leitura e gravação enquanto estiver conectado a um cluster de banco de dados e a uma sessão que tenha o encaminhamento de gravação habilitado.

Para ter mais informações sobre os modos de acesso a transações, consulte SET TRANSACTION.

Executar instruções de várias partes com o encaminhamento de gravação no Aurora PostgreSQL

Uma instrução DML pode consistir em várias partes, como uma instrução `INSERT ... SELECT` ou uma instrução `DELETE ... WHERE`. Nesse caso, a instrução inteira é encaminhada para o cluster primário e executada nele.

Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL

Os parameter groups de cluster do Aurora incluem configurações para o recurso de encaminhamento de gravação. Como são parâmetros de cluster, todas as instâncias de banco de dados em cada cluster têm os mesmos valores para essas variáveis. Detalhes sobre esses parâmetros são resumidos na tabela a seguir, com notas de uso após a tabela.

Nome	Escopo	Type	Valor padrão	Valores válidos
`apg_write_forward.connect_timeout`	Sessão	segundos	30	0–2147483647
`apg_write_forward.consistency_mode`	Sessão	enum	Sessão	`SESSION`, `EVENTUAL`, `GLOBAL`, `OFF`

Nome	Escopo	Type	Valor padrão	Valores válidos
`apg_write_forward.idle_in_transaction_session_timeout`	Sessão	milissegundos	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sessão	milissegundos	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Global	int	25	1–100

O parâmetro `apg_write_forward.max_forwarding_connections_percent` é o limite máximo em conexões de banco de dados que pode ser usado em consultas encaminhadas de leitores. Ele é expresso como uma porcentagem da configuração `max_connections` para a instância de banco de dados de gravação no cluster primário. Por exemplo, se `max_connections` for `800` e `apg_write_forward.max_forwarding_connections_percent` for `10`, o gravador permitirá um máximo de 80 sessões encaminhadas simultaneamente. Essas conexões vêm do mesmo grupo de conexões gerenciado pela configuração `max_connections`. Essa configuração se aplica somente ao cluster primário, quando, no mínimo, um cluster secundário tem o encaminhamento de gravação ativado.

Use as seguintes configurações no cluster secundário:

- `apg_write_forward.consistency_mode`: um parâmetro no nível de sessão que controla o grau de consistência de leitura no cluster secundário. Os valores válidos são `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Por padrão, o valor é definido como `SESSION`. Definir o valor como `OFF` desativa o encaminhamento de gravação na sessão. Para saber mais sobre os níveis de consistência, consulte Isolamento e consistência para o encaminhamento de gravação no Aurora PostgreSQL. Esse parâmetro é relevante somente em instâncias de leitor de clusters secundários que têm o encaminhamento de gravação habilitado e que estão em um banco de dados global Aurora.
- `apg_write_forward.connect_timeout`: o número máximo de segundos que o cluster secundário espera ao estabelecer uma conexão com o cluster primário antes de desistir. Um valor de `0` significa esperar indefinidamente.
- `apg_write_forward.idle_in_transaction_session_timeout`: o número de milissegundos que o cluster primário aguarda a atividade em uma conexão encaminhada de um

cluster secundário que tem uma transação aberta antes de fechá-la. Se a sessão permanecer ociosa além desse período na transação, o Aurora encerrará a sessão. Um valor de `0` desativa o tempo limite.

- `apg_write_forward.idle_session_timeout`: o número de milissegundos que o cluster primário aguarda a atividade em uma conexão encaminhada de um cluster secundário antes de fechá-la. Se a sessão permanecer ociosa além desse período, o Aurora encerrará a sessão. Um valor de `0` desativa o tempo limite.

Métricas do Amazon CloudWatch para o encaminhamento de gravação no Aurora PostgreSQL

As métricas do Amazon CloudWatch a seguir se aplicam ao cluster primário quando você usa o encaminhamento de gravação em um ou mais clusters secundários. Essas métricas são todas medidas na instância de banco de dados de gravador no cluster primário.

Métrica do CloudWatch	Unidades e descrição
`AuroraForwardingWriterDMLThroughput`	Contagem (por segundo). Número de instruções DML encaminhadas processadas a cada segundo por essa instância de banco de dados de gravador.
`AuroraForwardingWriterOpenSessions`	Contagem. Número de sessões abertas nessa instância de banco de dados de gravação que processa consultas encaminhadas.
`AuroraForwardingWriterTotalSessions`	Contagem. Número de sessões encaminhadas na instância de banco de dados de gravação.

As métricas do CloudWatch a seguir se aplicam a cada cluster secundário. Essas métricas são medidas em cada instância de banco de dados de leitor em um cluster secundário com o encaminhamento de gravação habilitado.

Métrica do CloudWatch	Unidade e descrição
`AuroraForwardingReplicaCommitThroughput`	Contagem (por segundo). Número de confirmações em sessões encaminhadas por essa réplica a cada segundo.
`AuroraForwardingReplicaDMLLatency`	Milissegundos. Tempo médio de resposta, em milissegundos, de DMLs encaminhadas na réplica.
`AuroraForwardingReplicaDMLThroughput`	Contagem (por segundo). Número de instruções DML encaminhadas processadas por segundo nessa réplica.
`AuroraForwardingReplicaErrorSessionsLimit`	Contagem. Número de sessões rejeitadas pelo cluster primário porque foi atingido o limite máximo de conexões ou o máximo de conexões de gravação e encaminhamento.
`AuroraForwardingReplicaOpenSessions`	Contagem. O número de sessões que estão usando o encaminhamento de gravação em uma instância de banco.
`AuroraForwardingReplicaReadWaitLatency`	Milissegundos. Tempo médio de espera, em milissegundos, que a réplica aguarda para ser consistente com a LSN do cluster primário. O grau em que a instância de banco de dados de leitura aguarda depende da configuração `apg_write_forward.consistency_mode` . Para obter mais informações sobre essa configuração, consulte the section called “Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL”.

Eventos de espera para encaminhamento de gravação no Aurora PostgreSQL

O Amazon Aurora gera os seguintes eventos de espera quando você usa o encaminhamento de gravação com o Aurora PostgreSQL.

Tópicos

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

O evento `IPC:AuroraWriteForwardConnect` ocorre quando um processo de back-end no cluster de banco de dados secundário está aguardando a abertura de uma conexão com o nó gravador do cluster de banco de dados primário.

Possíveis causas do maior número de esperas

Esse evento aumenta à medida que aumenta o número de tentativas de conexão do nó de leitura de uma região secundária no nó de gravação do cluster de banco de dados primário.

Ações do

Reduza o número de conexões simultâneas de um nó secundário para o nó de gravação da região primária.

IPC:AuroraWriteForwardConsistencyPoint

O evento `IPC:AuroraWriteForwardConsistencyPoint` descreve por quanto tempo uma consulta de um nó no cluster de banco de dados secundário aguardará até que os resultados das operações de gravação encaminhadas sejam replicados para a região atual. Esse evento só será gerado se o parâmetro do nível da sessão `apg_write_forward.consistency_mode` estiver definido como um dos seguintes casos:

- `SESSION`: as consultas em um nó secundário aguardam os resultados de todas as alterações feitas nessa sessão.
- `GLOBAL`: as consultas em um nó secundário aguardam os resultados das alterações feitas por essa sessão, além de todas as alterações confirmadas da região primária e de outras regiões secundárias no cluster global.

Para obter informações sobre as configurações do parâmetro `apg_write_forward.consistency_mode`, consulte the section called “Parâmetros de configuração para o encaminhamento de gravação no Aurora PostgreSQL”.

Possíveis causas do maior número de esperas

As causas comuns de tempos de espera mais longos incluem o seguinte:

- Aumento do atraso na réplica, conforme medido pela métrica do Amazon CloudWatch `ReplicaLag`. Para obter mais informações sobre essa métrica, consulte Monitorar a replicação do Aurora PostgreSQL.
- Aumento da carga no nó de gravação da região primária ou no nó secundário.

Ações do

Altere seu modo de consistência, dependendo dos requisitos da aplicação.

IPC:AuroraWriteForwardExecute

O evento `IPC:AuroraWriteForwardExecute` ocorre quando um processo de back-end no cluster de banco de dados secundário está aguardando a conclusão de uma consulta encaminhada e a obtenção dos resultados no nó de gravação do cluster de banco de dados primário.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- Buscar um grande número de linhas no nó de gravação da região primária.
- O aumento da latência da rede entre o nó secundário e o nó de gravação da região primária aumenta o tempo necessário para o nó secundário receber dados do nó de gravação.
- O aumento da carga no nó secundário pode atrasar a transmissão da solicitação de consulta do nó secundário no nó de gravação da região primária.

- O aumento da carga no nó de gravação da região primária pode atrasar a transmissão dos dados do nó de gravação para o nó secundário.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Otimize as consultas para recuperar somente os dados necessários.
- Otimize as operações de linguagem de manipulação de dados (DML) para modificar somente os dados necessários.
- Se o nó secundário ou o nó de gravação da região primária estiver limitado pela CPU ou pela largura de banda da rede, considere alterá-lo para um tipo de instância com mais capacidade de CPU ou mais largura de banda de rede.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

O evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` ocorre quando um processo de back-end no cluster de banco de dados secundário que está usando o modo de consistência GLOBAL está esperando para obter o ponto de consistência global do nó de gravação antes de executar uma consulta.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre o nó secundário e o nó de gravação da região primária aumenta o tempo necessário para o nó secundário receber dados do nó de gravação.
- O aumento da carga no nó secundário pode atrasar a transmissão da solicitação de consulta do nó secundário no nó de gravação da região primária.
- O aumento da carga no nó de gravação da região primária pode atrasar a transmissão dos dados do nó de gravação para o nó secundário.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Altere seu modo de consistência, dependendo dos requisitos da aplicação.

- Se o nó secundário ou o nó de gravação da região primária estiver limitado pela CPU ou pela largura de banda da rede, considere alterá-lo para um tipo de instância com mais capacidade de CPU ou mais largura de banda de rede.

IPC:AuroraWriteForwardXactAbort

O evento `IPC:AuroraWriteForwardXactAbort` ocorre quando um processo de back-end no cluster de banco de dados secundário está aguardando o resultado de uma consulta de limpeza remota. As consultas de limpeza são emitidas para retornar o processo ao estado apropriado após a interrupção de uma transação por gravação. O Amazon Aurora as executa porque um erro foi encontrado ou porque um usuário emitiu um comando `ABORT` explícito ou cancelou uma consulta em execução.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre o nó secundário e o nó de gravação da região primária aumenta o tempo necessário para o nó secundário receber dados do nó de gravação.
- O aumento da carga no nó secundário pode atrasar a transmissão da solicitação de consulta de limpeza do nó secundário no nó de gravação da região primária.
- O aumento da carga no nó de gravação da região primária pode atrasar a transmissão dos dados do nó de gravação para o nó secundário.

Ações do

Recomenda-se ações distintas, dependendo dos motivos do evento de espera.

- Investigue a causa da transação abortada.
- Se o nó secundário ou o nó de gravação da região primária estiver limitado pela CPU ou pela largura de banda da rede, considere alterá-lo para um tipo de instância com mais capacidade de CPU ou mais largura de banda de rede.

IPC:AuroraWriteForwardXactCommit

O evento `IPC:AuroraWriteForwardXactCommit` ocorre quando um processo de back-end no cluster de banco de dados secundário está aguardando o resultado de um comando de confirmação de transação encaminhada.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre o nó secundário e o nó de gravação da região primária aumenta o tempo necessário para o nó secundário receber dados do nó de gravação.
- O aumento da carga no nó secundário pode atrasar a transmissão da solicitação de consulta do nó secundário no nó de gravação da região primária.
- O aumento da carga no nó de gravação da região primária pode atrasar a transmissão dos dados do nó de gravação para o nó secundário.

Ações do

Se o nó secundário ou o nó de gravação da região primária estiver limitado pela CPU ou pela largura de banda da rede, considere alterá-lo para um tipo de instância com mais capacidade de CPU ou mais largura de banda de rede.

IPC:AuroraWriteForwardXactStart

O evento `IPC:AuroraWriteForwardXactStart` ocorre quando um processo de back-end no cluster de banco de dados secundário está aguardando o resultado de um comando de início de transação encaminhada.

Possíveis causas do maior número de esperas

As causas típicas incluem:

- O aumento da latência da rede entre o nó secundário e o nó de gravação da região primária aumenta o tempo necessário para o nó secundário receber dados do nó de gravação.
- O aumento da carga no nó secundário pode atrasar a transmissão da solicitação de consulta do nó secundário no nó de gravação da região primária.
- O aumento da carga no nó de gravação da região primária pode atrasar a transmissão dos dados do nó de gravação para o nó secundário.

Ações do

Se o nó secundário ou o nó de gravação da região primária estiver limitado pela CPU ou pela largura de banda da rede, considere alterá-lo para um tipo de instância com mais capacidade de CPU ou mais largura de banda de rede.

Usar a transição ou o failover no Amazon Aurora Global Database

O recurso Aurora Global Database oferece maior proteção à continuidade dos negócios e recuperação de desastres (BCDR) do que a alta disponibilidade padrão oferecida por um cluster de banco de dados do Aurora em uma única Região da AWS. Quando o Aurora Global Database é utilizado, é possível planejar a recuperação mais rápida de desastres regionais raros ou concluir depressa as interrupções de serviço.

Consulte as diretrizes e os procedimentos a seguir para planejar, testar e implementar sua estratégia de BCDR utilizando o recurso Aurora Global Database.

Tópicos

- Planejar a continuidade dos negócios e a recuperação de desastres
- Realizar transições para o Amazon Aurora Global Database
- Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada
- Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–
- Resiliência entre regiões para clusters secundários do Global Database

Planejar a continuidade dos negócios e a recuperação de desastres

Para planejar sua estratégia de continuidade de negócios e recuperação de desastres, é útil entender a terminologia do setor a seguir e como esses termos se relacionam aos recursos do Aurora Global Database.

A recuperação de desastres geralmente é motivada pelos dois objetivos de negócios a seguir:

- Objetivo de tempo de recuperação (RTO): o tempo que um sistema leva para retornar a um estado funcional após um desastre. Em outras palavras, o RTO mede o tempo de inatividade. Com relação ao Aurora Global Database, o RTO pode estar na ordem dos minutos.
- Objetivo de ponto de recuperação (RPO): a quantidade de dados que podem ser perdidos (medidos no tempo) após um desastre ou interrupção de serviço. Essa perda de dados costuma decorrer de um atraso na replicação assíncrona. Para um banco de dados globalAurora, o RPO é normalmente medido em segundos. Com um banco de dados global Aurora PostgreSQL–baseado, você pode usar o parâmetro `rds.global_db_rpo` para definir e rastrear o limite superior no RPO, mas isso pode afetar o processamento de transações no nó do gravador do cluster primário. Para obter mais informações, consulte Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–.

A realização de uma transição ou um failover com o Aurora Global Database envolve promover um cluster de banco de dados secundário a cluster de banco de dados primário. O termo “interrupção regional” é frequentemente usado para descrever uma variedade de cenários de falha. O pior cenário pode ser uma interrupção generalizada causada por um evento catastrófico que afeta centenas de quilômetros quadrados. No entanto, a maioria das interrupções é muito mais localizada, afetando apenas um pequeno subconjunto de serviços de nuvem ou sistemas de clientes. Considere o escopo completo da interrupção para garantir que o failover entre regiões seja a solução adequada e para escolher o método de failover apropriado para a situação. A opção pela abordagem de failover ou de transição depende do cenário específico de interrupção:

- Failover: use essa abordagem para se recuperar de uma interrupção não planejada. Com ela, você executa um failover entre regiões para um dos clusters de banco de dados secundários no banco de dados global do Aurora. O RPO para essa abordagem geralmente é um valor diferente de zero medido em segundos. A quantidade de perda de dados depende do atraso de replicação do banco de dados global do Aurora em todas as Regiões da AWS no momento da falha. Para saber mais, consulte Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada.
- Transição: anteriormente, essa operação era chamada de “failover planejado gerenciado”. Use essa abordagem para ambientes controlados, como manutenção operacional e outros procedimentos operacionais planejados em que todos os clusters do Aurora e outros serviços com os quais eles interagem estão íntegros. Como esse recurso sincroniza clusters de banco de dados secundários com o primário antes de fazer outras alterações, o RPO é 0 (sem perda de dados). Para saber mais, consulte Realizar transições para o Amazon Aurora Global Database.

> **Note**
>
> Para que seja possível realizar a transição ou o failover para um cluster de banco de dados secundário do Aurora sem periféricos/sem interface gráfica, será necessário adicionar uma instância de banco de dados a ele. Para obter mais informações sobre clusters de banco de dados sem cabeçalho, consulte Criando um cluster de banco de dados Aurora dedicado em uma região secundária.

Realizar transições para o Amazon Aurora Global Database

> **Note**
>
> Anteriormente, as transições eram chamadas de failovers planejados gerenciados.

Ao usar transições, é possível alterar a região do cluster primário rotineiramente. Essa abordagem destina-se a ambientes controlados, como manutenção operacional e outros procedimentos operacionais planejados.

Há três casos de uso comuns para o uso de transições.

- Para requisitos de “alternância regional” impostos a setores específicos. Por exemplo, os regulamentos de serviços financeiros podem querer que os sistemas de nível 0 mudem para uma região diferente por vários meses a fim de garantir que os procedimentos de recuperação de desastres sejam executados regularmente.
- Para aplicações multirregionais ininterruptas (dia e noite). Por exemplo, uma empresa pode querer fornecer gravações de menor latência em diferentes regiões com base no horário comercial em diferentes fusos horários.
- Como um método sem perda de dados para fazer failback para a região primária original após um failover.

> **Note**
>
> As transições foram projetadas para serem usadas em um Aurora Global Database, no qual todos os clusters do Aurora e outros serviços com os quais eles interagem estão íntegros. Para se recuperar de uma interrupção não planejada, siga o procedimento apropriado em Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada.
> Só é possível realizar transições gerenciadas entre regiões com o Aurora Global Database caso os clusters de banco de dados primário e secundário tenham as mesmas versões principal e secundária do mecanismo. Dependendo do mecanismo e de suas respectivas versões, os níveis de patch podem precisar ser idênticos ou podem ser diferentes.
> Para conferir uma lista de mecanismos e suas respectivas versões que permitem essas operações entre clusters primários e secundários com diferentes níveis de patch, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões.
> Antes de iniciar a transição, verifique as versões do mecanismo no cluster global para

garantir que elas sejam compatíveis com a transição gerenciada entre regiões e atualize-as se necessário.

Durante uma transição, o Aurora promove o cluster na região secundária escolhida a cluster primário. O mecanismo de transição mantém a topologia de replicação existente do banco de dados global: ele ainda tem o mesmo número de clusters do Aurora nas mesmas regiões. Antes de iniciar o processo de transição, o Aurora espera que os clusters de destino da região secundária estejam totalmente sincronizados com o cluster da região principal. Depois, o cluster de banco de dados na região primária se torna somente leitura. O cluster secundário escolhido promove um de seus nós somente leitura ao status de gravador completo, permitindo assim que o cluster secundário assuma o papel do cluster primário. Como o cluster de destino secundário foi sincronizado com o primário no início do processo, o novo primário continua as operações para o banco de dados global do Aurora sem perder dados. Seu banco de dados fica indisponível por um curto período enquanto os clusters primários e secundários selecionados estão assumindo suas novas funções.

> **Note**
>
> Para gerenciar slots de replicação para o Aurora PostgreSQL depois de realizar uma transição, consulte Gerenciar slots lógicos para o Aurora PostgreSQL.

Para otimizar a disponibilidade do aplicativo, recomendamos que você faça o seguinte antes de usar esse recurso:

- Execute essa operação em períodos de pouca demanda ou em outro momento quando as gravações no cluster de banco de dados primário forem mínimas.
- Verifique os tempos de atraso para ver se há todos os clusters de banco de dados Aurora secundários no banco de dados global Aurora. Para todos os bancos de dados globais baseados no Aurora PostgreSQL e para bancos de dados globais baseados no Aurora MySQL a partir das versões 3.04.0 e posteriores do mecanismo, ou 2.12.0 e posteriores, use o Amazon CloudWatch para visualizar a métrica `AuroraGlobalDBRPOLag` para todos os clusters de banco de dados secundários. Para versões secundárias anteriores dos bancos de dados globais baseados no Aurora MySQL, veja a métrica `AuroraGlobalDBReplicationLag`. Essa métrica informa a distância (em milissegundos) de um secundário em relação ao cluster de banco de dados primário. Esse valor é diretamente proporcional ao tempo que o Aurora levará para concluir a transição.

Portanto, quanto maior o valor do atraso, mais tempo a transição levará. Examine essas métricas por meio do cluster primário atual.

Para ter mais informações sobre métricas CloudWatch de Aurora, consulte Métricas no nível do cluster do Amazon Aurora.

- O cluster de banco de dados secundário que é promovido durante uma transição pode ter configurações diferentes do antigo cluster de banco de dados primário. Recomendamos manter os tipos de configuração a seguir consistentes em todos os clusters do Aurora Global Database. Isso ajuda a minimizar os problemas de desempenho, as incompatibilidades de workload e outros comportamentos anômalos após uma transição.
  - Configure o grupo de parâmetros do cluster de banco de dados do Aurora como o novo primário, se necessário: quando um cluster de banco de dados secundário é promovido a primário, o grupo de parâmetros do secundário pode ser configurado diferentemente do primário. Em caso afirmativo, modifique o parameter group do cluster de banco de dados secundário promovido para estar em conformidade com as configurações do cluster principal. Para saber como, consulte Modificando parâmetros para um banco de dados do Aurora global.
  - Configurar ferramentas e opções de monitoramento, como Amazon CloudWatch Events e alarmes – Configurar o cluster de banco de dados promovido com a mesma capacidade de registro, alarmes e assim por diante, conforme necessário para o banco de dados global. Tal como ocorre com os grupos de parâmetros, a configuração desses recursos não é herdada do primário no processo de transição. Algumas métricas do CloudWatch, como atraso na replicação, só estão disponíveis para regiões secundárias. Assim, uma transição altera a forma de visualizar essas métricas e definir alarmes para elas, e pode exigir alterações em qualquer painel predefinido. Para ter mais informações sobre o monitoramento e os clusters de banco de dados do Aurora, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.
  - Configurar integrações com outros serviços da AWS: se o banco de dados global do Aurora se integrar a serviços da AWS, como o AWS Secrets Manager, o AWS Identity and Access Management, o Amazon S3 e o AWS Lambda, configure as integrações com esses serviços conforme necessário. Para ter mais informações sobre como integrar Aurora Global Databases com IAM, Amazon S3 e Lambda, consulte Usar bancos de dados globais do Amazon Aurora com outros produtos da AWS. Para saber mais sobre o Secrets Manager, consulte Como automizar a replicação de segredos no AWS Secrets Manager entre Regiões da AWS.

Se você estiver usando o endpoint do gravador do Aurora Global Database, não precisará alterar as configurações de conexão na aplicação. Verifique se as alterações de DNS foram propagadas e se

você pode se conectar e realizar operações de gravação no novo cluster primário. Depois, é possível retomar a operação completa da aplicação.

Suponha que as conexões da aplicação usem o endpoint do cluster primário antigo em vez do endpoint do gravador global. Nesse caso, altere as configurações de conexão da aplicação para usar o endpoint do novo cluster primário. Se você aceitou os nomes fornecidos ao criar o banco de dados global Aurora, poderá alterar o endpoint removendo `-ro` da string de endpoint do cluster promovido em seu aplicativo. Por exemplo, o endpoint do cluster secundário `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` torna-se `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando esse cluster é promovido para primário.

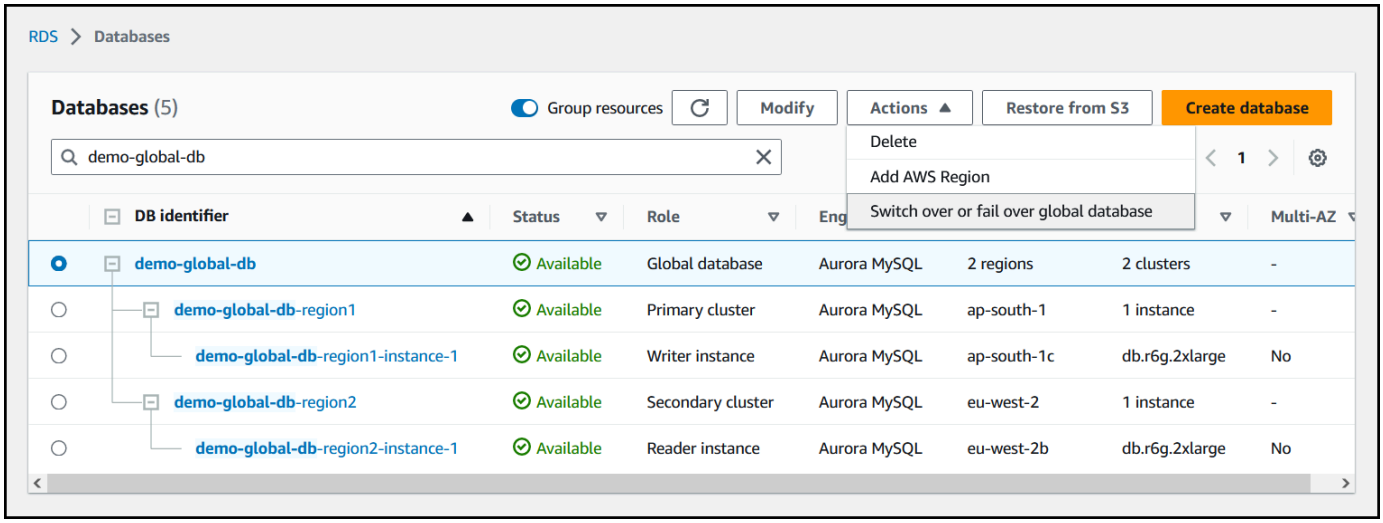
Se você estiver usando o RDS Proxy, redirecione as operações de gravação da aplicação para o endpoint de leitura/gravação apropriado do proxy associado ao novo cluster primário. Esse endpoint do proxy pode ser o endpoint padrão ou um endpoint de leitura/gravação personalizado. Para obter mais informações, consulte Como os endpoints do RDS Proxy funcionam com bancos de dados globais.

É possível realizar a transição do Aurora Global Database usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

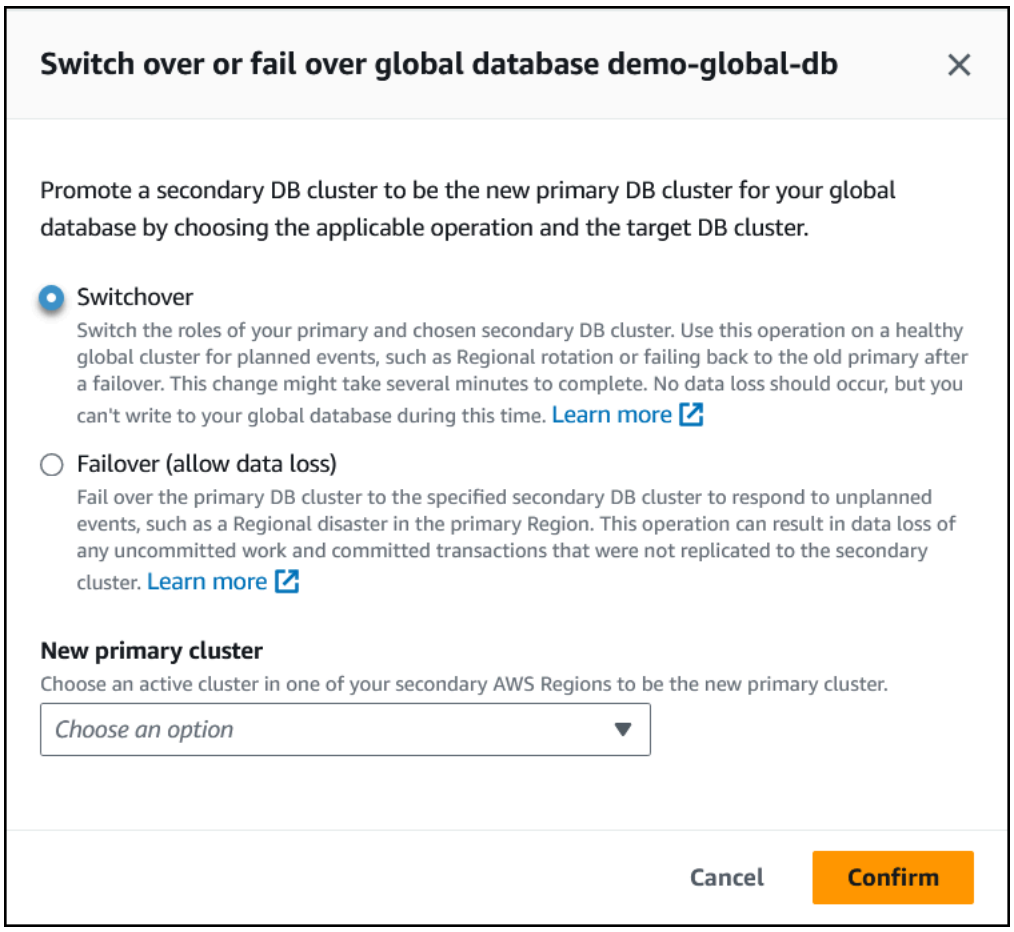
Console

Como realizar a transição no banco de dados global do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Bancos de dados e localize o Aurora Global Database no qual você deseja realizar a transição.
3. Selecione Fazer troca ou failover do banco de dados global no menu Ações.

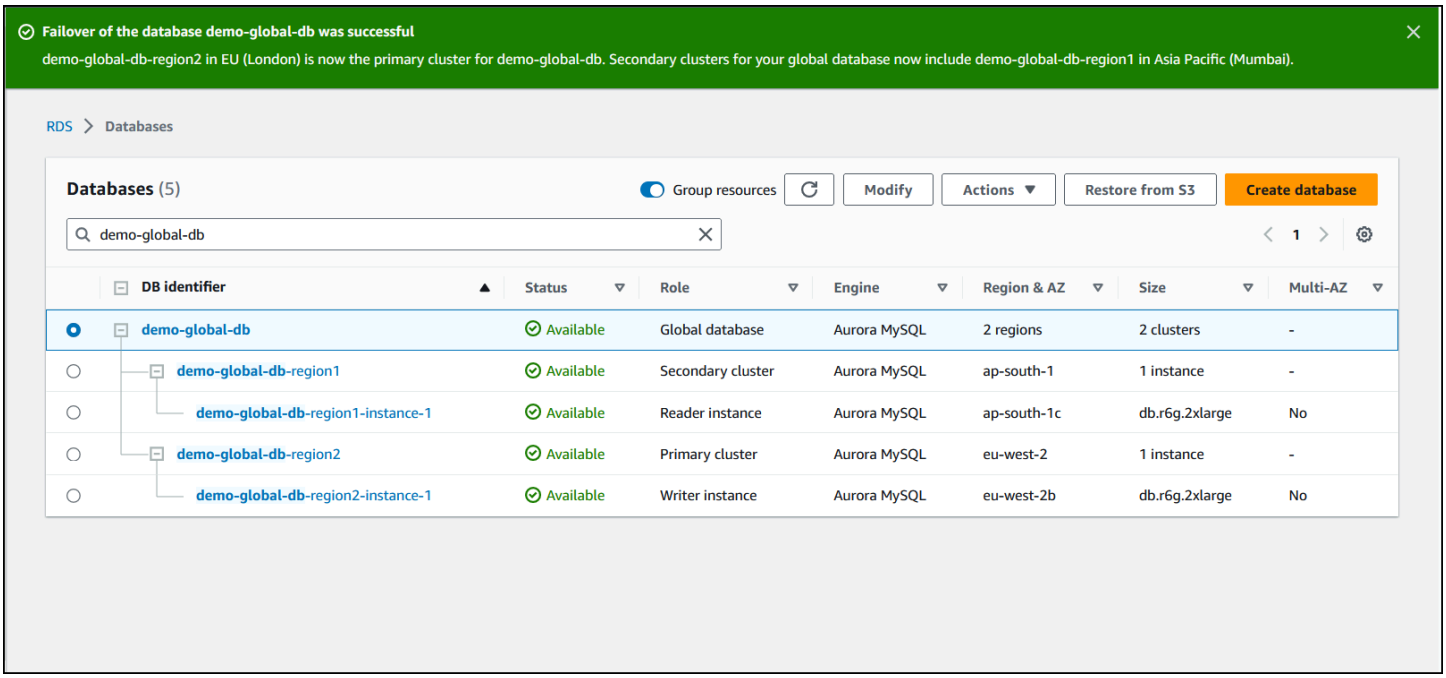


4. Escolha Troca.



5. Em Novo cluster primário, escolha um cluster ativo em uma das Regiões da AWS secundárias para ser o novo cluster primário.
6. Escolha Confirmar.

Quando a transição for concluída, você verá os clusters de banco de dados do Aurora e as funções atuais deles na lista Bancos de dados, como mostrado na imagem a seguir.



AWS CLI

Como realizar a transição em um banco de dados global do Aurora

Use o comando `switchover-global-cluster` da CLI para realizar a transição do Aurora Global Database. Com o comando, passe valores para os seguintes parâmetros.

- `--region`: especifique a Região da AWS em que o cluster de banco de dados primário do Aurora Global Database está sendo executado.
- `--global-cluster-identifier` – Especifique o nome do banco de dados globalAurora.
- `--target-db-cluster-identifier` – Especifique o Nome do recursos da Amazon (ARN) do cluster de banco de dados Aurora que você deseja promover para ser o principal para o banco de dados global Aurora.

Para Linux, macOS ou Unix:

```
aws rds --region region_of_primary \
   switchover-global-cluster --global-cluster-identifier global_database_id \
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

Para Windows:

```
aws rds --region region_of_primary ^
   switchover-global-cluster --global-cluster-identifier global_database_id ^
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

API do RDS

Para realizar a transição do Aurora Global Database, realize a operação de API SwitchoverGlobalCluster.

Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada

Em ocasiões raras, o Aurora Global Database pode sofrer uma interrupção inesperada na Região da AWS primária. Se isso acontecer, o cluster de banco de dados primário do Aurora e o respectivo nó de gravador não estarão disponíveis, e a replicação entre o cluster primário e os secundários cessará. Para minimizar o tempo de inatividade (RTO) e a perda de dados (RPO), você pode trabalhar rapidamente para executar um failover entre regiões.

O Aurora Global Database tem dois métodos de failover que você pode usar em uma situação de recuperação de desastres:

- Failover gerenciado: esse método é recomendado para recuperação de desastres. Ao usá-lo, o Aurora adiciona automaticamente a antiga região primária ao banco de dados global como uma região secundária quando ela se torna disponível novamente. Assim, a topologia original do cluster global é mantida. Para saber como usar esse método, consulte Failovers planejados gerenciados para o Aurora Global Database.
- Failover manual: esse método alternativo pode ser usado quando o failover gerenciado não for uma opção; por exemplo, quando as regiões primária e secundária estiverem executando versões de mecanismo incompatíveis. Para saber como usar esse método, consulte Realizar failovers manuais para bancos de dados globais do Aurora.

> **Important**
>
> Ambos os métodos de failover podem resultar na perda dos dados da transação de gravação que não foram replicados para o secundário escolhido antes da ocorrência do evento de failover. No entanto, o processo de recuperação que promove uma instância de banco de dados no cluster de banco de dados secundário escolhido para ser a instância de banco de

dados principal do gravador garante que os dados estejam em um estado transacionalmente consistente. Os failovers também são suscetíveis a problemas de cérebro dividido.

Failovers planejados gerenciados para o Aurora Global Database

Essa abordagem destina-se à continuidade dos negócios no caso de um desastre regional real ou de uma interrupção completa do nível de serviço.

Durante um failover gerenciado, o cluster secundário na região secundária escolhida se torna o novo cluster primário. O cluster secundário escolhido promove um dos nós somente leitura ao status de gravador completo. Essa etapa permite que o cluster assuma a função de cluster primário. O banco de dados ficará indisponível por um curto período enquanto o cluster estiver assumindo a nova função. Assim que essa região primária antiga estiver íntegra e disponível novamente, o Aurora a adicionará automaticamente ao cluster global como uma região secundária. Assim, a topologia de replicação existente do Aurora Global Database é mantida.

> **Note**
>
> Para gerenciar slots de replicação para o Aurora PostgreSQL após realizar um failover, consulte Gerenciar slots lógicos para o Aurora PostgreSQL.

> **Note**
>
> Só é possível realizar failovers gerenciados entre regiões com o Aurora Global Database caso os clusters de banco de dados primário e secundário tenham as mesmas versões principal e secundária do mecanismo. Dependendo do mecanismo e de suas respectivas versões, os níveis de patch podem precisar ser idênticos ou podem ser diferentes.
> Para conferir uma lista de mecanismos e suas respectivas versões que permitem essas operações entre clusters primários e secundários com diferentes níveis de patch, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões.
> Antes de iniciar o failover, confira as versões do mecanismo no cluster global para garantir que elas sejam compatíveis com a transição gerenciada entre regiões e atualize-as, se necessário. Se as versões do mecanismo exigirem níveis de patch idênticos, mas estiverem executando níveis de patch diferentes, será possível realizar o failover manualmente seguindo as etapas em Realizar failovers manuais para bancos de dados globais do Aurora.

O failover gerenciado não aguarda até os dados serem sincronizados entre a região secundária escolhida e a região primária atual. Como o Aurora Global Database replica dados de forma assíncrona, é possível que nem todas as transações sejam replicadas para a região secundária da AWS escolhida antes de ser promovida para aceitar recursos completos de leitura/gravação.

Para garantir que os dados estejam em um estado consistente, o Aurora cria um volume de armazenamento para a antiga região primária após a recuperação. Antes de criar o volume de armazenamento na região da AWS, o Aurora tenta gerar um snapshot do volume de armazenamento antigo no ponto da falha. Dessa forma, é possível restaurar o snapshot e recuperar todos os dados perdidos por meio dele. Se essa operação for bem-sucedida, o Aurora colocará esse snapshot denominado `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* na seção de snapshots do Console de gerenciamento da AWS. Também é possível usar o comando `describe-db-cluster-snapshots` da AWS CLI ou a operação de API `DescribeDBClusterSnapshots` para ver os detalhes do snapshot.

Quando um failover gerenciado é iniciado, o Aurora também tenta interromper o tráfego de gravação por meio da camada de armazenamento altamente disponível do Aurora. Chamamos esse mecanismo de “delimitação de gravação”. Se o processo for bem-sucedido, o Aurora emitirá um evento do RDS informando que as gravações foram interrompidas. No caso improvável de várias falhas de AZ em uma região, é possível que o processo de delimitação de gravação não seja bem-sucedido em tempo hábil. Nesse caso, o Aurora emite um evento do RDS informando que o processo para interromper as gravações atingiu o tempo limite. Se o antigo cluster primário estiver acessível na rede, o Aurora registrará esses eventos nele. Caso contrário, o Aurora registrará os eventos no novo cluster primário. Para saber mais sobre esses eventos, consulte Eventos de cluster de banco de dados. Como limitar a gravação é a melhor tentativa, é possível que as gravações sejam aceitas momentaneamente na antiga região primária, causando problemas de cérebro dividido.

Recomendamos que você conclua as tarefas a seguir antes de realizar um failover com o Aurora Global Database. Isso minimiza a possibilidade de problemas de cérebro dividido ou de recuperação de dados não replicados do snapshot do antigo cluster primário.

- Para evitar que as gravações sejam enviadas ao cluster primário do Aurora Global Database, desative as aplicações.
- Todas as aplicações que se conectam ao cluster de banco de dados primário devem usar o endpoint do gravador global. Esse endpoint tem um valor que permanece o mesmo quando uma nova região se torna o cluster primário devido a uma transição ou a um failover. O Aurora implementa proteções adicionais para minimizar a possibilidade de perda de dados nas operações

de gravação enviadas por meio do endpoint global. Para ter mais informações sobre os endpoints do gravador global, consulte Conectar-se ao Amazon Aurora Global Database.

- Se você estiver usando o endpoint do gravador global e os valores de DNS do cache de camadas de rede ou da aplicação, reduza o tempo de vida (TTL) do cache de DNS para um valor baixo, como 5 segundos. Dessa forma, a aplicação registra rapidamente as alterações de DNS no endpoint do gravador global. Embora o Aurora tente bloquear gravações na antiga região primária, não há garantia de êxito da ação. Reduzir a duração do cache de DNS reduz ainda mais a probabilidade de problemas de cérebro dividido. Como alternativa, é possível conferir o evento do RDS que informa quando o Aurora observou as alterações de DNS no endpoint do gravador global. Dessa forma, é possível confirmar se a aplicação também registrou a alteração de DNS antes de reiniciar o tráfego de gravação da aplicação.
- Confira os tempos de atraso de todos os clusters de banco de dados do Aurora secundários no Aurora Global Database. Escolher a região secundária com o menor atraso de replicação pode minimizar a perda de dados na atual região primária com falha.

  Para todas as versões de bancos de dados globais baseados no Aurora PostgreSQL e para bancos de dados globais baseados no Aurora MySQL com versão de mecanismo 3.04.0 e posterior ou 2.12.0 e posterior, use o Amazon CloudWatch para visualizar a métrica `AuroraGlobalDBRPOLag` para todos os clusters de banco de dados secundários. Para versões secundárias anteriores dos bancos de dados globais baseados no Aurora MySQL, veja a métrica `AuroraGlobalDBReplicationLag`. Essa métrica informa a distância (em milissegundos) de um secundário em relação ao cluster de banco de dados primário.

  Para ter mais informações sobre métricas CloudWatch de Aurora, consulte Métricas no nível do cluster do Amazon Aurora.

Durante um failover gerenciado, o cluster de banco de dados secundário escolhido é promovido à sua nova função como primário. No entanto, ele não herda as várias opções de configuração do cluster de banco de dados primário. Uma incompatibilidade na configuração pode levar a problemas de performance, incompatibilidades de workload e outros comportamentos anômalos. Para evitar esses problemas, recomendamos que você resolva as diferenças entre os clusters de banco de dados globais Aurora para o seguinte:

- Configure o grupo de parâmetros do cluster de banco de dados do Aurora para o novo primário, se necessário: é possível configurar os grupos de parâmetros do cluster de banco de dados do Aurora de maneira independente para cada cluster do Aurora no Aurora Global Database. Por isso, quando você promove um cluster de banco de dados secundário para assumir a função

primária, o grupo de parâmetros do secundário pode ser configurado diferentemente do grupo do primário. Em caso afirmativo, modifique o parameter group do cluster de banco de dados secundário promovido para estar em conformidade com as configurações do cluster principal. Para saber como, consulte Modificando parâmetros para um banco de dados do Aurora global.

- Configurar ferramentas e opções de monitoramento, como Amazon CloudWatch Events e alarmes – Configurar o cluster de banco de dados promovido com a mesma capacidade de registro, alarmes e assim por diante, conforme necessário para o banco de dados global. Tal como acontece com os parameter groups, a configuração desses recursos não é herdada do primário no processo de failover. Algumas métricas do CloudWatch, como atraso na replicação, só estão disponíveis para regiões secundárias. Assim, um failover altera a forma de visualizar essas métricas e definir alarmes para elas, e pode exigir alterações em qualquer painel predefinido. Para ter mais informações sobre o monitoramento dos clusters de banco de dados do Aurora, consulte Monitorar métricas do Amazon Aurora com o Amazon CloudWatch.
- Configure integrações com outros serviços da AWS: se o Aurora Global Database se integrar a outros serviços da AWS, como o AWS Secrets Manager, o AWS Identity and Access Management, o Amazon S3 e o AWS Lambda, será necessário garantir que eles estejam configurados, conforme necessário, para acessar de quaisquer regiões secundárias. Para ter mais informações sobre como integrar Aurora Global Databases com IAM, Amazon S3 e Lambda, consulte Usar bancos de dados globais do Amazon Aurora com outros produtos da AWS. Para saber mais sobre o Secrets Manager, consulte Como automizar a replicação de segredos no AWS Secrets Manager entre Regiões da AWS.

Normalmente, o cluster secundário escolhido assume a função principal em alguns minutos. Assim que a instância de banco de dados do gravador da nova região primária estiver disponível, será possível conectar as aplicações a ela e retomar as workloads. Depois que o Aurora promove o novo cluster primário, ele reconstrói automaticamente todos os clusters adicionais da região secundária.

Como os bancos de dados globais do Aurora usam replicação assíncrona, o atraso na replicação em cada região secundária pode variar. O Aurora reconstrói essas regiões secundárias para ter exatamente os mesmos dados de um ponto no tempo que o novo cluster da região primária. A duração da tarefa de reconstrução completa pode levar de alguns minutos a várias horas, dependendo do tamanho do volume de armazenamento e da distância entre as regiões. Quando os clusters da região secundária terminam de ser reconstruídos com base na nova região primária, eles ficam disponíveis para acesso de leitura.

Assim que o novo gravador principal for promovido e disponibilizado, o cluster da nova região primária poderá lidar com operações de leitura e gravação no banco de dados global do Aurora.

Se você estiver usando o endpoint global, não precisará alterar as configurações de conexão na aplicação. Verifique se as alterações de DNS foram propagadas e se você pode se conectar e realizar operações de gravação no novo cluster primário. Depois, é possível retomar a operação completa da aplicação.

Se você não estiver usando o endpoint global, altere o endpoint da aplicação para usar o endpoint do cluster de banco de dados recém-promovido a primário. Se você aceitou os nomes fornecidos ao criar o banco de dados global Aurora, poderá alterar o endpoint removendo `-ro` da string de endpoint do cluster promovido em seu aplicativo.

Por exemplo, o endpoint do cluster secundário `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` torna-se `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando esse cluster é promovido para primário.

Se você estiver usando o RDS Proxy, redirecione as operações de gravação de sua aplicação para o endpoint de leitura/gravação apropriado do proxy associado ao novo cluster primário. Esse endpoint do proxy pode ser o endpoint padrão ou um endpoint de leitura/gravação personalizado. Para obter mais informações, consulte Como os endpoints do RDS Proxy funcionam com bancos de dados globais.

Para restaurar a topologia original do cluster de banco de dados global, o Aurora monitora a disponibilidade da antiga região primária. Assim que essa região estiver íntegra e disponível novamente, o Aurora a adicionará automaticamente ao cluster global como uma região secundária. Antes de criar o novo volume de armazenamento na antiga região primária, o Aurora tenta capturar um snapshot do volume de armazenamento antigo no ponto da falha. Ele faz isso para que você possa usá-lo para recuperar os dados perdidos. Se essa operação for bem-sucedida, o Aurora criará um snapshot denominado `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster-timestamp`*. É possível encontrar esse snapshot na seção Snapshots do Console de gerenciamento da AWS. Você também pode ver esse snapshot listado nas informações retornadas pela operação de API DescribeDBClusterSnapshots.

> **Note**
>
> O snapshot do volume de armazenamento antigo é um snapshot do sistema sujeito ao período de retenção de backup configurado no antigo cluster primário. Para preservar esse instantâneo fora do período de retenção, você pode copiá-lo para salvá-lo como um snapshot manual. Para saber mais sobre como copiar snapshots, incluindo os preços, consulte Cópia de snapshot de cluster de banco de dados.

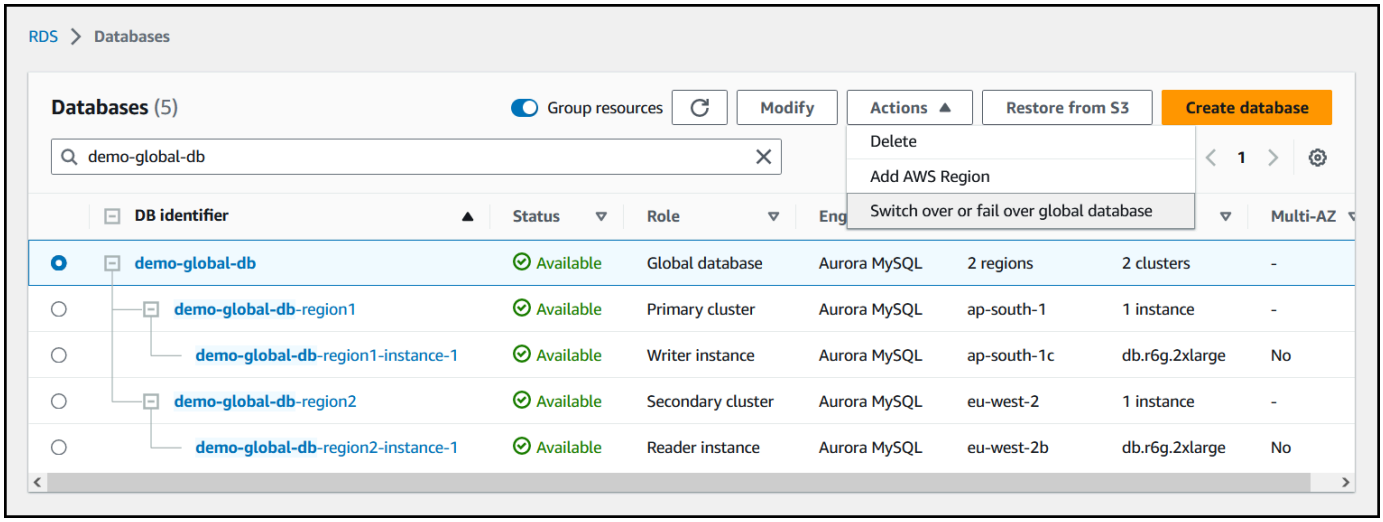
Depois que a topologia original for restaurada, você poderá fazer failback do banco de dados global para a região primária original executando uma operação de transição quando fizer mais sentido para seus negócios e sua workload. Para isso, siga as etapas em Realizar transições para o Amazon Aurora Global Database.

É possível realizar um failover com o Aurora Global Database usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

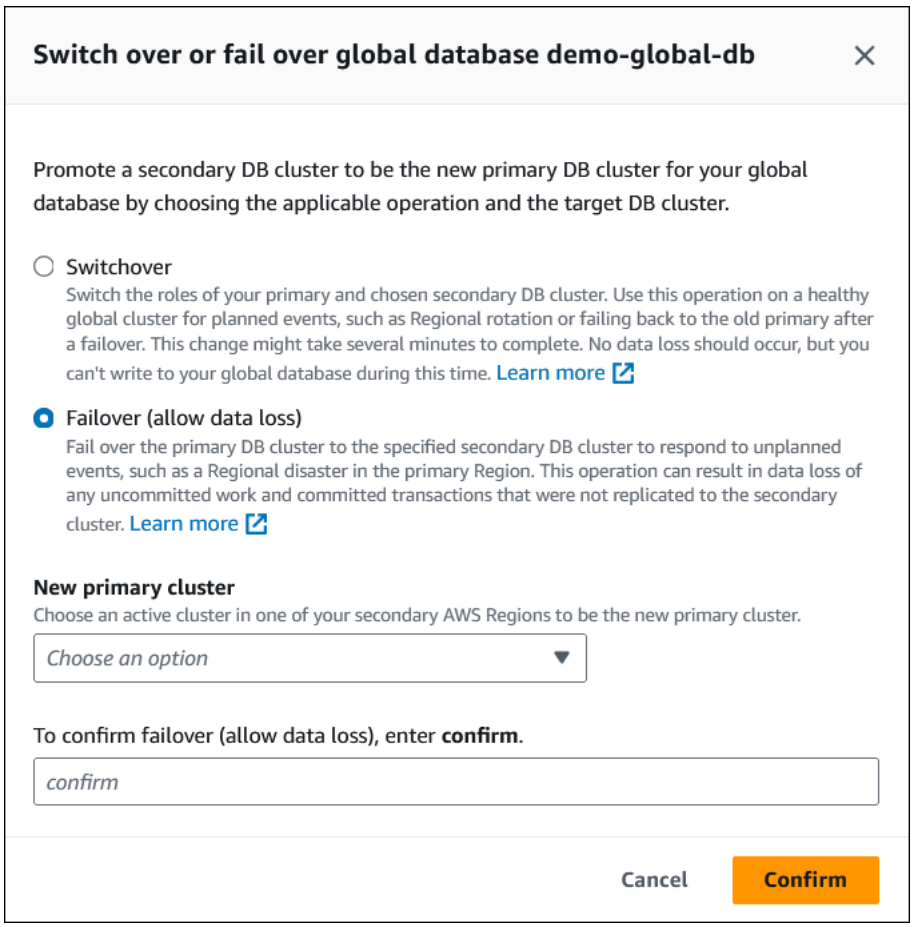
Console

Como iniciar o processo de failover no banco de dados global do Aurora

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Bancos de dados e localize o Aurora Global Database no qual você deseja fazer o failover.
3. Selecione Fazer troca ou failover do banco de dados global no menu Ações.

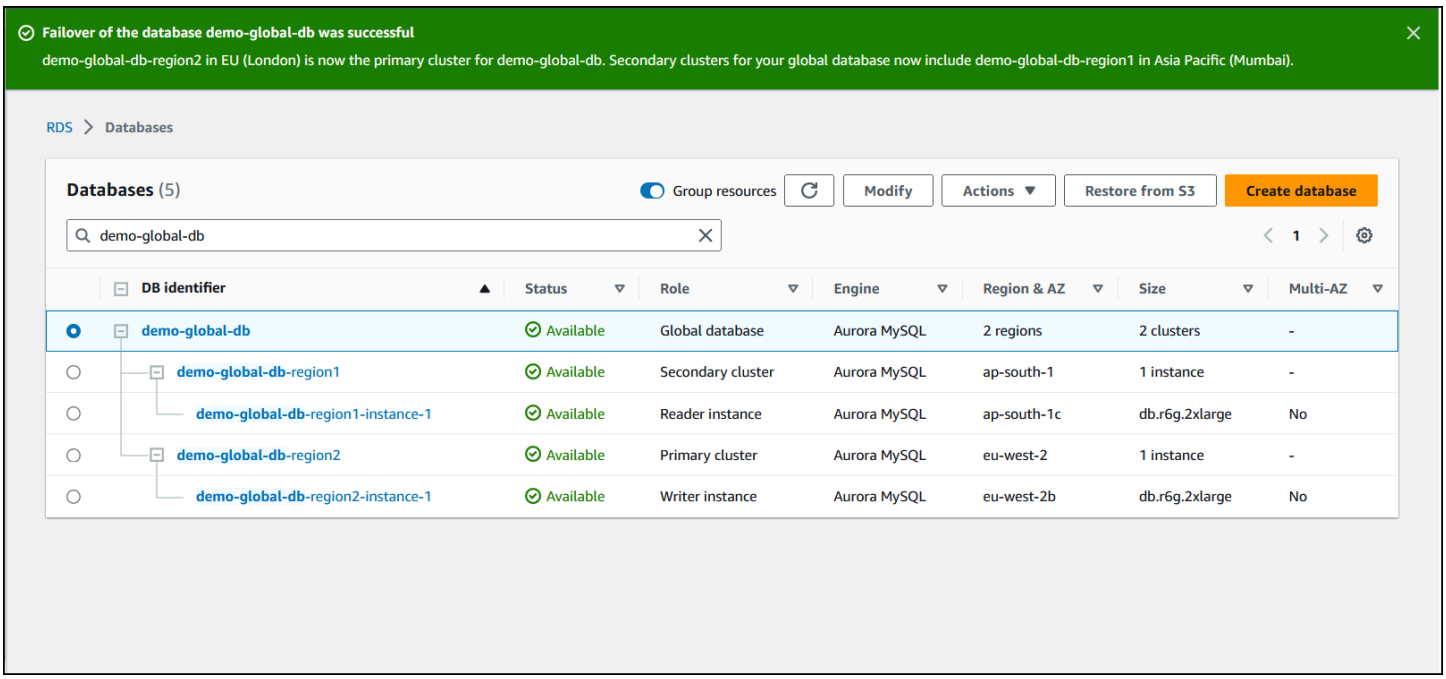


4. Escolha Fazer failover (permitir perda de dados).



5. Em Novo cluster primário, escolha um cluster ativo em uma das Regiões da AWS secundárias para ser o novo cluster primário.
6. Se solicitado, insira **`confirm`** e escolha Confirmar.

Quando o failover for concluído, você verá os clusters de banco de dados do Aurora e o estado atual deles na lista Bancos de dados, como mostrado na imagem a seguir.



AWS CLI

Como iniciar o processo de failover no banco de dados global do Aurora

Use o comando `failover-global-cluster` da CLI para realizar um failover com o Aurora Global Database. Com o comando, passe valores para os seguintes parâmetros.

- `--region`: especifique a Região da AWS onde está sendo executado o cluster de banco de dados secundário que você quer que seja o novo primário para o banco de dados global do Aurora.
- `--global-cluster-identifier` – Especifique o nome do banco de dados globalAurora.
- `--target-db-cluster-identifier`: especifique o nome do recurso da Amazon (ARN) do cluster de banco de dados do Aurora que você deseja promover para que seja o novo principal para o banco de dados global do Aurora.
- `--allow-data-loss`: torne isso explicitamente uma operação de failover em vez de uma operação de transição. Uma operação de failover pode resultar em alguma perda de dados se os componentes de replicação assíncrona não tiverem concluído o envio de todos os dados replicados para a região secundária.

Para Linux, macOS ou Unix:

```
aws rds --region region_of_selected_secondary \
```

```
   failover-global-cluster --global-cluster-identifier global_database_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Para Windows:

```
aws rds --region region_of_selected_secondary ^
   failover-global-cluster --global-cluster-identifier global_database_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

API do RDS

Para realizar um failover com o Aurora Global Database, execute a operação de API FailoverGlobalCluster.

Realizar failovers manuais para bancos de dados globais do Aurora

Em alguns ambientes, você pode não conseguir usar o processo de failover gerenciado. Um exemplo é se os clusters de banco de dados primários e secundários não estiverem executando versões de mecanismo compatíveis. Nesse caso, é possível seguir esse processo manual para realizar um failover para a região secundária de destino.

> **Tip**
>
> Recomendamos que você entenda esse processo antes de usá-lo. Tenha um plano pronto para prosseguir rapidamente ao primeiro sinal de uma questão em toda a região. Você pode se preparar para identificar a região secundária com o menor atraso de replicação usando o Amazon CloudWatch regularmente para rastrear os tempos de atraso dos clusters secundários. Procure testar seu plano para verificar se os procedimentos estão completos e precisos, bem como se a equipe está treinada para executar um failover de recuperação de desastres antes que isso realmente aconteça.

Como realizar um failover manual para um cluster secundário após uma interrupção não planejada na região primária

1. Interrompa a emissão de instruções de DML e outras operações de gravação para o cluster do banco de dados Aurora principal na Região da AWS com a interrupção.

2. Identifique um cluster de banco de dados Aurora de uma Região da AWS secundária para usar como um novo cluster de banco de dados primário. Se você tiver duas (ou mais) Regiões da AWS secundárias no banco de dados global do Aurora, escolha o cluster secundário que tem o menor atraso de replicação.
3. Desanexe o cluster de banco de dados secundário escolhido do banco de dados global Aurora.

   A remoção de um cluster de banco de dados secundário de um banco de dados global Aurora interrompe imediatamente a replicação do primário para este secundário e promove-o ao cluster de banco de dados Aurora provisionado autônomo com recursos completos de leitura/gravação. Quaisquer outros clusters de banco de dados Aurora secundários associados ao cluster primário na região com a interrupção ainda estão disponíveis e podem aceitar chamadas de seu aplicativo. Eles também consomem recursos. Como você está recriando o Aurora Global Database, remova os outros clusters de banco de dados secundários antes de criar o Aurora Global Database nas etapas a seguir. Isso evita inconsistências de dados entre os clusters de banco de dados no Aurora Global Database (problemas de cérebro dividido).

   Para obter as etapas detalhadas para desanexar, consulte Remover um cluster de um banco de dados global do Amazon Aurora.
4. Reconfigure seu aplicativo para enviar todas as operações de gravação para esse cluster de banco de dados Aurora autônomo agora usando seu novo endpoint. Se você aceitou os nomes fornecidos ao criar o banco de dados global Aurora, poderá alterar o endpoint removendo a `-ro` string do endpoint do cluster em seu aplicativo.

   Por exemplo, o endpoint do cluster secundário `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` se torna `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando esse cluster é separado do banco de dados global Aurora.

   Esse cluster de banco de dados Aurora se torna o cluster principal de um novo Aurora Global Database quando você começa a adicionar regiões a ele, na próxima etapa.

   Se você estiver usando o RDS Proxy, redirecione as operações de gravação de sua aplicação para o endpoint de leitura/gravação apropriado do proxy associado ao novo cluster primário. Esse endpoint do proxy pode ser o endpoint padrão ou um endpoint de leitura/gravação personalizado. Para ter mais informações, consulte Como os endpoints do RDS Proxy funcionam com bancos de dados globais.
5. Adicione uma Região da AWS ao cluster de banco de dados. Quando você faz isso, o processo de replicação de primário para secundário começa. Para obter as etapas detalhadas para

adicionar uma região, consulte Adicionar uma Região da AWS a um Amazon Aurora Global Database.

6. Adicione mais Regiões da AWS conforme necessário para recriar a topologia necessária para oferecer suporte à aplicação.

Certifique-se de que as gravações de aplicações sejam enviadas para o cluster de banco de dados Aurora correto antes, durante e depois de fazer essas alterações. Isso evita inconsistências de dados entre os clusters de banco de dados no Aurora Global Database (problemas de cérebro dividido).

Se você executou essa reconfiguração em resposta a uma interrupção em uma Região da AWS, você pode tornar a Região da AWS a principal novamente após a interrupção ser resolvida. Para isso, adicione a antiga Região da AWS ao novo banco de dados global e, depois, use o processo de transição para trocar a respectiva função. O banco de dados global do Aurora deve usar uma versão do Aurora PostgreSQL ou Aurora MySQL que comporte transições. Para obter mais informações, consulte Realizar transições para o Amazon Aurora Global Database.

Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–

Com um banco de dados global baseado em Aurora PostgreSQL, você pode gerenciar o objetivo de ponto de recuperação (RPO) para o banco de dados global do Aurora usando o parâmetro `rds.global_db_rpo`. O RPO representa a quantidade máxima de dados que podem ser perdidos em caso de interrupção.

Quando você define um RPO para seu banco de dados global baseado emAurora PostgreSQL–, Aurora monitora o tempo de atraso de RPO de todos os clusters secundários para garantir que pelo menos um cluster secundário permaneça dentro da janela RPO de destino. O tempo de atraso do RPO é outra métrica baseada no tempo.

O RPO é usado quando o banco de dados retoma as operações em uma nova Região da AWS após um failover. O Aurora avalia os tempos de atraso de RPO e RPO para confirmar (ou bloquear) transações no primário da seguinte forma:

- Compromete a transação se pelo menos um cluster de banco de dados secundário tiver um tempo de atraso de RPO menor que o RPO.
- Bloqueia a transação se todos os clusters de banco de dados secundários tiverem tempos de atraso de RPO maiores que o RPO. Ele também registra o evento no arquivo de log PostgreSQL e emite eventos de “espera” que mostram as sessões bloqueadas.

Em outras palavras, se todos os clusters secundários estiverem atrás do RPO de destino, Aurora pausará as transações no cluster primário até que pelo menos um dos clusters secundários seja atingidos. As transações pausadas são retomadas e confirmadas assim que o tempo de atraso de pelo menos um cluster de banco de dados secundário se torna menor que o RPO. O resultado é que nenhuma transação pode confirmar até que o RPO seja atendido.

O parâmetro `rds.global_db_rpo` é dinâmico. Se você decidir que não quer que todas as transações de gravação sejam proteladas até que o atraso diminua o suficiente, você pode redefini-lo rapidamente. Nesse caso, a Aurora reconhece e implementa a alteração após um pequeno atraso.

> **Important**
>
> Em um banco de dados global com apenas duas regiões da AWS, recomendamos manter o valor padrão do parâmetro `rds.global_db_rpo` no grupo de parâmetros da região secundária. Caso contrário, realizar um failover devido à perda da região primária da AWS pode fazer com que o Aurora pause as transações. Em vez disso, antes de alterar esse parâmetro para impor um RPO máximo, espere até que o Aurora conclua a reconstrução do cluster na antiga região da AWS com falha.

Se você definir esse parâmetro como descrito a seguir, também poderá monitorar as métricas geradas. Você pode fazer isso usando `psql` ou outra ferramenta para consultar o cluster do banco de dados primário do banco de dados global Aurora e obter informações detalhadas sobre as operações do banco de dados global baseado em Aurora PostgreSQL–. Para saber como, consulte Monitorar banco de dados globais baseados no Aurora PostgreSQL.

Tópicos

- Configurar o objetivo de ponto de recuperação
- Visualizar o objetivo de ponto de recuperação
- Desabilitar o objetivo de ponto de recuperação

Configurar o objetivo de ponto de recuperação

O parâmetro `rds.global_db_rpo` controla a configuração de RPO para um banco de dados PostgreSQL. Este parâmetro é tem suporte de Aurora PostgreSQL. Valores válidos de intervalo `rds.global_db_rpo` de 20 segundos a 2.147.483.647 segundos (68 anos). Escolha um valor

realista para atender às suas necessidades de negócios e caso de uso. Por exemplo, você pode querer permitir até 10 minutos para o seu RPO, caso em que você define o valor para 600.

Você pode definir esse valor para seu banco de dados global baseado no Aurora PostgreSQL usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como definir o RPO

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha o cluster principal do seu banco de dados global Aurora e abra a guia Configuration (Configuração) para localizar seu parameter group de cluster de banco de dados. Por exemplo, o parameter group padrão para um cluster de banco de dados primário executando Aurora PostgreSQL 11.7 é `default.aurora-postgresql11`.

   Os grupos de parâmetros não podem ser editados diretamente. Em vez disso, você pode fazer o seguinte:

   - Crie um parameter group de cluster de banco de dados personalizado usando o parameter group padrão apropriado como o ponto de partida. Por exemplo, crie um parameter group de cluster de banco de dados personalizado com base no `default.aurora-postgresql11`.
   - Em seu parameter group de banco de dados personalizado, defina o valor do parâmetro rds.global_db_rpo para atender ao seu caso de uso. Os valores válidos são de 20 segundos até o valor inteiro máximo de 2.147.483.647 (68 anos).
   - Aplique o parameter group de cluster de banco de dados modificado ao cluster de Aurora banco de dados.

Para ter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

AWS CLI

Para definir o parâmetro `rds.global_db_rpo`, use o comando CLI modify-db-cluster-parameter-group. No comando, especifique o nome do grupo de parâmetros do cluster primário e os valores para o parâmetro RPO.

O exemplo a seguir define o RPO como 600 segundos (10 minutos) para o grupo de parâmetros de cluster de banco de dados primário chamado `my_custom_global_parameter_group`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

API do RDS

Para modificar o parâmetro `rds.global_db_rpo`, use a operação de API ModifyDBClusterParameterGroup do Amazon RDS.

Visualizar o objetivo de ponto de recuperação

O objetivo de ponto de recuperação (RPO) de um banco de dados global é armazenado no parâmetro `rds.global_db_rpo` de cada cluster de banco de dados. Você pode se conectar ao endpoint do cluster secundário que deseja visualizar e usar `psql` para consultar a instância para esse valor.

```
db-name=>show rds.global_db_rpo;
```

Se este parâmetro não estiver definido, a consulta retornará o seguinte:

```
rds.global_db_rpo
-------------------
 -1
(1 row)
```

Esta próxima resposta é de um cluster de banco de dados secundário que tenha uma configuração de RPO de 1 minuto.

```
rds.global_db_rpo
-------------------
 60
(1 row)
```

Você pode igualmente usar a CLI para obter valores para descobrir se `rds.global_db_rpo` é ativo em alguns dos conjuntos de Aurora banco de dados usando a CLI para obter valores de todos os `user` parâmetros para o cluster.

Para Linux, macOS ou Unix:

```
aws rds describe-db-cluster-parameters \
 --db-cluster-parameter-group-name lab-test-apg-global \
 --source user
```

Para Windows:

```
aws rds describe-db-cluster-parameters ^
 --db-cluster-parameter-group-name lab-test-apg-global *
 --source user
```

O comando retorna uma saída semelhante à seguinte para todos os parâmetros `user` que não são `default-engine` ou parâmetros de cluster de banco de dados `system`.

```
{
    "Parameters": [
        {
            "ParameterName": "rds.global_db_rpo",
            "ParameterValue": "60",
            "Description": "(s) Recovery point objective threshold, in seconds, that
 blocks user commits when it is violated.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "20-2147483647",
            "IsModifiable": true,
            "ApplyMethod": "immediate",
            "SupportedEngineModes": [
                "provisioned"
            ]
        }
```

```
    ]
}
```

Para saber mais sobre a visualização de parâmetros do grupo de parâmetros do cluster, consulte Visualizar valores de parâmetros de um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

Desabilitar o objetivo de ponto de recuperação

Para desabilitar o RPO, redefina o parâmetro `rds.global_db_rpo`. É possível redefinir parâmetros usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como desabilitar o RPO

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros de cluster do banco de dados primário.
4. Escolha Edit parameters.
5. Escolha a caixa ao lado do parâmetro rds.global_db_rpo.
6. Escolha Redefinir.
7. Quando a tela mostrar Redefinir parâmetros no grupo de parâmetros do banco de dados, escolha Redefinir parâmetros.

Para ter mais informações sobre como redefinir um parâmetro com o console, consulte Modificar parâmetros em um grupo de parâmetros de cluster de banco de dadosno Amazon Aurora.

AWS CLI

Para redefinir o parâmetro `rds.global_db_rpo`, use o comando reset-db-cluster-parameter-group .

Para Linux, macOS ou Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name global_db_cluster_parameter_group \
```

```
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name global_db_cluster_parameter_group ^
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

API do RDS

Para redefinir o parâmetro `rds.global_db_rpo`, use a operação ResetDBClusterParameterGroup da API do Amazon RDS.

Resiliência entre regiões para clusters secundários do Global Database

As versões 16.6, 15.10, 14.15, 13.18, 12.22 ou posteriores do Aurora PostgreSQL ou as versões 3.09 do Aurora MySQL contêm melhorias de disponibilidade que permitem que réplicas de leitura da região secundária mantenham a continuidade do serviço durante eventos não planejados, como falhas de hardware, interrupções na rede entre regiões da AWS, grandes volumes de transferências de dados entre os clusters e outros.

Embora as réplicas de leitura permaneçam disponíveis para as solicitações da aplicação, o atraso na replicação pode continuar aumentando até a resolução do evento não planejado. Você pode monitorar o atraso entre os clusters primário e secundário usando a métrica `AuroraGlobalDBProgressLag` do CloudWatch. Para medir o atraso de ponta a ponta, incluindo qualquer atraso entre o volume do cluster e as instâncias de banco de dados do cluster secundário, adicione os valores das métricas `AuroraGlobalDBProgressLag` e `AuroraReplicaLag` do CloudWatch. Para obter mais informações sobre as métricas, consulte Referência de métricas do Amazon Aurora.

A disponibilidade de leitura do Global Database para o Aurora MySQL e versões anteriores do Aurora PostgreSQL pode ser afetada durante esses eventos não planejados.

Para ter mais informações sobre os novos recursos no Aurora PostgreSQL 16.6, 15.10, 14.15, 13.18 e 12.22, consulte PostgreSQL 16.6 em Notas de lançamento do Aurora PostgreSQL.

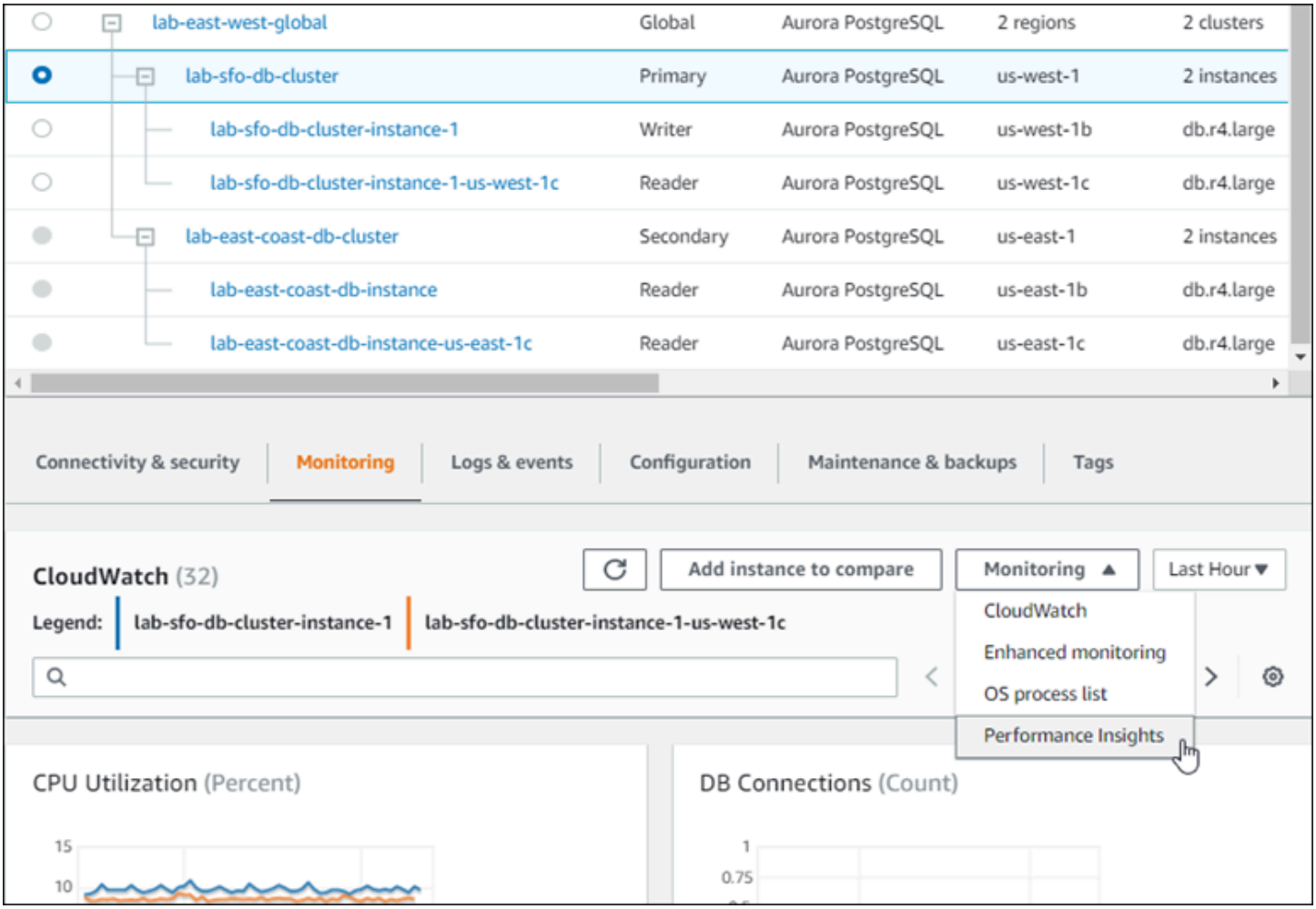
Para ter informações sobre novos recursos no Aurora MySQL versões 3.09 e posteriores, consulte Database engine updates for Amazon Aurora MySQL version 3 em Notas de lançamento do Aurora MySQL.

Monitorar um banco de dados global do Amazon Aurora

Ao criar os clusters de bancos de dados Aurora que compõem seu banco de dados Aurora global, você pode escolher muitas opções que permitem monitorar a performance do cluster de banco de dados. As opções incluem o seguinte:

- Amazon RDS Performance Insights – Habilita o esquema de performance no mecanismo de banco de dados do Aurora subjacente Para saber mais sobre Performance Insights e bancos de dados Aurora globais, consulte Monitorando um banco de dados Amazon Aurora global com Performance Insights Amazon RDS.
- Monitoramento aprimorado – Gera métricas para utilização de processos ou thread na CPU. Para saber mais sobre o monitoramento aprimorado, consulte Monitorar métricas do SO com o monitoramento avançado.
- Amazon CloudWatch Logs – Publica tipos de log especificados para CloudWatch Logs. Os logs de erros são publicados por padrão, mas você pode escolher outros logs específicos para o mecanismo de banco de dados do Aurora.
  - Para clusters de bancos de dados Aurora baseados em Aurora MySQL–, você pode exportar o log de auditoria, o log geral e o log de consulta lenta.
  - Para clusters de bancos de dados do Aurora baseados no Aurora PostgreSQL, você pode exportar o log do PostgreSQL.
- Quanto aos bancos de dados globais baseados no Aurora MySQL, você pode consultar tabelas `information_schema` específicas para conferir o status do banco de dados global do Aurora e as respectivas instâncias. Para saber como, consulte Monitorar bancos de dados globais baseados no Aurora MySQL.
- Quanto a bancos de dados globais baseados no Aurora PostgreSQL, você pode usar funções específicas para conferir o status do banco de dados global do Aurora e as respectivas instâncias. Para saber como, consulte Monitorar banco de dados globais baseados no Aurora PostgreSQL.

A captura de tela a seguir mostra algumas das opções disponíveis na guia Monitoring (Monitoramento) de um cluster de bancos de dados Aurora primário em um banco de dados Aurora global.



Para obter mais informações, consulte Monitorar métricas em um cluster do Amazon Aurora.

Monitorando um banco de dados Amazon Aurora global com Performance Insights Amazon RDS

É possível usar o Performance Insights Amazon RDS e seus bancos de dados do Aurora globais. Você habilita esse recurso individualmente, para cada cluster de bancos de dados Aurora em seu banco de dados do Aurora global. Para isso, escolha Enable Performance Insights (Habilitar Performance Insights) na seção Additional configuration (Configuração adicional) da página Create database (Criar banco de dados). Ou, você pode Modificar seus clusters de banco de dados Aurora para usar esse recurso depois que estiverem em execução. É possível habilitar ou desabilitar o Performance Insights para cada cluster que faz parte do seu banco de dados do Aurora global.

Os relatórios criados pelo Performance Insights aplicam-se a cada cluster no banco de dados global. Ao adicionar uma nova Região da AWS secundária ao Aurora Global Database, que já está usando o

Performance Insights, certifique-se de habilitar o Performance Insights no cluster recém-adicionado. Ele não herda a configuração de Performance Insights do banco de dados global existente.

É possível alternar entre Regiões da AWS enquanto visualiza a página do Performance Insights de uma instância de banco de dados anexada a um banco de dados global. No entanto, talvez você não veja as informações sobre a performance imediatamente após alternar entre as Regiões da AWS. Embora as instâncias de banco de dados possam ter nomes idênticos em cada Região da AWS, o URL associado ao Performance Insights é diferente para cada instância de banco de dados. Após alternar entre Regiões da AWS, selecione o nome da instância de banco de dados novamente no painel de navegação de Performance Insights.

Para instâncias de banco de dados associadas a um banco de dados global, os fatores que afetam a performance podem ser diferentes em cada Região da AWS. Por exemplo, as instâncias de banco de dados em cada Região da AWS podem ter uma capacidade diferente.

Para saber mais sobre como usar o Performance Insights, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

Monitorar bancos de dados globais Aurora com fluxos de atividades de banco de dados

Ao usar fluxos de atividade de banco de dados, você pode monitorar e definir alarmes para atividades de auditoria nos cluster de banco de dados em seu banco de dados global. Você inicia um fluxo de atividade de banco de dados em cada cluster de banco de dados separadamente. Cada cluster fornece dados de auditoria ao seu próprio fluxo do Kinesis dentro de sua própria Região da AWS. Para obter mais informações, consulte Monitorar o Amazon Aurora com o recurso Database Activity Streams.

Monitorar bancos de dados globais baseados no Aurora MySQL

Para visualizar o status de um banco de dados global baseado no Aurora MySQL, consulte as tabelas information_schema.aurora_global_db_status e information_schema.aurora_global_db_instance_stat

> **Note**
>
> As tabelas `information_schema.aurora_global_db_status` e `information_schema.aurora_global_db_instance_status` só estão disponíveis com bancos de dados globais do Aurora MySQL versão 3.04.0 e posterior.

Como monitorar um banco de dados global baseado no Aurora MySQL

1. Conecte-se ao endpoint do cluster primário do banco de dados global usando um cliente MySQL. Para obter mais informações sobre como se conectar, consulte Conectar-se ao Amazon Aurora Global Database.
2. Consulte a tabela `information_schema.aurora_global_db_status` em um comando mysql para listar os volumes primário e secundário. Essa consulta retorna os tempos de atraso dos clusters de banco de dados secundários do banco de dados global, como no exemplo a seguir.

```
mysql> select * from information_schema.aurora_global_db_status;
```

```
AWS_REGION | HIGHEST_LSN_WRITTEN | DURABILITY_LAG_IN_MILLISECONDS |
 RPO_LAG_IN_MILLISECONDS | LAST_LAG_CALCULATION_TIMESTAMP | OLDEST_READ_VIEW_TRX_ID
-----------+---------------------+--------------------------------
+------------------------+---------------------------------
+------------------------
us-east-1  |           183537946 |                              0   |
      0 |  1970-01-01 00:00:00.000000     |               0
us-west-2  |           183537944 |                              428 |
      0 |  2023-02-18 01:26:41.925000     |               20806982
(2 rows)
```

A saída inclui uma linha para cada cluster de banco de dados global que contém as seguintes colunas:

- AWS_REGION: a Região da AWS em que esse cluster de banco de dados está. Para ver as tabelas que listam as Regiões da AWS por mecanismo, consulte Disponibilidade de regiões.
- atualmente gravado nesse cluster de banco deHIGHEST_LSN_WRITTEN: o maior número de sequência de log (LSN) atualmente gravado nesse cluster de banco de dados.

  Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.
- DURABILITY_LAG_IN_MILLISECONDS: a diferença nos valores de carimbo de data e hora entre o `HIGHEST_LSN_WRITTEN` em um cluster de banco de dados secundário e o `HIGHEST_LSN_WRITTEN` no cluster de banco de dados primário. Esse valor é sempre 0 no cluster de banco de dados primário do banco de dados global do Aurora.

- RPO_LAG_IN_MILLISECONDS: o atraso do objetivo de ponto de recuperação (RPO). O atraso do RPO é o tempo necessário para que a transação COMMIT mais recente do usuário seja armazenada em um cluster de banco de dados secundário após ser armazenada no cluster de banco de dados primário de um banco de dados Aurora global. Esse valor é sempre 0 no cluster de banco de dados primário do banco de dados global do Aurora.

  Em termos simples, essa métrica calcula o objetivo do ponto de recuperação de cada cluster de banco de dados do Aurora MySQL no banco de dados global do Aurora, ou seja, quantos dados poderão ser perdidos se houver uma interrupção. Tal como o atraso, o RPO é medido em tempo.
- LAST_LAG_CALCULATION_TIMESTAMP: o carimbo de data e hora que especifica quando os valores foram calculados pela última vez para `DURABILITY_LAG_IN_MILLISECONDS` e `RPO_LAG_IN_MILLISECONDS`. Um valor de tempo como `1970-01-01 00:00:00+00` indica que este é o cluster de banco de dados primário.
- OLDEST_READ_VIEW_TRX_ID: o ID da transação mais antiga que a instância de banco de dados do gravador pode limpar.

3. Consulte a tabela `information_schema.aurora_global_db_instance_status` para listar todas as instâncias de banco de dados secundárias tanto para o cluster de banco de dados primário como para os clusters de banco de dados secundários.

```
mysql> select * from information_schema.aurora_global_db_instance_status;
```

```
SERVER_ID            |              SESSION_ID              | AWS_REGION
 | DURABLE_LSN | HIGHEST_LSN_RECEIVED | OLDEST_READ_VIEW_TRX_ID |
 OLDEST_READ_VIEW_LSN | VISIBILITY_LAG_IN_MSEC
--------------------+--------------------------------------+------------
+-------------+----------------------+-------------------------
+----------------------+-----------------------
ams-gdb-primary-i2   | MASTER_SESSION_ID                    | us-east-1  |
 183537698   |                    0 |                      0 |
 0 |                      0
ams-gdb-secondary-i1 | cc43165b-bdc6-4651-abbf-4f74f08bf931 | us-west-2  |
 183537689   |           183537692 |                20806928 |
 183537682 |                      0
ams-gdb-secondary-i2 | 53303ff0-70b5-411f-bc86-28d7a53f8c19 | us-west-2  |
 183537689   |           183537692 |                20806928 |
 183537682 |                    677
```

```
ams-gdb-primary-i1   | 5af1e20f-43db-421f-9f0d-2b92774c7d02 | us-east-1  |
 183537697   |            183537698 |               20806930 |
 183537691 |                    21
(4 rows)
```

A saída inclui uma linha para cada instância de banco de dados global que contém as seguintes colunas:

- SERVER_ID: o identificador do servidor para a instância de banco de dados.
- SESSION_ID: um identificador exclusivo para a sessão atual. O valor de `MASTER_SESSION_ID` identifica a instância de banco de dados do leitor (primário).
- AWS_REGION: a Região da AWS em que essa instância de banco de dados está. Para ver as tabelas que listam as Regiões da AWS por mecanismo, consulte Disponibilidade de regiões.
- DURABLE_LSN: o LSN tornou-se durável no armazenamento.
- HIGHEST_LSN_RECEIVED: o LSN mais alto recebido pela instância de banco de dados da instância de banco de dados do gravador.
- OLDEST_READ_VIEW_TRX_ID: o ID da transação mais antiga que a instância de banco de dados do gravador pode limpar.
- OLDEST_READ_VIEW_LSN: o LSN mais antigo usado pela instância de banco de dados para fazer a leitura no armazenamento.
- VISIBILITY_LAG_IN_MSEC: para leitores no cluster de banco de dados primário, até que ponto essa instância de banco de dados está atrasada em relação à instância de banco de dados de gravador em milissegundos. Para leitores em um cluster de banco de dados secundário, até que ponto essa instância de banco de dados está atrasada em relação ao volume secundário em milissegundos.

Para ver como esses valores mudam ao longo do tempo, considere o bloco de transação a seguir, em que uma inserção de tabela leva uma hora.

```
mysql> BEGIN;
mysql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
mysql> COMMIT;
```

Em alguns casos, pode haver uma desconexão de rede entre o cluster de banco de dados primário e o cluster de banco de dados secundário após a declaração `BEGIN`. Em caso afirmativo, o

valor DURABILITY_LAG_EM_MILISSEGUNDOS do cluster de banco de dados secundário começa a aumentar. No final da declaração `INSERT`, o valor DURABILITY_LAG_EM_MILISSEGUNDOS é de 1 hora. No entanto, o valor RPO_LAG_IN_MILLISECONDS é 0 porque todos os dados do usuário confirmados entre o cluster de banco de dados primário e o cluster de banco de dados secundário ainda são os mesmos. Assim que a declaração `COMMIT` é concluída, o valor RPO_LAG_IN_MILISSEGUNDOS aumenta.

Monitorar banco de dados globais baseados no Aurora PostgreSQL

Para visualizar o status de um banco de dados global baseado no Aurora PostgreSQL, use as funções `aurora_global_db_status` e `aurora_global_db_instance_status`.

> **Note**
>
> Somente o Aurora PostgreSQL oferece suporte às funções `aurora_global_db_status` e `aurora_global_db_instance_status`.

Para monitorar um banco de dados global baseado em Aurora PostgreSQL

1. Conecte-se ao endpoint de cluster primário do banco de dados global usando um utilitário do PostgreSQL, como psql. Para obter mais informações sobre como se conectar, consulte Conectar-se ao Amazon Aurora Global Database.
2. Use a função `aurora_global_db_status` em um comando psql para listar os volumes primário e secundário. Isso mostra os tempos de atraso dos clusters de banco de dados secundários do banco de dados global.

```
postgres=> select * from aurora_global_db_status();
```

```
aws_region | highest_lsn_written | durability_lag_in_msec | rpo_lag_in_msec |
 last_lag_calculation_time  | feedback_epoch | feedback_xmin
------------+---------------------+------------------------+-----------------
+----------------------------+----------------+--------------
us-east-1  |         93763984222 |                     -1 |              -1 |
 1970-01-01 00:00:00+00     |              0 |              0
us-west-2  |         93763984222 |                    900 |            1090 |
 2020-05-12 22:49:14.328+00 |              2 |    3315479243
(2 rows)
```

A saída inclui uma linha para cada cluster de banco de dados global que contém as seguintes colunas:

- aws_region: a Região da AWS em que esse cluster de banco de dados está. Para ver as tabelas que listam as Regiões da AWS por mecanismo, consulte Disponibilidade de regiões.
- highest_lsn_written: o maior número de sequência de log (LSN) atualmente gravado nesse cluster de banco de dados.

  Um número de sequência de log (LSN) é um número sequencial exclusivo que identifica um registo no log de transações do banco de dados. LSNs são ordenados de forma que um LSN maior represente uma transação posterior.
- durability_lag_in_msec: a diferença de time stamp entre o número de sequência de log mais alto gravado em um cluster de banco de dados secundário (`highest_lsn_written`) e o `highest_lsn_written` no cluster de banco de dados primário.
- rpo_lag_in_msec: o atraso do Objetivo de ponto de recuperação gerenciado (RPO). Esse atraso é a diferença de tempo entre a confirmação da transação do usuário mais recente armazenada em um cluster de banco de dados secundário e a confirmação da transação do usuário mais recente armazenada no cluster do banco de dados primário.
- last_lag_calculation_time: o time stamp em que os valores foram calculados pela última vez para `durability_lag_in_msec` e `rpo_lag_in_msec`.
- feedback_epoch: epoch que um cluster de banco de dados secundário usa quando gera informações de standby a quente.

  O standby a quente acontece quando um cluster de banco de dados pode se conectar e consultar enquanto o servidor está no modo de recuperação ou de espera. O feedback de standby a quente são informações sobre o cluster de banco de dados quando ele está em standby a quente. Para obter mais informações, consulte Hot standby na documentação do PostgreSQL.
- feedback_xmin: o ID mínimo (mais antigo) de transação ativo usado por um cluster de banco de dados secundário.

3. Use a função `aurora_global_db_instance_status` para listar todas as instâncias de banco de dados secundárias tanto para o cluster de banco de dados primário como para os clusters de banco de dados secundários.

```
postgres=> select * from aurora_global_db_instance_status();
```

```
server_id                                    |              session_id
  | aws_region | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
---------------------------------------------+-------------------------------------
+-----------+-------------+-----------------+----------------+--------------
+----------------------+-----------------------
apg-global-db-rpo-mammothrw-elephantro-1-n1 | MASTER_SESSION_ID
 | us-east-1  | 93763985102 |                 |                |              |
                     |
apg-global-db-rpo-mammothrw-elephantro-1-n2 | f38430cf-6576-479a-b296-dc06b1b1964a
 | us-east-1  | 93763985099 |      93763985102 |              2 |    3315479243 |
        93763985095 |                      10
apg-global-db-rpo-elephantro-mammothrw-n1   | 0d9f1d98-04ad-4aa4-8fdd-e08674cbbbfe
 | us-west-2  | 93763985095 |      93763985099 |              2 |    3315479243 |
        93763985089 |                    1017
(3 rows)
```

A saída inclui uma linha para cada instância de banco de dados global que contém as seguintes colunas:

- server_id: o identificador do servidor para a instância de banco de dados.
- session_id: um identificador exclusivo para a sessão atual.
- aws_region: a Região da AWS em que essa instância de banco de dados está. Para ver as tabelas que listam as Regiões da AWS por mecanismo, consulte Disponibilidade de regiões.
- durable_lsn – o LSN tornou-se durável no armazenamento.
- highest_lsn_rcvd: o LSN mais alto recebido pela instância de banco de dados da instância de banco de dados do gravador.
- feedback_epoch: a época que a instância de banco de dados usa quando gera informações de standby a quente.

  O standby a quente acontece quando uma instância de banco de dados pode se conectar e consultar enquanto o servidor está no modo de recuperação ou espera. O feedback de standby a quente são informações sobre a instância de banco de dados quando ela está em standby a quente. Para obter mais informações, consulte a documentação do PostgreSQL em Hot standby.
- feedback_xmin: o ID mínimo (mais antigo) de transação ativo usado pela instância de banco de dados.

- oldest_read_view_lsn: o LSN mais antigo usado pela instância de banco de dados para fazer a leitura no armazenamento.
- visibility_lag_in_msec: até que ponto essa instância de banco de dados está atrasada em relação à instância de banco de dados do gravador.

Para ver como esses valores mudam ao longo do tempo, considere o bloco de transação a seguir, em que uma inserção de tabela leva uma hora.

```
psql> BEGIN;
psql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
psql> COMMIT;
```

Em alguns casos, pode haver uma desconexão de rede entre o cluster de banco de dados primário e o cluster de banco de dados secundário após a instrução `BEGIN`. Nesse caso, o valor `durability_lag_in_msec` do cluster de banco de dados secundário começa a aumentar. No final da instrução `INSERT`, o `durability_lag_in_msec` valor é 1 hora. No entanto, o valor `rpo_lag_in_msec` é 0 porque todos os dados do usuário confirmados entre o cluster de banco de dados primário e o cluster de banco de dados secundário ainda são os mesmos. Assim que a instrução `COMMIT` for concluída, o valor `rpo_lag_in_msec` aumentará.

Usar implantações azul/verde para o Amazon Aurora Global Database

Implantações azul/verde do Amazon RDS oferecem a capacidade de testar alterações de banco de dados com segurança. Para seu banco de dados global, as implantações azul/verde permitem que você realize atualizações e operações de manutenção com o mínimo de tempo de inatividade. Você pode criar um ambiente de preparação totalmente gerenciado (verde) que espelhe todo o seu ambiente de produção (azul), incluindo o cluster primário e todas as regiões secundárias associadas em várias regiões da AWS. O ambiente de preparação reflete sua configuração de produção, permitindo que você teste as alterações de forma confiável antes de migrar para o novo ambiente. Durante todo o processo, as implantações azul/verde mantêm os ambientes de preparação e produção sincronizados. Quando estiver tudo pronto para transformar seu ambiente de preparação no novo ambiente de produção, realize uma transição azul/verde. Essa operação transfere suas regiões principal e todas as secundárias para o ambiente verde, com tempo de inatividade normalmente inferior a um minuto. O serviço renomeia automaticamente clusters, instâncias e endpoints para corresponder ao ambiente de produção original, permitindo que

suas aplicações acessem o novo ambiente de produção sem exigir alterações na configuração e minimizando a sobrecarga operacional.

Tópicos

- Benefícios de usar implantações azul/verde com o Aurora Global Database
- Como implantações azul/verde funcionam com o Aurora Global Database

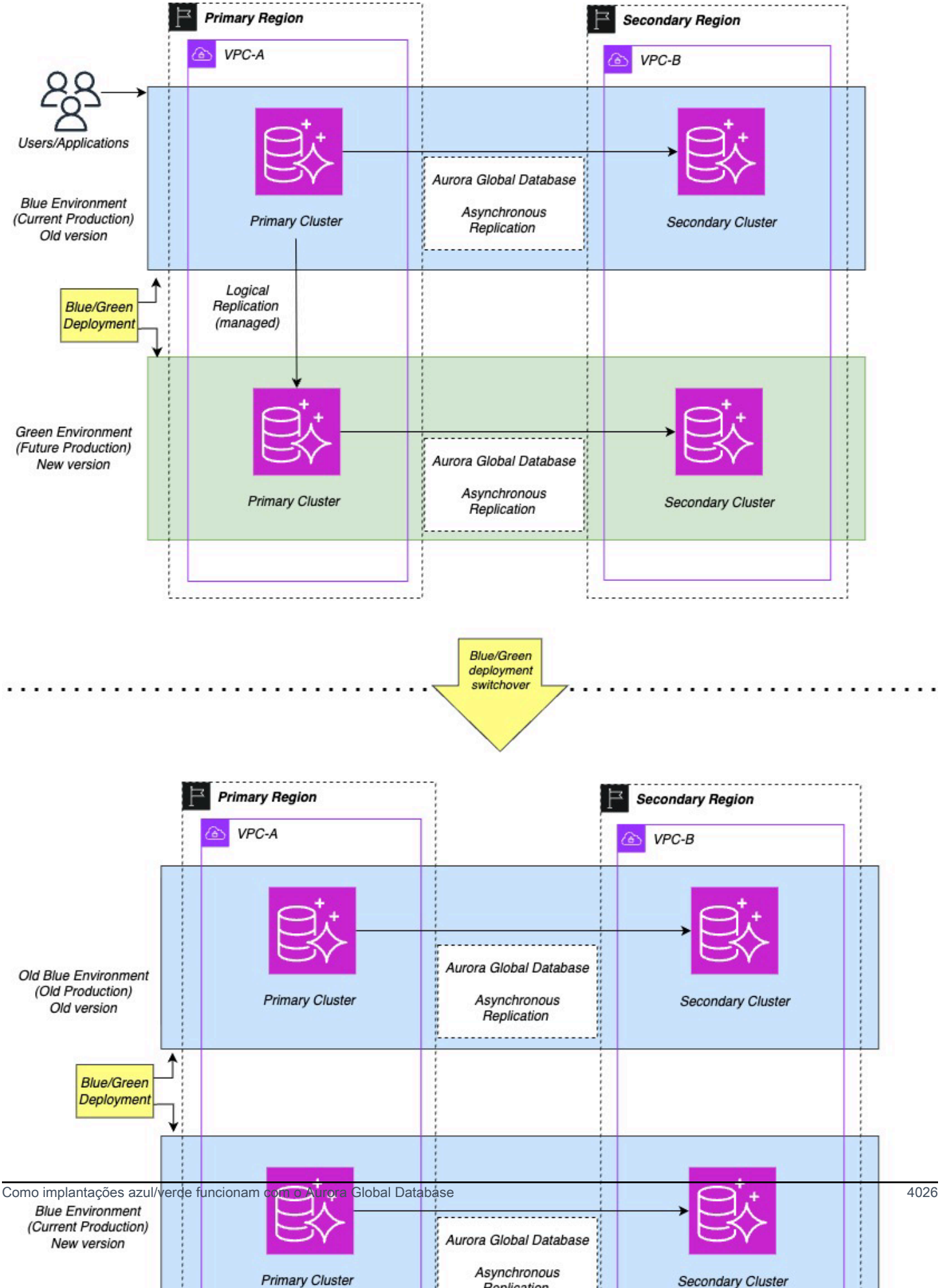
Benefícios de usar implantações azul/verde com o Aurora Global Database

- Realize atualizações da versão principal, da versão secundária e do sistema, incluindo patches de banco de dados e atualizações do sistema operacional nos Aurora Global Databases, mantendo o mínimo de tempo de inatividade.
- Crie um ambiente de preparação (verde) pronto para produção nas regiões principal e secundária do banco de dados global para testar e implementar recursos mais novos do banco de dados.
- Faça a transição com segurança usando barreiras de proteção de transição integradas com tempo de inatividade normalmente inferior a um minuto, dependendo de sua workload.
- Mantém os recursos de recuperação de desastres durante o processo de transição azul/verde, permitindo o failover do banco de dados global durante a transição azul/verde.
- Seu tráfego será direcionado para o novo ambiente de produção sem exigir nenhuma alteração na aplicação.

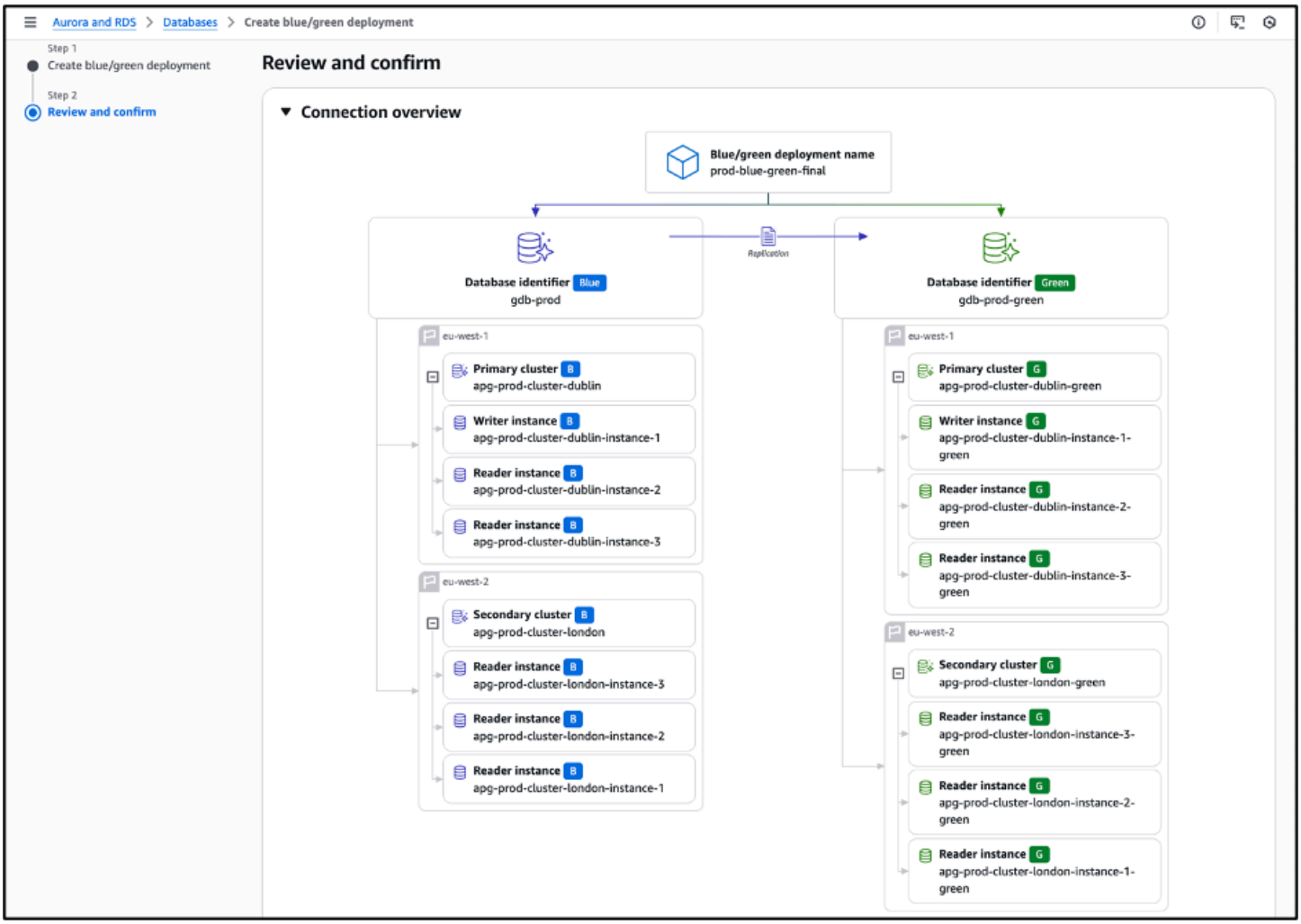
Como implantações azul/verde funcionam com o Aurora Global Database

Para acessar detalhes sobre como criar, visualizar, alternar e excluir uma implantação azul/verde, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados. Você pode usar isso para atualizações de versões primárias ou secundárias, melhorias na performance do banco de dados por meio de atualizações de parâmetros e adoção de novos recursos de banco de dados, com tempo de inatividade mínimo.

Uma representação da aparência de uma implantação azul/verde do Aurora Global Database com uma região secundária antes e depois de uma transição azul/verde é mostrada abaixo.


Primary Region
VPC-A
Secondary Region
VPC-B
Users/Applications
Blue Environment
(Current Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Logical
Replication
(managed)
Green Environment
(Future Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
deployment
switchover
Primary Region
VPC-A
Secondary Region
VPC-B
Old Blue Environment
(Old Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Blue Environment
(Current Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster

Você pode criar uma implantação azul/verde a partir da região principal do seu banco de dados global. Selecione as configurações do mecanismo, como versão principal ou secundária do mecanismo, grupo de parâmetros de banco de dados e grupo de parâmetros do cluster de banco de dados para o ambiente verde. O Amazon RDS copia a topologia do ambiente azul no ambiente verde. Uma representação visual no Console de gerenciamento da AWS é mostrada abaixo.



> **Note**
>
> O failover global é aceito durante uma transição azul/verde, mas a transição global não é aceita durante uma transição azul/verde.

Quando você inicia um failover global durante uma transição azul/verde do RDS, a região de destino é revertida automaticamente para o ambiente azul ou avança para o ambiente verde antes que o failover global ocorra.

Para acessar informações sobre como criar, visualizar, fazer a transição e excluir implantações azul/verde, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados. Siga o mesmo fluxo de trabalho para bancos de dados globais, com instruções específicas anotadas nas etapas relevantes.

Usar bancos de dados globais do Amazon Aurora com outros produtos da AWS

Você pode usar seus bancos de dados globais do Aurora com outros produtos da AWS, como o Amazon S3 e AWS Lambda. Isso requer que todos os clusters de banco de dados Aurora em seu banco de dados global tenham os mesmos privilégios, funções externas etc. nas respectivas Regiões da AWS. Como um cluster de banco de dados Aurora secundário somente leitura em um banco de dados Aurora global pode ser promovido para a função de primário, recomendamos que você configure privilégios de gravação com antecedência, em todos os clusters de banco de dados de Aurora para quaisquer serviços que você planeja usar com seu banco de dados do Aurora global.

Os procedimentos a seguir resumem as ações a serem tomadas para cada AWS service (Serviço da AWS).

Para invocar as funções do AWS Lambda de um banco de dados global Aurora

1. Para todos os clusters do Aurora que constituam o banco de dados global Aurora, realize os procedimentos em Invocar uma função do Lambda a partir de um cluster de banco de dados do Amazon Aurora MySQL.
2. Para cada cluster no banco de dados Aurora global, defina o (ARN) da nova função (IAM) do IAM.
3. Para permitir que os usuários do banco de dados em um banco de dados global Aurora invoquem funções do Lambda, associe a função criada em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS com cada cluster no banco de dados global Aurora.
4. Configure cada cluster no banco de dados global Aurora para permitir conexões de saída com o Lambda. Para obter instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

Carregar dados do Amazon S3.

1. Para todos os clusters do Aurora que constituam o banco de dados global Aurora, realize os procedimentos em Carregar dados em um cluster de banco de dados do Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.
2. Para cada cluster do Aurora no banco de dados global, defina o parâmetro de cluster de banco de dados `aurora_load_from_s3_role` ou `aws_default_s3_role` como o ARN da nova função do IAM. Se uma função do IAM não for especificada para `aurora_load_from_s3_role`, o Aurora usará a função do IAM especificada em `aws_default_s3_role`.
3. Para permitir que os usuários de um banco de dados global Aurora acessem o S3, associe a função criada em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS a cada cluster do Aurora no banco de dados global.
4. Configure cada cluster do Aurora no banco de dados global para permitir conexões de saída com o S3. Para obter instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

Para salvar dados consultados em Amazon S3

1. Para todos os clusters do Aurora que constituem o banco de dados global do Aurora, realize os procedimentos em Salvar dados a partir de um cluster de banco de dados do Amazon Aurora MySQL em arquivos de texto de um bucket do Amazon S3 ou em Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3.
2. Para cada cluster do Aurora no banco de dados global, defina o parâmetro de cluster de banco de dados `aurora_select_into_s3_role` ou `aws_default_s3_role` como o ARN da nova função do IAM. Se uma função do IAM não for especificada para `aurora_select_into_s3_role`, o Aurora usará a função do IAM especificada em `aws_default_s3_role`.
3. Para permitir que os usuários de um banco de dados global Aurora acessem o S3, associe a função criada em Criar uma função do IAM para permitir que o Amazon Aurora acesse produtos da AWS a cada cluster do Aurora no banco de dados global.
4. Configure cada cluster do Aurora no banco de dados global para permitir conexões de saída com o S3. Para instruções, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.

Usar o Controlador de Recuperação de Aplicações (ARC) da Amazon com o Aurora Global Database

Ao planejar sua estratégia de continuidade de negócios e recuperação de desastres, é necessário orquestrar a recuperação entre as pilhas de aplicações e as respectivas dependências. O Controlador de Recuperação de Aplicações da Amazon (ARC) se integra ao Aurora Global Database para automatizar esse processo por meio do comutador de regiões do ARC, uma solução centralizada para recuperação automatizada de aplicações em várias regiões. O comutador de regiões orquestra as etapas de failover entre contas e regiões da AWS, oferece painéis de recuperação em tempo real e gera relatórios de conformidade agregando dados entre recursos e contas. Saiba mais sobre como usar o comutador de regiões no Aurora Global Database.

Atualizar um Amazon Aurora Global Database

A atualização de um Aurora Global Database segue os mesmos procedimentos que a atualização de clusters de banco de dados Aurora. No entanto, veja a seguir algumas diferenças importantes a serem observadas antes de iniciar o processo.

Recomendamos que você atualize os clusters de banco de dados primário e secundário para a mesma versão. Você só poderá realizar um failover gerenciado de banco de dados entre regiões em um banco de dados global do Aurora se os clusters de banco de dados primário e secundário tiverem as mesmas versões principal, secundária e nível de patch do mecanismo. No entanto, os níveis de patch podem ser diferentes, dependendo da versão secundária do mecanismo. Para ter mais informações, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões.

Atualizações de versão principal

Quando você executa uma atualização de versão principal de um banco de dados Amazon Aurora global, você atualiza o cluster de banco de dados global em vez dos clusters individuais que ele contém.

Para saber como atualizar um banco de dados Aurora PostgreSQL global para uma versão principal superior, consulte Atualizações principais de bancos de dados globais.

> **Note**
>
> Com um banco de dados global Aurora baseado no Aurora PostgreSQL, você não poderá executar uma atualização de versão principal do mecanismo de banco de dados Aurora se o recurso do objetivo de ponto de recuperação (RPO) estiver ativado. Para ter mais informações sobre o recurso RPO, consulte Gerenciamento de RPOs para bancos de dados globais baseados em Aurora PostgreSQL–.

Para saber como atualizar um banco de dados Aurora MySQL global para uma versão principal superior, consulte Principais atualizações no local para bancos de dados globais.

> **Note**
>
> Com um banco de dados global do Aurora baseado no Aurora MySQL, será possível realizar uma atualização no local do Aurora MySQL versão 2 para a versão 3 somente se você definir o parâmetro `lower_case_table_names` como padrão e reinicializar o banco de dados global.
> Para realizar uma atualização de versão principal para o Aurora MySQL versão 3 enquanto usa `lower_case_table_names`, use o seguinte processo:
>
> 1. Remova todas as regiões secundárias do cluster global. Siga as etapas em Remover um cluster de um banco de dados global do Amazon Aurora.
> 2. Atualize a versão do mecanismo da região primária para o Aurora MySQL versão 3. Siga as etapas em Como realizar uma atualização local.
> 3. Adicione regiões secundárias ao cluster global. Siga as etapas em Adicionar uma Região da AWS a um Amazon Aurora Global Database.
>
> Você também pode usar o método de restauração de snapshot. Para ter mais informações, consulte Restauração de um snapshot de um cluster de banco de dados.

Atualizações de versões secundárias

É possível atualizar o Aurora Global Database para uma versão mais recente de mecanismo secundário em todas as regiões com uma única operação gerenciada e tempo de inatividade

mínimo, o que elimina a necessidade de atualizar manual e separadamente cada cluster e reduz os custos operacionais indiretos do gerenciamento de clusters globais.

Noções básicas sobre atualizações de versões secundárias de banco de dados globais

É possível atualizar a versão secundária de um banco de dados global por meio da API do RDS, da AWS CLI ou do Console de gerenciamento da AWS. Essa operação única orquestra a atualização no cluster primário e em todos os clusters secundários (espelho). Se ocorrerem problemas durante a atualização, o serviço será revertido automaticamente para a versão existente.

> **Note**
>
> No momento, esse recurso gerenciado é oferecido somente para mecanismos compatíveis com o Aurora PostgreSQL.

Ao iniciar uma atualização de versão secundária do banco de dados global usando o comando `modify-global-cluster`, especifique a versão do mecanismo de destino para que o serviço coordene a atualização em todos os clusters. Essa atualização é aplicada imediatamente.

Para Linux, macOS ou Unix:

```
aws rds modify-global-cluster \
    --global-cluster-identifier global_cluster_identifier \
    --engine-version target_engine_version
```

Para Windows:

```
aws rds modify-global-cluster ^
    --global-cluster-identifier global_cluster_identifier ^
    --engine-version target_engine_version
```

Considerações sobre atualizações de versões secundárias

Ao planejar uma atualização de versão secundária para um banco de dados global, pense no seguinte:

- O recurso gerenciado se aplica somente a atualizações de versões secundárias. As atualizações de versão de patch continuam usando as ações existentes de manutenção de atualização do sistema.
- O recurso gerenciado pode ser usado somente com clusters globais do Aurora PostgreSQL.

É possível atualizar cada cluster individualmente na topologia de cluster global. Se optar por essa abordagem, atualize todos os clusters secundários antes de atualizar o cluster primário. Ao atualizar, garanta que os clusters de banco de dados primário e secundário sejam atualizados para o mesmo nível de versão secundária e de patch. Para atualizar o nível do patch, aplique todas as ações de manutenção pendentes no cluster secundário. Para saber como atualizar um banco de dados Aurora PostgreSQL global para uma versão secundária superior, consulte Como realizar atualizações de versão secundária e aplicar patches.

Atualizações de versões secundárias de um banco de dados global do Aurora MySQL

Para saber como atualizar um banco de dados Aurora MySQL global para uma versão secundária superior, consulte Atualizar o Aurora MySQL modificando a versão do mecanismo.

Antes de realizar a atualização, revise as seguintes considerações:

- O upgrade da versão secundária de um cluster secundário não afeta de forma alguma a disponibilidade ou o uso do cluster primário.
- Um cluster secundário deve ter pelo menos uma instância de banco de dados para executar uma atualização secundária.
- Se fizer upgrade de um banco de dados global do Aurora MySQL para uma versão 2.11.*, você deverá atualizar os clusters de banco de dados primário e secundário para exatamente a mesma versão, inclusive o nível de patch.
- Para oferecer suporte a transições ou failovers gerenciados entre regiões, é possível atualizar os clusters de banco de dados primário e secundário para a mesma versão exata, inclusive o nível de patch. Esse requisito se aplica ao Aurora MySQL e a algumas versões do Aurora PostgreSQL. Para conferir uma lista de versões que permitem transições e failovers entre clusters que executam diferentes níveis de patch, consulte Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões.

Compatibilidade em nível de patch para transições e failovers gerenciados entre regiões

Se o Aurora Global Database estiver executando uma das versões secundárias do mecanismo, será possível realizar transições ou failovers gerenciados entre regiões mesmo se os níveis de patch dos clusters de banco de dados primário e secundário forem diferentes. Para versões secundárias do mecanismo anteriores às desta lista, os clusters de banco de dados primário e secundário deverão executar os mesmos níveis principal, secundário e de patch a fim de realizar transições ou failovers gerenciados entre regiões. Revise as informações da versão e as observações na tabela a seguir ao planejar as atualizações para o cluster primário, secundário ou ambos.

> **Note**
>
> Para failovers manuais entre regiões, você pode realizar o processo de failover desde que o cluster de banco de dados secundário de destino esteja executando a mesma versão de mecanismo principal e secundária do cluster de banco de dados primário. Nesse caso, os níveis de patch não precisam ser iguais.
> Se as versões do mecanismo exigirem níveis de patch idênticos, será possível realizar o failover manualmente seguindo as etapas em Realizar failovers manuais para bancos de dados globais do Aurora.

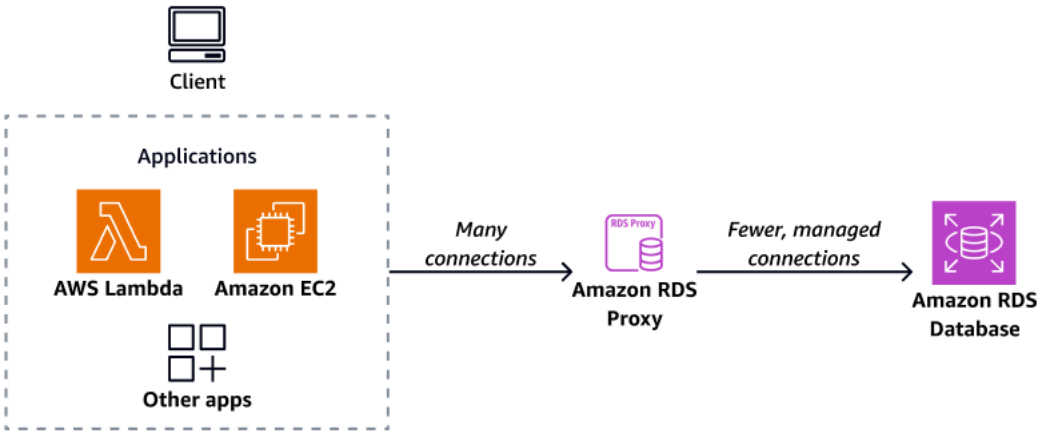
Mecanismo do banco de dados	Versões secundárias do mecanismo	Observações
Aurora MySQL	Sem versões secundárias	Nenhuma das versões secundárias do Aurora MySQL permite realizar transições ou failovers gerenciados entre regiões com níveis de patch diferentes entre os clusters de banco de dados primário e secundário.
Aurora PostgreSQL	• Versão 15 ou versões principais posteriores • Versão 14.5 ou versões secundárias posteriores	Com as versões do mecanismo listadas na coluna anterior, é possível realizar transições ou failovers gerenciados entre regiões de um cluster de banco de dados

Mecanismo do banco de dados	Versões secundárias do mecanismo	Observações
	• Versão 13.8 ou versões secundárias posteriores • Versão 12.12 ou versões secundárias posteriores • Versão 11.17 ou versões secundárias posteriores	primário com um nível de patch para um cluster de banco de dados secundário com um nível de patch diferente. Com todas as versões secundárias anteriores a essas, só será possível realizar transições ou failovers gerenciados entre regiões se os níveis de patch dos clusters de banco de dados primário e secundário forem os mesmos. **Warning** Ao atualizar um cluster em seu banco de dados global para qualquer uma das versões de patch a seguir, você não poderá realizar transições ou failovers entre regiões enquanto todos os clusters no banco de dados global não estiverem executando uma dessas versões de patch ou uma mais recente. • Versões de patch 16.1.6, 16.2.4, 16.3.2 e 16.4.2 • Versões de patch 15.3.8, 15.4.9, 15.5.6, 15.6.4, 15.7.2 e 15.8.2

Mecanismo do banco de dados	Versões secundárias do mecanismo	Observações
		• Versões de patch 14.8.8, 14.9.9, 14.10.6, 14.11.4, 14.12.2 e 14.13.2

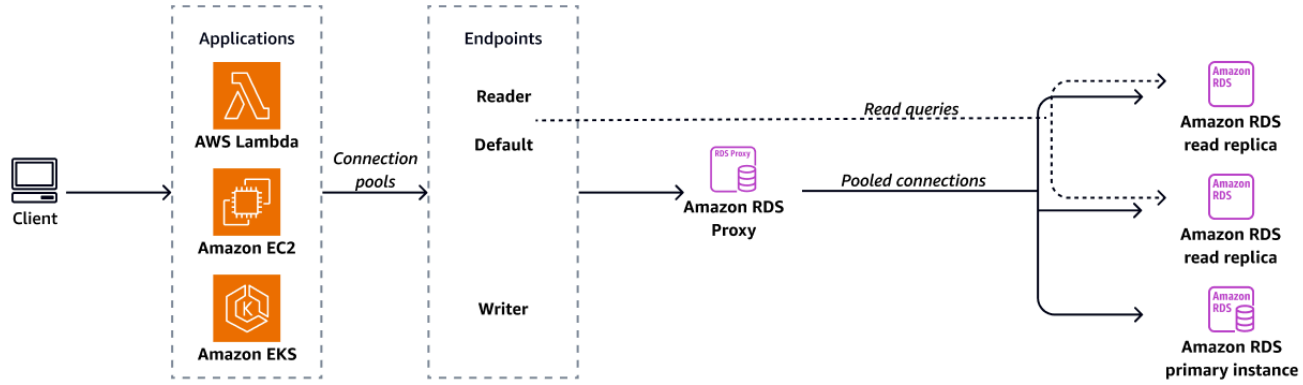
Amazon RDS Proxypara Aurora

Com o proxy do Amazon RDS, você pode permitir que suas aplicações agrupem e compartilhem conexões de banco de dados para melhorar sua capacidade de escala. O RDS Proxy torna as aplicações mais resilientes a falhas de banco de dados conectando-se automaticamente a uma instância de banco de dados em espera e preservando as conexões de aplicações. Ao usar o RDS Proxy, você pode aplicar a autenticação do AWS Identity and Access Management (IAM) para clientes que se conectam ao proxy, e o proxy pode se conectar aos bancos de dados usando a autenticação do banco de dados do IAM ou as credenciais armazenadas no AWS Secrets Manager.



Com o RDS Proxy, você pode lidar com picos imprevisíveis no tráfego de banco de dados. Caso contrário, esses picos podem causar problemas devido a conexões com excesso de assinaturas ou à criação rápida de conexões. O RDS Proxy estabelece um grupo de conexões de banco de dados e reutiliza conexões nesse grupo. Essa abordagem evita sobrecarregar a memória e a CPU de abrir uma nova conexão de banco de dados todas as vezes. Para proteger um banco de dados contra o excesso de assinaturas, é possível controlar o número de conexões do banco de dados criadas.

O RDS Proxy coloca na fila ou limita as conexões das aplicações que não podem ser atendidas imediatamente do grupo de conexões. Embora as latências possam aumentar, sua aplicação pode continuar a escalar sem falhar abruptamente ou sobrecarregar o banco de dados. Se as solicitações de conexão excederem os limites especificados, o proxy do RDS rejeitará as conexões de aplicações (ou seja, grandes quantidades de carga). Ao mesmo tempo, ele mantém uma performance previsível para a carga que pode ser atendida pelo RDS com a capacidade disponível.



Você pode reduzir a sobrecarga para processar credenciais e estabelecer uma conexão segura para cada nova conexão. O proxy do RDS pode lidar com parte desse trabalho em nome do banco de dados.

O proxy do RDS é totalmente compatível com as versões dos mecanismos com os quais é compatível. É possível habilitar o proxy do RDS para a maioria das aplicações sem alterações de código. Para obter uma lista das versões dos mecanismos compatíveis, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Amazon RDS Proxy.

Tópicos

- Disponibilidade de região e versão
- Cotas e limitações do RDS Proxy
- Planejar onde usar o RDS Proxy
- Conceitos e terminologia do RDS Proxy
- Conceitos básicos do RDS Proxy
- Gerenciar um RDS Proxy
- Como trabalhar com endpoints do proxy do Amazon RDS
- Monitorar métricas do proxy do RDS com o Amazon CloudWatch
- Trabalhar com eventos do RDS Proxy
- Exemplos de linha de comando do RDS Proxy
- Solução de problemas do RDS Proxy
- Usar o proxy do RDS com o AWS CloudFormation
- Usa o RDS Proxy com bancos de dados globais do Aurora

Disponibilidade de região e versão

Para saber mais sobre o suporte a versões de mecanismos de banco de dados e a disponibilidade do RDS Proxy em determinada região da Região da AWS, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Amazon RDS Proxy.

Cotas e limitações do RDS Proxy

As seguintes limitações aplicam-se ao RDS Proxy:

- Cada ID de Conta da AWS é limitado a 20 proxies. Se a aplicação exigir mais proxies, solicite um aumento na página Service Quotas no Console de gerenciamento da AWS. Na página Service Quotas, selecione Amazon Relational Database Service (Amazon RDS) e localize Proxies para solicitar um aumento de cota. A AWS pode aumentar automaticamente a cota ou aguardar a análise da solicitação pelo Suporte.
- Cada proxy pode ter até duzentos segredos do Secrets Manager. Assim, as conexões podem ser limitadas a até duzentas contas de usuário diferentes ao usar segredos.
- Cada proxy tem um endpoint padrão que é provisionado em apenas duas zonas de disponibilidade selecionadas nas sub-redes configuradas do proxy. Se houver sub-redes configuradas em mais de duas AZs, quaisquer duas serão selecionadas.
- É possível acrescentar até vinte endpoints de proxy adicionais a cada proxy. Esses endpoints adicionais são provisionados em todas as zonas de disponibilidade especificadas durante a respectiva criação. É possível criar, visualizar, modificar e excluir esses endpoints.
- Em um cluster do Aurora, todas as conexões que usam o endpoint padrão do proxy são tratadas pela instância de gravação do Aurora. Para executar o balanceamento de carga para workloads com uso intensivo de leitura, você pode criar um endpoint somente leitura para um proxy. Esse endpoint passa conexões para o endpoint de leitor do cluster. Dessa forma, suas conexões de proxy podem tirar proveito da escalabilidade de leitura do Aurora. Para ter mais informações, consulte Visão geral dos endpoints de proxy.
- É possível usar o RDS Proxy com clusters do Aurora Serverless v2, mas não com clusters do Aurora Serverless v1.
- O proxy do RDS deve estar na mesma nuvem privada virtual (VPC) que o banco de dados. O proxy não pode ser acessível publicamente, embora o banco de dados sim. Por exemplo, se você estiver fazendo protótipos do banco de dados em um host local, não poderá se conectar ao proxy, a menos que você configure os requisitos de rede necessários para permitir a conexão com o proxy. O motivo disso é que o host local está fora da VPC do proxy.

> **Note**
>
> Para clusters de bancos de dados Aurora, é possível habilitar o acesso entre VPCs. Para isso, crie um endpoint adicional para um proxy e especifique uma VPC, sub-redes e grupos de segurança diferente com esse endpoint. Para ter mais informações, consulte Acessar bancos de dados Aurora entre VPCs.

- Não é possível usar o proxy do RDS com uma VPC cuja locação está definida como `dedicated`.
- Não é possível usar o RDS Proxy em uma VPC que tenha controles de criptografia com o `Enforce Mode` habilitado.
- Para os tipos de rede de endpoint IPv6, configure sua VPC e sub-redes para comportar apenas IPv6. Para os tipos de rede de conexão de destino IPv4 e IPv6, configure sua VPC e sub-redes para comportar o modo de pilha dupla.
- Se você usar o RDS Proxy com um cluster de banco de dados do Aurora com a autenticação do IAM habilitada, o proxy poderá se conectar ao banco de dados usando a autenticação do IAM ou com as credenciais armazenadas no Secrets Manager. Os clientes que se conectam ao proxy devem se autenticar usando as credenciais do IAM. Para receber instruções detalhadas de configuração, consulte Configurar credenciais de banco de dados para RDS Proxy e Configurar a autenticação do IAM para o RDS Proxy.
- Você não pode usar o RDS Proxy com DNS personalizado ao usar a validação do nome de host SSL.
- Cada proxy pode ser associado a um único cluster de banco de dados de destino. No entanto, é possível associar vários proxies ao mesmo cluster de banco de dados.
- Qualquer instrução com um tamanho de texto maior do que 16 KB faz com que o proxy fixe a sessão à conexão atual.
- Certas regiões têm restrições de zona de disponibilidade (AZ) a serem consideradas ao criar seu proxy. A região Leste dos EUA (Norte da Virgínia) não é compatível com o RDS Proxy na zona de `use1-az3` disponibilidade. A região Oeste dos EUA (Norte da Califórnia) não é compatível com o RDS Proxy na zona de `usw1-az2` disponibilidade. Ao selecionar sub-redes ao criar seu proxy, certifique-se de não selecionar sub-redes nas zonas de disponibilidade mencionadas acima.
- No momento, o RDS Proxy não comporta nenhuma chave de contexto de condição global.

  Para obter mais informações sobre chaves de contexto de condição global, consulte Chaves de contexto de condição global AWS no Guia do usuário do IAM.

- Não é possível usar o RDS Proxy com o RDS Custom para SQL Server.
- Para refletir qualquer modificação do grupo de parâmetros do banco de dados no proxy, é necessária uma reinicialização da instância, mesmo que você opte por aplicar as alterações imediatamente. Com relação a parâmetros no nível do cluster, é necessária uma reinicialização em todo o cluster.
- Seu proxy cria automaticamente o usuário do banco de dados `rdsproxyadmin` quando você registra um destino de proxy. Este é um usuário protegido que é essencial para a funcionalidade de proxy. Você deve evitar alterar o usuário `rdsproxyadmin` em qualquer função. A exclusão ou modificação do usuário `rdsproxyadmin` ou das respectivas permissões pode afetar a disponibilidade do proxy para sua aplicação.

Para saber as limitações adicionais para cada mecanismo de banco de dados, consulte as seguintes seções:

- Limitações adicionais do Aurora MySQL
- Limitações adicionais do Aurora PostgreSQL

Limitações adicionais do Aurora MySQL

As seguintes limitações adicionais se aplicam ao RDS Proxy com bancos de dados do Aurora MySQL:

- O suporte do RDS Proxy para autenticação `caching_sha2_password` requer uma conexão segura (TLS).
- Sabe-se que o suporte do RDS Proxy para `caching_sha2_password` tem problemas de compatibilidade com determinadas versões do driver go-sql.
- Ao usar o driver MySQL 8.4 C, a API `mysql_stmt_bind_named_param` poderá formar pacotes malformados se a contagem de parâmetros exceder a contagem de espaços reservados em instruções preparadas. Isso resulta em respostas incorretas. Para obter mais informações, consulte Relatório de bugs do MySQL.
- No momento, todos os proxies escutam na porta 3306 para o MySQL. Os proxies ainda se conectam ao banco de dados usando a porta especificada nas configurações do banco de dados.
- Não é possível usar o proxy do RDS com bancos de dados MySQL autogerenciados em instâncias do EC2.

- Você não pode usar o proxy do RDS com uma instância de banco de dados do RDS para MySQL que tenha o parâmetro `read_only` em seu grupo de parâmetros de banco de dados definido como `1`.
- O proxy do RDS não é compatível com o modo compactado do MySQL. Por exemplo, ele não é compatível com a compactação usada pelas opções `--compress` ou `-C` do comando `mysql`.
- Conexões de banco de dados que processam um comando `GET DIAGNOSTIC` podem retornar informações imprecisas quando o RDS Proxy reutiliza a mesma conexão de banco de dados para executar outra consulta. Isso pode acontecer quando o RDS Proxy realiza multiplexação de conexões de banco de dados.
- Algumas funções e declarações SQL, como `SET LOCAL`, podem alterar o estado da conexão sem causar fixação. Para obter o comportamento de fixação mais atual, consulte Evitar a fixação de um RDS Proxy.
- Usar a função `ROW_COUNT()` em uma consulta com várias declarações não é aceito.
- O RDS Proxy não comporta aplicações clientes que não conseguem lidar com várias mensagens de resposta em um registro do TLS.
- O RDS Proxy não aceita senhas duplas do MySQL.
- O RDS Proxy pode não funcionar conforme o esperado quando você configura o parâmetro `init_connect` no grupo de parâmetros do banco de dados do RDS para definir variáveis de estado da sessão. Em vez disso, defina a consulta de inicialização para que o proxy execute as instruções de inicialização da sessão ao usar o proxy para se conectar ao seu banco de dados.

> **Important**
>
> Para proxies associados a bancos de dados MySQL, não defina o parâmetro de configuração `sql_auto_is_null` como `true` ou um valor diferente de zero na consulta de inicialização. Isso pode causar um comportamento incorreto da aplicação.

Limitações adicionais do Aurora PostgreSQL

As seguintes limitações adicionais se aplicam ao RDS Proxy com bancos de dados do Aurora PostgreSQL:

- O proxy do RDS não é compatível com os filtros de fixação de sessão do PostgreSQL.
- Atualmente, todos os proxies escutam na porta 5432 para o PostgreSQL.

- No PostgreSQL, o proxy do RDS atualmente não é compatível com o cancelamento de uma consulta de um cliente emitindo um `CancelRequest`. Esse é o caso, por exemplo, quando você cancela uma consulta de longa duração em uma sessão psql interativa usando Ctrl+C.
- Os resultados da função do PostgreSQL lastval nem sempre são precisos. Como uma solução alternativa, use a instrução INSERT com a cláusula `RETURNING`.
- Atualmente, o RDS Proxy não é compatível com o modo de replicação de streaming.
- O banco de dados `postgres` padrão deve existir na instância do RDS para PostgreSQL para que o RDS Proxy funcione. Não exclua esse banco de dados, mesmo que sua aplicação use bancos de dados diferentes.
- Se você usar `ALTER ROLE` para alterar o perfil do usuário com `SET ROLE`, as conexões subsequentes desse usuário com o proxy talvez não usem essa configuração de perfil, se essas conexões encontrarem fixação. Para evitar isso, ao usar o proxy, use `SET ROLE` na consulta de inicialização do proxy. Para obter mais informações, consulte Consulta de inicialização no Criar um proxy para o Amazon Aurora.

> **Important**
>
> Para proxies existentes com bancos de dados PostgreSQL, se você modificar a autenticação do banco de dados para usar apenas `SCRAM`, o proxy ficará indisponível por até 60 segundos. Para evitar o problema, faça o seguinte:
>
> - O banco de dados deve permitir tanto a autenticação `SCRAM` quanto a `MD5`.
> - Para usar somente a autenticação `SCRAM`, crie um proxy, migre o tráfego da aplicação para o novo proxy e exclua o proxy anteriormente associado ao banco de dados.

Planejar onde usar o RDS Proxy

Você pode determinar quais de suas instâncias de banco de dados, clusters e aplicações podem se beneficiar mais com o uso do RDS Proxy. Para fazer isso, considere estes fatores:

- Qualquer cluster de banco de dados que encontrar erros de “conexões em excesso” é um bom candidato para associação a um proxy. Isso geralmente é caracterizado por um alto valor da métrica `ConnectionAttempts` do CloudWatch. O proxy permite que as aplicações abram muitas conexões de cliente enquanto gerencia um número menor de conexões de longa duração para o cluster de banco de dados.

- Para clusters de bancos de dados que usam classes de instância menores da AWS, como T2 ou T3, o uso de um proxy pode ajudar a evitar condições de falta de memória. Isso também pode ajudar a reduzir a sobrecarga da CPU para estabelecer conexões. Essas condições podem ocorrer ao lidar com um grande número de conexões.
- É possível monitorar certas métricas do Amazon CloudWatch para determinar se um cluster de banco de dados está se aproximando de certos tipos de limite. Esses limites são para o número de conexões e para a memória associada ao gerenciamento de conexão. Você também pode monitorar certas métricas do CloudWatch para determinar se um cluster de banco de dados está lidando com muitas conexões de curta duração. A abertura e o fechamento dessas conexões podem aplicar um sobrecarga de performance a seu banco de dados. Para obter informações sobre as métricas a serem monitoradas, consulte Monitorar métricas do proxy do RDS com o Amazon CloudWatch.
- AWS LambdaAs funções do também podem ser boas candidatas para o uso de um proxy. Essas funções fazem conexões curtas frequentes com o banco de dados que se beneficiam do grupo de conexões oferecido pelo RDS Proxy. É possível aproveitar qualquer autenticação do IAM que você já tenha para funções do Lambda, em vez de gerenciar credenciais de banco de dados em seu código de aplicação do Lambda.
- Essas aplicações que geralmente abrem e encerram um grande número de conexões de banco de dados e não têm mecanismos internos de agrupamento de conexões são ideais para usar um proxy.
- As aplicações que mantêm um grande número de conexões abertas por longos períodos geralmente são boas candidatas para o uso de um proxy. As aplicações em setores como software como serviço (SaaS) ou comércio eletrônico geralmente minimizam a latência de solicitações de banco de dados deixando as conexões abertas. Com o RDS Proxy, uma aplicação pode manter mais conexões abertas do que quando se conecta diretamente ao cluster de banco de dados.
- Talvez você não tenha adotado a autenticação do IAM e do Secrets Manager devido à complexidade da configuração dessa autenticação para todos os clusters de banco de dados. O proxy pode aplicar as políticas de autenticação para conexões de cliente para aplicações específicas. É possível aproveitar qualquer autenticação do IAM que você já tenha para funções do Lambda, em vez de gerenciar credenciais de banco de dados em seu código de aplicação do Lambda.
- O RDS Proxy pode ajudar a tornar as aplicações mais resilientes e transparentes às falhas do banco de dados. Ele ignora os caches do Sistema de Nomes de Domínio (DNS) para reduzir os tempos de failover em até 66% para bancos de dados multi-AZ do Aurora. O RDS Proxy também

direciona automaticamente o tráfego para uma nova instância de banco de dados, preservando as conexões da aplicação. Isso torna os failovers mais transparentes para as aplicações.

Conceitos e terminologia do RDS Proxy

É possível simplificar o gerenciamento das conexões das instâncias de banco de dados do Amazon RDS e de clusters de bancos de dados Amazon Aurora usando o RDS Proxy.

O proxy do RDS lida com o tráfego de rede entre a aplicação cliente e o banco de dados. Ele faz isso de uma maneira ativa primeiro, entendendo o protocolo de banco de dados. Depois, ele ajusta seu comportamento com base nas operações SQL da aplicação e nos conjuntos de resultados do banco de dados.

O proxy do RDS reduz a sobrecarga de memória e de CPU para gerenciamento de conexão no banco de dados. O banco de dados precisa de menos memória e recursos de CPU quando as aplicações abrem muitas conexões simultâneas. Ele também não requer lógica em suas aplicações para fechar e reabrir conexões que permanecem ociosas por um longo período. Da mesma forma, ele requer menos lógica nas aplicações para restabelecer conexões em caso de um problema de banco de dados.

A infraestrutura do proxy do RDS é altamente disponível e implantada em várias zonas de disponibilidade (AZs). A computação, a memória e o armazenamento do RDS Proxy são independentes do cluster de banco de dados do Aurora. Essa separação ajuda a reduzir a sobrecarga em seus servidores de banco de dados, para que eles possam dedicar seus recursos para atender às workloads do banco de dados. Os recursos de computação do proxy do RDS não têm servidor, realizando a escalabilidade automática com base na workload do banco de dados.

Tópicos

- Visão geral dos conceitos do RDS Proxy
- Agrupamento de conexões
- Segurança do RDS Proxy
- Failover
- Transações

Visão geral dos conceitos do RDS Proxy

O proxy do RDS lida com a infraestrutura para executar o agrupamento de conexões e os outros recursos descritos nas seções a seguir. Os proxies estão representados no console do RDS na página Proxies.

Cada proxy lida com conexões com um único cluster de bancos de dados do Aurora. O proxy determina automaticamente a instância de gravador atual para as instâncias de banco de dados multi-AZ do RDS e para os clusters provisionados do Aurora.

As conexões que um proxy mantém abertas e disponíveis para a aplicação de banco de dados usar compõem o grupo de conexões.

Por padrão, o proxy do RDS pode reutilizar uma conexão após cada transação em sua sessão. Essa reutilização em nível de transação é chamada de multiplexação. Quando o proxy do RDS remove temporariamente uma conexão do grupo de conexões para reutilizá-la, essa operação é chamada de empréstimo da conexão. Quando é seguro fazer isso, o proxy do RDS retorna essa conexão ao grupo de conexões.

Em alguns casos, o proxy do RDS não consegue ter certeza de que é seguro reutilizar uma conexão de banco de dados fora da sessão atual. Nesses casos, ele mantém a sessão na mesma conexão até que a sessão termine. Esse comportamento de fallback é chamado de fixação.

Um proxy tem um endpoint padrão. Você se conecta a esse endpoint ao trabalhar com uma instância de banco de dados do RDS ou com um cluster de banco de dados do Aurora. Você faz isso em vez de se conectar ao endpoint de leitura/gravação que se conecta diretamente à instância ou ao cluster. Os endpoints de finalidade especial de um cluster do Aurora permanecem disponíveis para uso. Para clusters de bancos de dados do Aurora, também é possível criar endpoints adicionais de leitura/gravação e somente leitura. Para obter mais informações, consulte Visão geral dos endpoints de proxy.

Por exemplo, você ainda pode se conectar ao endpoint do cluster para conexões de leitura/gravação sem agrupamento de conexões. Você ainda pode se conectar ao endpoint de leitor para conexões somente leitura com balanceamento de carga. Você ainda pode se conectar aos endpoints da instância para diagnóstico e solução de problemas de instâncias de banco de dados específicas em um cluster do . Se você usa outros serviços da AWS, como o AWS Lambda para se conectar a bancos de dados do RDS, altere as configurações de conexão para usar o endpoint do proxy. Por exemplo, você especifica o endpoint do proxy para permitir que as funções do Lambda acessem seu banco de dados enquanto aproveita a funcionalidade do RDS Proxy.

Cada proxy contém um grupo de destino. Esse grupo de destino incorpora o cluster de bancos de dados do Aurora ao qual o proxy pode se conectar. Para um cluster do Aurora, por padrão, o grupo de destino é associado a todas as instâncias de banco de dados nesse cluster. Dessa forma, o proxy pode se conectar a qualquer instância de bancos de dados Aurora promovida para ser a instância de gravação no cluster. O cluster de banco de dados do Aurora associado a um proxy é chamado de destino desse proxy. Por conveniência, quando você cria um proxy pelo console, o proxy do RDS também cria o grupo de destino correspondente e registra os destinos associados automaticamente.

Uma família de mecanismos é um conjunto relacionado de mecanismos de banco de dados que usam o mesmo protocolo de banco de dados. Escolha a família de mecanismos para cada proxy criado.

Agrupamento de conexões

Cada proxy executa o agrupamento de conexões separadamente para a instância de gravação e leitura de seu ou do Aurora associado. O agrupamento de conexões é uma otimização que reduz a sobrecarga associada à abertura e ao fechamento de conexões e à manutenção de muitas conexões abertas simultaneamente. Essa sobrecarga inclui a memória necessária para lidar com cada nova conexão. Também envolve a sobrecarga da CPU para fechar cada conexão e abrir uma nova. Os exemplos incluem handshaking TLS/SSL (Transport Layer Security/Secure Sockets Layer), autenticação, recursos de negociação etc. O agrupamento de conexões simplifica a lógica da aplicação. Você não precisa escrever código da aplicação para minimizar o número de conexões abertas simultâneas.

Cada proxy também executa a multiplexação de conexões, também conhecida como reutilização de conexões. Com a multiplexação, o RDS Proxy executa todas as operações para uma transação usando uma conexão de banco de dados subjacente. Depois, o RDS pode usar uma conexão diferente para a próxima transação. Você pode abrir muitas conexões simultâneas com o proxy, e o proxy mantém um número menor de conexões abertas com a instância ou o cluster de banco de dados. Isso minimiza ainda mais a sobrecarga na memória de conexões no servidor de banco de dados. Essa técnica também reduz a possibilidade de erros de “excesso de conexões”.

Segurança do RDS Proxy

O proxy do RDS usa os mecanismos de segurança do RDS existentes, como TLS/SSL e o AWS Identity and Access Management (IAM). Para obter informações gerais sobre esses recursos de segurança, consulte Segurança no Amazon Aurora. Além disso, familiarize-se com a forma como o RDS e o Aurora trabalham com autenticação, autorização e outras áreas de segurança.

O proxy do RDS pode atuar como uma camada adicional de segurança entre aplicações cliente e o banco de dados subjacente. Por exemplo, você pode se conectar ao proxy usando o TLS 1.3, mesmo que a instância de banco de dados subjacente ofereça suporte a uma versão mais antiga do TLS. Você pode se conectar ao proxy usando um perfil do IAM, mesmo que o proxy se conecte ao banco de dados usando o método de autenticação de usuário e senha do banco de dados. Usando essa técnica, você pode aplicar requisitos de autenticação fortes para aplicações de banco de dados sem um esforço de migração dispendioso para as próprias instâncias de banco de dados.

Você pode usar os seguintes métodos de autenticação com o RDS Proxy:

- Credenciais do banco de dados
- Autenticação padrão do IAM
- Autenticação de ponta a ponta do IAM

Usar o IAM com o RDS Proxy

O RDS Proxy oferece dois métodos de autenticação do IAM:

- Autenticação padrão do IAM: aplique a autenticação do IAM para conexões com seu proxy enquanto ele se conecta ao banco de dados usando credenciais armazenadas no Secrets Manager. Isso aplica a autenticação do IAM para acesso ao banco de dados mesmo que os bancos de dados usem autenticação nativa de senha. O proxy recupera as credenciais do banco de dados do Secrets Manager e gerencia a autenticação no banco de dados em nome da sua aplicação.
- Autenticação de ponta a ponta do IAM: aplica a autenticação do IAM para conexões diretamente de suas aplicações com o seu banco de dados por meio do proxy. A autenticação de ponta a ponta do IAM simplifica sua configuração de segurança e evita o gerenciamento de credenciais do banco de dados no Secrets Manager. Essa camada adicional de segurança aplica o controle de acesso baseado no IAM da aplicação cliente ao banco de dados.

Para usar a autenticação padrão do IAM, configure seu proxy para usar segredos do Secrets Manager para autenticação e habilite a autenticação do IAM para conexões de clientes. Suas aplicações fazem a autenticação no proxy usando o IAM, enquanto o proxy faz a autenticação no banco de dados usando as credenciais recuperadas do Secrets Manager.

Para usar a autenticação de ponta a ponta do IAM, configure seu proxy para usar a autenticação do IAM ao definir o esquema de autenticação padrão ao criar ou modificar seu proxy.

Para a autenticação de ponta a ponta do IAM, você deve atualizar o perfil do IAM associado ao proxy para conceder a permissão `rds-db:connect`. Com a autenticação de ponta a ponta do IAM, isso elimina a necessidade de registrar usuários individuais do banco de dados com o proxy por meio dos segredos do Secrets Manager.

Usar TLS/SSL com o RDS Proxy

Você pode se conectar ao proxy do RDS usando o protocolo TLS/SSL.

> **Note**
>
> O proxy do Amazon RDS usa certificados do AWS Certificate Manager (ACM). Se você estiver usando o RDS Proxy, não será necessário baixar os certificados do Amazon RDS ou atualizar as aplicações que usam conexões RDS Proxy.

Para aplicar o TLS a todas as conexões entre o proxy e o banco de dados, é possível especificar uma configuração Exigir Transport Layer Security ao criar ou modificar um proxy no Console de gerenciamento da AWS.

O proxy do RDS também pode garantir que a sessão use TLS/SSL entre o cliente e o endpoint do RDS Proxy. Para que o proxy do RDS faça isso, especifique o requisito no lado do cliente. As variáveis de sessão SSL não são definidas para conexões SSL a um banco de dados usando o RDS Proxy.

- Para o Aurora MySQL, especifique o requisito no lado do cliente com o parâmetro `--ssl-mode` ao executar o comando `mysql`.
- No e no Aurora PostgreSQL, especifique `sslmode=require` como parte da string `conninfo` ao executar o comando `psql`.

O RDS Proxy é compatível com o protocolo TLS versões 1.0, 1.1, 1.2 e 1.3. É possível se conectar ao proxy usando uma versão mais recente do TLS do que a usada no banco de dados subjacente.

Por padrão, os programas cliente estabelecem uma conexão criptografada com o RDS Proxy, com controle adicional disponível por meio da opção `--ssl-mode`. No lado do cliente, o proxy do RDS oferece suporte a todos os modos SSL.

Para o cliente, os modos SSL são os seguintes:

PREFERRED

O SSL é a primeira opção, mas não é obrigatória.

DISABLED

Nenhum SSL é permitido.

REQUIRED

Impor SSL.

VERIFY_CA

Impor SSL e verificar a autoridade de certificação (CA - certificate authority).

VERIFY_IDENTITY

Impor SSL e verificar a CA, além do nome de host da CA.

Ao usar um cliente com `--ssl-mode VERIFY_CA` ou `VERIFY_IDENTITY`, especifique a opção `--ssl-ca` apontando para uma CA em formato `.pem`. Para o arquivo `.pem` a ser usado, baixe todos os PEMs de CA raiz de Amazon Trust Services e coloque-os em um único arquivo `.pem`.

O RDS Proxy usa certificados curinga, que se aplicam a um domínio e aos seus subdomínios. No momento, se você usa o cliente `mysql` para se conectar ao modo SSL `VERIFY_IDENTITY`, é necessário usar o comando `mysql` compatível com MySQL 8.0.

Failover

O failover é um recurso de alta disponibilidade que substitui uma instância de banco de dados por outra quando a instância original fica indisponível. Um failover pode ocorrer devido a um problema com uma instância de banco de dados. Ele também pode fazer parte de procedimentos normais de manutenção, como durante uma atualização de banco de dados. e a clusters de bancos de dados do Aurora com uma ou mais instâncias de leitor além da instância de gravador.

A conexão por meio de um proxy torna a aplicação mais resiliente a failovers de banco de dados. Quando a instância de banco de dados original se torna indisponível, o proxy do RDS conecta-se ao banco de dados em espera sem eliminar as conexões de aplicações ociosas. Isso ajuda a acelerar e simplificar o processo de failover. Isso causa menos interrupções na aplicação do que uma reinicialização típica ou um problema de banco de dados.

Sem o RDS Proxy, um failover envolve uma breve interrupção. Durante a interrupção, não é possível executar operações de gravação no banco de dados em failover. Todas as conexões de banco de dados existentes são interrompidas e sua aplicação precisa reabri-las. O banco de dados fica disponível para novas conexões e operações de gravação quando uma instância de banco de dados somente leitura é promovida para substituir a que não está disponível.

Durante os failovers de banco de dados, o proxy do RDS continua aceitando conexões no mesmo endereço IP e direciona conexões automaticamente para a nova instância do banco de dados primário. Os clientes que se conectam pelo proxy do RDS não são suscetíveis ao seguinte:

- Atrasos de propagação do Domain Name System (DNS) no failover.
- Cache DNS local.
- Tempos limite de conexão.
- Incerteza sobre qual instância de banco de dados é o gravador atual.
- Espera pela resposta de uma consulta de um gravador anterior que ficou indisponível sem fechar conexões.

Para aplicações que mantêm seu próprio grupo de conexões, passar pelo proxy do RDS significa que a maioria das conexões permanecem ativas durante failovers ou outras interrupções. Somente conexões que estão no meio de uma transação ou de uma instrução SQL são canceladas. O proxy do RDS aceita novas conexões imediatamente. Quando o gravador do banco de dados não estiver disponível, o proxy do RDS coloca as solicitações de entrada em fila.

Para aplicações que não mantêm seus próprios grupos de conexão, o proxy do RDS oferece taxas de conexão mais rápidas e conexões mais abertas. Ele descarrega a sobrecarga cara de reconexões frequentes do banco de dados. E faz isso reutilizando as conexões de banco de dados mantidas no grupo de conexões do RDS Proxy. Essa abordagem é particularmente importante para conexões TLS, em que os custos da configuração são significativos.

Transações

Todas as instruções dentro de uma única transação sempre usam a mesma conexão ao banco de dados subjacente. A conexão se torna disponível para uso de outra sessão quando a transação termina. O uso da transação como a unidade de granularidade tem as seguintes consequências:

- A reutilização da conexão pode ocorrer após cada declaração individual, quando a configuração `autocommit` do Aurora MySQL estiver ativada.

- Por outro lado, quando a configuração `autocommit` está desativada, a primeira declaração emitida em uma sessão inicia uma nova transação. Por exemplo, suponhamos que você insira uma sequência de `SELECT`, `INSERT`, `UPDATE` e outras declarações em linguagem de manipulação de dados (DML). Nesse caso, a reutilização da conexão não ocorrerá até que você emita um `COMMIT`, `ROLLBACK` ou encerre a transação.
- A inserção de uma instrução DDL (Data Definition Language, linguagem de definição de dados) faz com que a transação termine depois que essa instrução é concluída.

O proxy do RDS detecta quando uma transação termina por meio do protocolo de rede usado pela aplicação cliente de banco de dados. A detecção da transação não depende de palavras-chave, como `COMMIT` ou `ROLLBACK`, que aparecem no texto da instrução SQL.

Em alguns casos, o proxy do RDS pode detectar uma solicitação de banco de dados que torna impraticável mover a sessão para uma conexão diferente. Nesses casos, ele desativa a multiplexação para essa conexão o restante da sessão. A mesma regra se aplicará se o proxy do RDS não puder ter certeza de que a multiplexação é prática para a sessão. Essa operação é chamada de fixação. Para obter formas de detectar e minimizar a fixação, consulte Evitar a fixação de um RDS Proxy.

Conceitos básicos do RDS Proxy

Use as informações nas páginas a seguir para configurar, gerenciar Amazon RDS Proxypara Aurora e definir opções de segurança relacionadas. As opções de segurança controlam quem pode acessar cada proxy e como cada proxy se conecta a instâncias de banco de dados.

Se você é novo no RDS Proxy, recomendamos seguir as páginas na ordem em que as apresentamos.

Tópicos

- Configurar os pré-requisitos de rede para o RDS Proxy
- Configurar credenciais de banco de dados para RDS Proxy
- Configurar a autenticação do IAM para o RDS Proxy
- Criar um proxy para o Amazon Aurora
- Visualizar um proxy
- Conectar-se a um banco de dados pelo RDS Proxy

Configurar os pré-requisitos de rede para o RDS Proxy

O uso do RDS Proxy requer que você tenha uma nuvem privada virtual (VPC) comum entre o cluster de banco de dados do Aurora e o RDS Proxy. Essa VPC deve ter um mínimo de duas sub-redes em zonas de disponibilidade diferentes. Sua conta pode possuir essas sub-redes ou compartilhá-las com outras contas. Para obter informações sobre compartilhamento de VPC, consulte Trabalhar com VPCs compartilhadas.

Para suporte a IPv6, é necessária uma configuração de rede adicional:

- Tipos de rede de endpoint IPv6: sua VPC e sub-redes devem estar configuradas para comportar IPv6. Isso inclui ter blocos CIDR IPv6 atribuídos à VPC e às sub-redes.
- Tipos de rede de endpoint de pilha dupla: sua VPC e sub-redes devem comportar endereçamento IPv4 e IPv6.
- Tipos de rede de conexão de destino IPv6: seu banco de dados deve estar configurado para o modo de pilha dupla a fim de comportar conexões IPv6 do proxy.

Seus recursos de aplicações cliente, como Amazon EC2, Lambda ou Amazon ECS, podem estar na mesma VPC que o proxy. Ou podem estar em uma VPC separada do proxy. Se você se conectou com êxito a quaisquer clusters de banco de dados do Aurora, você já terá os recursos de rede necessários.

Tópicos

- Obter informações sobre suas sub-redes
- Planejar a capacidade de endereços IP

Obter informações sobre suas sub-redes

Se você está apenas começando a usar o Aurora, aprenda o básico sobre a conexão com um banco de dados seguindo os procedimentos em Configuração de seu ambiente para Amazon Aurora. Você também pode seguir o tutorial em Conceitos básicos do Amazon Aurora.

Para criar um proxy, é necessário fornecer as sub-redes e a VPC em que o proxy opera. O exemplo do Linux a seguir mostra comandos da AWS CLI que examinam as VPCs e as sub-redes pertencentes à sua Conta da AWS. Em particular, você passa IDs de sub-rede como parâmetros quando cria um proxy usando o CLI.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

O exemplo do Linux a seguir mostra comandos da AWS CLI para determinar os IDs de sub-rede correspondentes a um cluster de banco de dados específico do Aurora.

Para um cluster do Aurora, primeiro você encontra o ID de uma das instâncias de banco de dados associadas. Você pode extrair os IDs de sub-rede usados por essa instância de banco de dados. Para isso, examine os campos aninhados nos atributos `DBSubnetGroup` e `Subnets` na saída de descrição da instância de banco de dados. Você especifica alguns ou todos esses IDs de sub-rede ao configurar um proxy para esse servidor de banco de dados.

```
$ # Find the ID of any DB instance in the cluster.
$  aws rds describe-db-clusters --db-cluster-identifier my_cluster_id --query '*[].
[DBClusterMembers]||[0]||[0][*].DBInstanceIdentifier' --output text
```

```
my_instance_id
instance_id_2
instance_id_3
```

Depois de encontrar o identificador da instância de banco de dados, examine a VPC associada para encontrar suas sub-redes. O exemplo do Linux a seguir mostra como fazer isso.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0]||[Subnets]||[0]||[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

Planejar a capacidade de endereços IP

O RDS Proxy ajusta automaticamente sua capacidade com base no tamanho na configuração das instâncias de banco de dados registradas nele. Em relação a instâncias provisionadas, isso é determinado pelo tamanho da instância e, no caso de instâncias do Aurora Serverless v2, pela capacidade máxima da ACU. Algumas operações também podem precisar de capacidade adicional de proxy, como aumentar o tamanho de um banco de dados registrado ou operações internas de manutenção do RDS Proxy. Durante essas operações, seu proxy pode precisar de mais endereços IP para provisionar a capacidade extra. Esses endereços adicionais possibilitam que seu proxy se expanda sem afetar a workload. A falta de endereços IP gratuitos em suas sub-redes impede que um proxy aumente a escala verticalmente. Isso pode ocasionar maiores latências de consulta ou falhas na conexão do cliente. O RDS notifica você por meio de um evento `RDS-EVENT-0243` quando não há endereços IP livres suficientes em suas sub-redes. Para obter informações sobre esse evento, consulte Trabalhar com eventos do RDS Proxy.

Reserve os números mínimos a seguir de endereços IP livres nas sub-redes para o proxy, com base no tamanho das classes de instâncias de banco de dados.

DB instance class	Mínimo de endereços IP livres
db.*.xlarge ou menor	10
db.*.2xlarge	15
db.*.4xlarge	25

DB instance class	Mínimo de endereços IP livres
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

Para Aurora Serverless v2, reserve os números mínimos a seguir de endereços IP livres nas sub-redes para o proxy, com base na capacidade máxima de ACU.

`Maximum ACU Capacity`	Mínimo de endereços IP livres
16 ou menor	10
32	15
64	25
96	30
128	40
160	50
192	55
224	65
256	75

> **Note**
>
> O RDS Proxy não consome mais de 215 endereços IP para cada proxy em uma VPC.

O RDS Proxy exige um mínimo de dez endereços IP para seu banco de dados Aurora. Esses números de endereços IP recomendados são estimativas para um proxy com apenas o endpoint padrão. Para cada endpoint personalizado adicional, recomendamos reservar mais três endereços IP. Para cada instância do leitor do Aurora, recomendamos que você reserve endereços IP adicionais conforme especificado na tabela com base nas ACUs máximas desse leitor para o destino do Aurora Serverless v2 ou o tamanho da instância de banco de dados para um destino provisionado.

Para estimar os endereços IP necessários para um proxy associado a um cluster de banco de dados do Aurora com:

- Uma instância de gravador provisionada de tamanho `db.r5.8xlarge` e uma instância de leitor provisionada de tamanho `db.r5.2xlarge`.
- O proxy anexado a esse cluster tem um endpoint padrão e um endpoint personalizado com a função de somente leitura.

Nesse caso, o proxy precisa de aproximadamente 63 endereços IP livres (45 para a instância do gravador, 15 para a instância do leitor e três para o endpoint personalizado adicional).

Para estimar os endereços IP necessários para um proxy associado a um cluster de banco de dados do Aurora que tenha:

- Uma instância de gravador do Aurora Serverless v2 com capacidade máxima de 256 ACUs e uma instância de leitor do v2 sem servidor com capacidade máxima de 192 ACUs.
- O proxy anexado a esse cluster tem o endpoint padrão e um endpoint personalizado com o perfil de somente leitura.

Nesse caso, o proxy precisa de aproximadamente 133 endereços IP livres (75 para a instância do gravador, 55 para a instância do leitor e 3 para o endpoint personalizado adicional).

Para estimar os endereços IP necessários para um proxy associado a um cluster do Aurora que tenha:

- Uma instância de gravador provisionada com tamanho de instância de banco de dados db.r5.4xlarge e uma instância de leitor do v2 sem servidor com capacidade máxima de 64 ACUs.
- O proxy anexado a esse cluster tem o endpoint padrão e um endpoint personalizado com o perfil de somente leitura.

Nesse caso, o proxy precisa de aproximadamente 53 endereços IP livres (25 para a instância do gravador, 25 para a instância do leitor e 3 para o endpoint personalizado adicional).

Para estimar os endereços IP necessários para um proxy associado a um cluster de banco de dados do Aurora que tenha:

- Uma instância de gravador provisionada de tamanho db.r5.24xlarge e três instâncias de leitor provisionadas de tamanho db.r5.8xlarge.
- O proxy conectado a esse cluster de banco de dados tem o endpoint padrão e um endpoint personalizado com a função de somente leitura.

Nesse caso, o proxy precisa de 215 endereços IP gratuitos. Embora os cálculos sugiram 248 IPs (110 + (3*45) + 3), o RDS Proxy não consome mais de 215 endereços IP para cada proxy em uma VPC.

Configurar credenciais de banco de dados para RDS Proxy

O RDS Proxy no Amazon RDS usa o AWS Secrets Manager para armazenar e gerenciar credenciais de banco de dados com segurança. Em vez de incorporar credenciais na aplicação, você associa um proxy a um segredo do Secrets Manager que contenha os detalhes de autenticação necessários. Crie um segredo separado do Secrets Manager para cada conta de usuário de banco de dados ao qual o proxy se conecta no cluster de bancos de dados do Aurora.

Você também pode configurar o RDS Proxy para usar a autenticação de ponta a ponta do IAM, o que elimina a necessidade de armazenar as credenciais do banco de dados no Secrets Manager. O RDS Proxy usa a autenticação do IAM para conexões de cliente para proxy e de proxy para banco de dados. Isso fornece uma solução de autenticação baseada no IAM totalmente integrada que não exige o gerenciamento de segredos ou senhas. Para acessar informações sobre como adicionar um novo usuário de banco de dados do IAM, consulte Criar uma conta de banco de dados usando autenticação do IAM.

Tópicos

- Criar segredos para usá-los com o RDS Proxy

Criar segredos para usá-los com o RDS Proxy

Antes de criar um proxy, você deve primeiro criar pelo menos um segredo que armazene suas credenciais de banco de dados.

Console

Como criar um segredo

1. Abra o console Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Armazenar Novo Segredo.
3. Escolha Credenciais para um banco de dados do Amazon RDS.
4. Insira um nome de usuário e senha. As credenciais que você inserir devem corresponder às credenciais de um usuário do banco de dados que exista no banco de dados do RDS correlacionado. O RDS Proxy usa essas credenciais para autenticar e estabelecer conexões com o banco de dados em nome das aplicações.

   Se houver alguma incompatibilidade, você poderá atualizar o segredo para que corresponda à senha do banco de dados. Enquanto você não atualizar o segredo, as tentativas de conexão por meio do proxy usando esse segredo falharão, mas as conexões que usam outros segredos válidos ainda funcionarão.

   > **Note**
   >
   > No caso do RDS para SQL Server, o RDS Proxy exige um segredo com distinção entre maiúsculas e minúsculas no Secrets Manager, independentemente das configurações de agrupamento da instância de banco de dados. Se a aplicação permitir nomes de usuário com diferentes usos de maiúsculas e minúsculas, como “Admin” e “admin”, você deverá criar segredos separados para cada um. O RDS Proxy não aceita autenticação de nome de usuário sem distinção entre maiúsculas e minúsculas entre o cliente e o proxy.
   > Para ter mais informações sobre agrupamento no SQL Server, consulte a documentação do Microsoft SQL Server.

5. Em Banco de dados, selecione o banco de dados do Amazon RDS que o segredo acessará.
6. Preencha outras configurações referentes ao segredo e escolha Armazenar. Para obter instruções mais abrangentes, consulte Creating an AWS Secrets Manager secret no Guia do usuário do AWS Secrets Manager.

AWS CLI

Ao criar um proxy por meio da AWS CLI, especifique os nomes de recurso da Amazon (ARNs) dos segredos correspondentes. Você faz isso para todas as contas de usuário do banco de dados que o proxy pode acessar. No Console de gerenciamento da AWS, escolha os segredos por seus nomes descritivos.

- Para criar um segredo do Secrets Manager e utilizá-lo com o RDS Proxy, use o comando create-secret:

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- Também é possível criar uma chave personalizada para criptografar o segredo do Secrets Manager. O comando a seguir cria uma chave de exemplo.

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
```

```
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
        "Resource":"*"
      }
    ]
}'
```

Por exemplo, os seguintes comandos criam segredos do Secrets Manager para dois usuários de banco de dados:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

Para criar esses segredos criptografados com a chave do AWS KMS personalizada, use os seguintes comandos:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
```

```
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

Para ver os segredos que pertencem à sua conta da AWS, use o comando list-secrets:

```
aws secretsmanager list-secrets
```

Ao criar um proxy utilizando a CLI, você transmite os nomes de recursos da Amazon (ARNs) de um ou mais segredos ao parâmetro `--auth`. O exemplo a seguir mostra como preparar um relatório com apenas o nome e o ARN de cada segredo pertencente à sua conta da AWS. Este exemplo usa o parâmetro `--output table` que está disponível na AWS CLI versão 2. Se você estiver usando a AWS CLI versão 1, use `--output text` em vez disso.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

Para confirmar se o segredo contém as credenciais corretas no formato adequado, use o comando get-secret-value. Substitua *your_secret_name* pelo nome abreviado ou ARN do segredo.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

A saída contém uma linha com um valor codificado em JSON semelhante ao seguinte:

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

Configurar a autenticação do IAM para o RDS Proxy

Para configurar a autenticação do AWS Identity and Access Management (IAM) para o RDS Proxy no Amazon RDS, crie e configure uma política do IAM que conceda as permissões necessárias.

Este tópico apresenta as etapas para configurar a autenticação do IAM para o RDS Proxy, bem como para criar a política do IAM necessária e anexá-la a um perfil do IAM.

> **Tip**
>
> Este procedimento só é necessário se você quiser criar seu próprio perfil do IAM. Caso contrário, o RDS pode criar automaticamente o perfil necessário ao configurar o proxy, para que você possa pular essas etapas.

Pré-requisitos

Antes de configurar a autenticação do IAM para o RDS Proxy, você deve ter o seguinte:

- AWS Secrets Manager: pelo menos um segredo armazenado que contenha as credenciais do banco de dados. Para obter instruções sobre como criar esses segredos, consulte the section called “Configurar de credenciais de banco de dados”.

  Isso não será necessário se você estiver usando a autenticação de ponta a ponta do IAM.
- Permissões do IAM: um perfil ou usuário do IAM com permissões para criar e gerenciar políticas, perfis e segredos do IAM no AWS Secrets Manager.

Criar uma política do IAM para a autenticação de ponta a ponta do IAM

Ao usar a autenticação de ponta a ponta do IAM, o RDS Proxy se conecta ao seu banco de dados usando a autenticação do IAM em vez de recuperar as credenciais do Secrets Manager. Isso requer configurar seu perfil do IAM com permissões `rds-db:connect` para as contas de banco de dados que você deseja usar com o proxy.

Para autenticar o RDS Proxy no banco de dados usando o IAM, crie um perfil do IAM com uma política que conceda as permissões de conexão com o banco de dados.

Console

Como criar um perfil para autenticação de ponta a ponta do IAM com seu proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Crie uma política de permissões para o perfil. Para ver as etapas gerais, consulte Criar políticas do IAM (console).

   Cole essa política no editor de JSON e faça as seguintes alterações:

- Substitua o ID da sua própria conta.
- Substitua `us-east-2` pela região onde o proxy deverá residir.
- Substitua os IDs de recursos do banco de dados e os nomes de usuário pelos que você deseja usar. O formato dos IDs de recursos difere entre as instâncias do RDS e Aurora clusters.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}
```

3. Crie o perfil e anexe a política de permissões a ele. Para ver as etapas gerais, consulte Criar um perfil para delegar permissões a um serviço da AWS.

   Em Tipo de entidade confiável, escolha Serviço da AWS. Em Caso de uso, selecione RDS e escolha RDS - Adicionar perfil ao banco de dados para o caso de uso.

4. Em Políticas de permissões, escolha a política que você criou.
5. Em Selecionar entidades confiáveis, insira a seguinte política de confiança para o perfil:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Para criar o perfil usando a AWS CLI, envie a seguinte solicitação:

```
aws iam create-role \
  --role-name my_e2e_iam_role_name \

  --assume-role-policy-document '{"Version":"2012-10-17",
     "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Depois, anexe a política ao perfil:

```
aws iam put-role-policy \
  --role-name my_e2e_iam_role_name \
  --policy-name e2e_iam_db_connect_policy \
  --policy-document '{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}'
```

Com as permissões e o perfil do IAM configurados para a autenticação de ponta a ponta do IAM, agora você pode criar um proxy com `DefaultAuthScheme` definido como `IAM_AUTH`. Esse proxy

se autentica diretamente no banco de dados usando o IAM sem exigir segredos do Secrets Manager. Para instruções, consulte the section called “Criar um proxy”.

Ao usar a autenticação de ponta a ponta do IAM, garanta que os usuários do banco de dados estejam configurados para a autenticação do IAM, conforme descrito em Criar uma conta de banco de dados usando autenticação do IAM.

Criar uma política do IAM para acesso ao Secrets Manager

Para permitir que o RDS Proxy recupere as credenciais do banco de dados do Secrets Manager, crie um perfil do IAM com uma política que conceda as permissões necessárias.

Console

Como criar um perfil para acessar segredos e usá-los com o proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Crie uma política de permissões para o perfil. Para ver as etapas gerais, consulte Criar políticas do IAM (console).

   Cole essa política no editor de JSON e faça as seguintes alterações:

   - Substitua o ID da sua própria conta.
   - Substitua `us-east-2` pela região onde o proxy residirá.
   - Substitua os nomes dos segredos pelos que você criou. Para ter mais informações, consulte Specifying KMS keys in IAM policy statements.
   - Substitua o ID da chave do KMS pelo que você usou para criptografar os segredos do Secrets Manager, seja a chave padrão ou sua própria chave.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
```

```
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_1",
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-
east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. Crie o perfil e anexe a política de permissões a ele. Para ver as etapas gerais, consulte Criar um perfil para delegar permissões a um serviço da AWS.

   Em Tipo de entidade confiável, escolha Serviço da AWS. Em Caso de uso, selecione RDS e escolha RDS - Adicionar perfil ao banco de dados para o caso de uso.

4. Em Políticas de permissões, escolha a política que você criou.
5. Em Selecionar entidades confiáveis, insira a seguinte política de confiança para o perfil:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

AWS CLI

Para criar o perfil usando a AWS CLI, envie a seguinte solicitação:

```
aws iam create-role \
  --role-name my_role_name \
  --assume-role-policy-document '{"Version": "2012-10-17",
      "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Depois, anexe a política ao perfil:

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

Com o perfil e as permissões do IAM configurados, agora você pode criar um proxy e associá-lo a esse perfil. Isso permite que o proxy recupere seguramente as credenciais do banco de dados por meio do AWS Secrets Manager e habilite a autenticação do IAM para suas aplicações. Para instruções, consulte the section called “Criar um proxy”.

Criar um proxy para o Amazon Aurora

É possível usar o Amazon RDS Proxy para melhorar a escalabilidade, disponibilidade e segurança das aplicações de banco de dados agrupando conexões e gerenciando failovers de banco de dados com maior eficiência. Este tópico descreve como criar um proxy. Antes de começar, assegure-se de que o banco de dados atenda aos pré-requisitos necessários, como permissões do IAM e configuração de VPC.

É possível associar um proxy a um cluster de banco de dados do Aurora MySQL ou Aurora PostgreSQL.

Console

Como criar um proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Escolha Create proxy (Criar proxy).
4. Defina as configurações de proxy a seguir.

Configuração	Descrição
Família de mecanismos	O protocolo de rede de banco de dados que o proxy reconhece quando interpreta o tráfego de rede de entrada e saída do banco de dados. **Note** Para usar o Aurora PostgreSQL, mantenha o banco de dados `postgres` na sua instância. Consulte Solução de problemas de exclusão do banco de dados `postgres`.

Configuração	Descrição
Identificador do proxy	Um nome exclusivo no ID da conta da AWS e na região da AWS atual.
Tempo-limite de inatividade da conexão do cliente	O proxy fechará a conexão do cliente se ela permanecer inativa por determinado período. Por padrão, é 1.800 segundos (30 minutos). Uma conexão fica ociosa quando a aplicação não envia uma nova solicitação dentro do período especificado após a conclusão da solicitação anterior. O proxy mantém a conexão do banco de dados subjacente aberta e a devolve ao grupo de conexões, disponibilizando-a para novas conexões de clientes. Para que o proxy remova proativamente as conexões ociosas, reduza o tempo-limite de inatividade da conexão do cliente. Para minimizar os custos de conexão durante picos de workload, aumente o tempo-limite.
Banco de dados	O cluster de bancos de dados do Aurora para acesso por meio desse proxy. A lista inclui somente instâncias e clusters de banco de dados com mecanismos de banco de dados compatíveis, versões de mecanismo e outras configurações. Se a lista estiver vazia, crie uma instância ou cluster de banco de dados compatível com o RDS Proxy. Para fazer isso, siga o procedimento em Criar um cluster de bancos de dados do Amazon Aurora. Depois, tente criar o proxy novamente.
Conexões máximas do grupo de conexões	Um valor entre 1 e 100 para definir a porcentagem do limite de `max_connections` que o RDS Proxy pode usar. Se pretender usar apenas um proxy com esse cluster ou instância de banco de dados, defina esse valor como 100. Para ter mais informaçõ es sobre como o RDS Proxy usa essa configuração, consulte the section called “MaxConnectionsPercent”.

Configuração	Descrição
Filtros de fixação de sessão	Impede que o RDS Proxy fixe determinados estados de sessão detectados, o que ignora as medidas de segurança padrão para conexões de multiplexação. No momento, o PostgreSQL não aceita essa configuração, e a única opção disponível é `EXCLUDE_VARIABLE_SETS`. Habilitá-la pode fazer com que as variáveis de sessão de uma conexão afetem outras, gerando erros ou problemas de correção caso as consultas dependam de variáveis de sessão definidas fora da transação atual. Use essa opção somente depois de confirmar que as aplicações podem compartilhar conexões de banco de dados com segurança. Os seguintes padrões são considerados seguros: • Declarações `SET` nas quais não haja alteração no valor efetivo da variável de sessão. Em outras palavras, não há alteração na variável de sessão. • Você altera o valor da variável de sessão e executa uma instrução na mesma transação. Para obter mais informações, consulte Evitar a fixação de um RDS Proxy.
Tempo-limite de empréstimo da conexão	Se você espera que o proxy use todas as conexões de banco de dados disponíveis, defina o tempo de espera antes que ele gere um erro de tempo-limite. É possível especificar até cinco minutos. Essa configuração só se aplica quando o proxy atingiu o número máximo de conexões e todas as conexões já estão em uso.

Configuração	Descrição
Consulta de inicialização	(Opcional) Adicione uma consulta de inicialização ou modifique a atual. Você pode especificar uma ou mais instruções SQL para o proxy executar ao abrir cada nova conexão de banco de dados. A configuração normalmente é usada com instruções SET para garantir que cada conexão tenha configurações idênticas. Verifique se a consulta adicionada é válida. Para incluir diversas variáveis em uma única instrução SET, use vírgulas como separadores. Por exemplo: `SET variable1 =value1, variable2 =value2` Para várias instruções, use um ponto e vírgula como separador. **Important** Como você pode acessar a consulta de inicialização como parte da configuração do grupo de destino, ela não é protegida por autenticação ou métodos criptográficos. Qualquer pessoa com acesso para visualizar ou gerenciar a configuração do seu grupo de destino de proxy pode visualizar a consulta de inicialização. Você não deve adicionar dados sensíveis, como senhas ou chaves de criptografia de longa duração, a essa opção.
AWS Identity and Access Management (Perfil do (IAM	Um perfil do IAM com permissão para acessar os segredos do Secrets Manager, que representam as credenciais das contas de usuário do banco de dados que o proxy pode usar. Também é possível criar um perfil do IAM no Console de gerenciamento da AWS.

Configuração	Descrição
Segredos do Secrets Manager	Crie ou escolha segredos do Secrets Manager que representam as credenciais das contas de usuários do banco de dados que podem usar o proxy. Quando o esquema de autenticação padrão está definido como Nenhum, esse campo é obrigatório. Quando o esquema de autenticação padrão está definido como autenticação do IAM, esse campo se torna opcional e é marcado como tal no console. Você pode escolher um ou mais segredos na lista suspensa ou criar outro segredo usando o link Criar novo segredo.
Tipo de autenticação de cliente	O tipo de autenticação usada pelo proxy para conexões de clientes. Sua escolha se aplica a todos os segredos do Secrets Manager que você associa a esse proxy. Se você precisar especificar um tipo de autenticação de cliente diferente para cada segredo, crie seu proxy usando a AWS CLI ou a API. Especifique essa opção somente quando sua conexão de cliente usa credenciais de banco de dados para autenticação.
Autenticação do IAM	Especifique Obrigatório ou Não Permitido para a autenticação do IAM para conexões com seu proxy. Sua escolha se aplica a todos os segredos do Secrets Manager que você associa a esse proxy. Se você precisar especificar um tipo de autenticação do IAM diferente para cada segredo, crie seu proxy usando a AWS CLI ou a API.

Configuração	Descrição
Esquema de autenticação padrão	Escolha o tipo de autenticação padrão usado pelo proxy para conexões do cliente com o proxy e as conexões do proxy com o banco de dados subjacente. Você tem as seguintes opções: • Nenhum (padrão): o proxy recupera as credenciais do banco de dados dos segredos do Secrets Manager. • Autenticação do IAM: o proxy usa a autenticação do IAM para se conectar ao banco de dados, permitindo a autenticação de ponta a ponta do IAM. Quando você seleciona a autenticação do IAM, um alerta de informações aparece lembrando que você deve habilitar a autenticação do banco de dados do IAM para os bancos de dados na configuração do grupo de destino. **Note** Essa opção comporta somente as famílias de mecanismos MySQL, PostgreSQL e MariaDB.

Configuração	Descrição
Contas de banco de dados para autenticação do IAM	Esse campo aparece somente quando o esquema de autenticação padrão está definido como autenticação do IAM e o perfil do Identity and access management (IAM) está definido como Criar perfil do IAM. Nomeie as contas de usuário do banco de dados que o proxy usará com a autenticação do IAM. Este é um campo obrigatório. Especifique várias contas: • Digitando um nome de usuário do banco de dados para adicioná-lo como uma tag. • Usando nomes de usuário específicos do banco de dados (por exemplo, `db_user`, `jane_doe`). • Usando padrões curinga para vários usuários (por exemplo, `db_test_*`). Cada conta aparece como uma tag removível que você pode excluir clicando no ícone X. O console usa esses valores para criar as permissões `rds-db:connect` apropriadas na política de perfis do IAM.
Exigir Transport Layer Security	Impõe TLS/SSL para todas as conexões de cliente. O proxy usa a mesma configuração de criptografia para sua conexão com o banco de dados subjacente, quer a conexão do cliente seja criptografada ou não criptografada.

Configuração	Descrição
Tipo de rede de conexão de destino	A versão do IP que o proxy usa para se conectar ao banco de dados de destino. Escolha uma das seguintes opções: • IPv4: o proxy se conecta ao banco de dados usando endereços IPv4. • IPv6: o proxy se conecta ao banco de dados usando endereços IPv6. O padrão é IPv4. Para usar o IPv6, seu banco de dados deve aceitar o modo de pilha dupla. O modo de pilha dupla não está disponível para as conexões de destino.
Tipo de rede de endpoint	A versão do IP do endpoint de proxy que os clientes usam para se conectar ao proxy. Escolha uma das seguintes opções: • IPv4: o endpoint de proxy usa somente endereços IPv4. • IPv6: o endpoint de proxy usa somente endereços IPv6. • Pilha dupla: o endpoint de proxy comporta endereços IPv4 e IPv6. O padrão é IPv4. Para usar IPv6 ou pilha dupla, sua VPC e sub-redes devem ser configuradas para comportar o tipo de rede selecionado.
Sub-redes	Esse campo é pré-preenchido com todas as sub-redes associadas à sua VPC. É possível remover quaisquer sub-redes que não sejam necessárias para o proxy, mas pelo menos duas sub-redes devem ser mantidas. Para tipos de rede de endpoint IPv6 ou de pilha dupla, garanta que as sub-redes selecionadas comportem o tipo de rede escolhido.

Configuração	Descrição
VPC grupo de segurança (Grupo de segurança da VPC	Escolha um grupo de segurança da VPC existente ou crie um no Console de gerenciamento da AWS. Configure as regras de entrada para possibilitar que suas aplicações acessem o proxy e as regras de saída para permitir o tráfego de seus destinos de banco de dados. **Note** O grupo de segurança deve permitir conexões do proxy com o banco de dados. Ele serve para entrada das aplicações para o proxy e saída do proxy para o banco de dados. Por exemplo, se você usar o mesmo grupo de segurança para o banco de dados e o proxy, verifique se os recursos nesse grupo de segurança podem se comunicar entre si. Ao usar uma VPC compartilhada, evite usar o grupo de segurança padrão para a VPC ou um associado a outra conta. Em vez disso, escolha um grupo de segurança que pertença à sua conta. Se não houver nenhum, crie um. Para ter mais informações, consulte Trabalhar com VPCs compartilhadas. O RDS implanta um proxy em várias zonas de disponibilidade para garantir a alta disponibilidade. Para habilitar a comunicaç ão entre AZs, a lista de controle de acesso (ACL) de rede da sub-rede de proxy deve permitir saída na porta do mecanismo e entrada em todas as portas. Para obter mais informações sobre ACLs de rede, consulte Controlar o tráfego para sub-redes com ACLs de rede. Se a ACL de rede do proxy e do destino for idêntica, você deverá adicionar uma regra de entrada do protocolo TCP em que a Fonte esteja definida como o CIDR da VPC. Você também deve adicionar uma regra de saída de

Configuração	Descrição
	protocolo TCP específica da porta do mecanismo em que o Destino esteja definido como o CIDR da VPC.
Ativar o registro em log aprimorado	Habilite essa configuração para solucionar problemas de compatibilidade ou de performance do proxy. Quando habilitado, o RDS Proxy registra em log informações detalhadas de performance para ajudar você a depurar o comportamento do SQL ou a performance e a escalabilidade da conexão do proxy. Habilite essa configuração apenas para depuração e garanta que as medidas de segurança adequadas estejam em vigor para proteger as informações confidenciais nos logs. Para minimizar as despesas operacionais, o RDS Proxy desabilita automaticamente essa configuração 24 horas após a ativação. Habilite-a temporariamente para solucionar problemas específicos.

5. Escolha Create proxy (Criar proxy).

AWS CLI

Para criar um proxy utilizando a AWS CLI, chame o comando create-db-proxy com os seguintes parâmetros obrigatórios:

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--vpc-subnet-ids`

O valor `--engine-family` diferencia letras maiúsculas de minúsculas.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-proxy \
```

```
    --db-proxy-name proxy_name \
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--default-auth-scheme { NONE | IAM_AUTH }] \
    [--auth ProxyAuthenticationConfig_JSON_string] \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] \
    [--target-connection-network-type { IPV4 | IPV6 }] \
    [--tags comma_separated_list]
```

Para Windows:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--default-auth-scheme { NONE | IAM_AUTH }] ^
    [--auth ProxyAuthenticationConfig_JSON_string] ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] ^
    [--target-connection-network-type { IPV4 | IPV6 }] ^
    [--tags comma_separated_list]
```

Veja a seguir um exemplo do valor JSON da opção `--auth`. Este exemplo aplica um tipo de autenticação de cliente diferente a cada segredo.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },
```

```
  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

O parâmetro `--endpoint-network-type` especifica a versão do IP do endpoint de proxy que os clientes usam para se conectar ao proxy. Os valores válidos são:

- `IPV4`: o endpoint de proxy usa somente endereços IPv4 (padrão).
- `IPV6`: o endpoint de proxy usa somente endereços IPv6.
- `DUAL`: o endpoint de proxy comporta endereços IPv4 e IPv6.

O parâmetro `--target-connection-network-type` especifica a versão do IP que o proxy usa para se conectar ao banco de dados de destino. Os valores válidos são:

- `IPV4`: o proxy se conecta ao banco de dados usando endereços IPv4 (padrão).
- `IPV6`: o proxy se conecta ao banco de dados usando endereços IPv6.

Para usar tipos de rede de endpoint IPv6 ou de pilha dupla, sua VPC e sub-redes devem ser configuradas para comportar o tipo de rede selecionado. Para usar o tipo de rede de conexão de destino IPv6, seu banco de dados deve comportar o modo de pilha dupla.

> **Tip**
>
> Se você ainda não souber os IDs de sub-rede que serão usados para o parâmetro `--vpc-subnet-ids`, consulte Configurar os pré-requisitos de rede para o RDS Proxy para obter exemplos de como encontrá-los.

> **Note**
>
> O grupo de segurança deve permitir acesso ao banco de dados ao qual o proxy se conecta. O mesmo grupo de segurança é usado para entrada das aplicações para o proxy e para saída do proxy para o banco de dados. Por exemplo, suponha que você use o mesmo grupo de segurança para o banco de dados e o proxy. Nesse caso, especifique que os recursos nesse grupo de segurança podem se comunicar com outros recursos no mesmo grupo de segurança.
>
> Ao usar uma VPC compartilhada, você não pode usar o grupo de segurança padrão para a VPC ou um grupo de segurança pertencente a outra conta. Escolha um grupo de segurança que pertença à sua conta. Se não houver, crie um. Para ter mais informações sobre essa limitação, consulte Trabalhar com VPCs compartilhadas.

Para criar as associações corretas para o proxy, também é possível usar o comando register-db-proxy-targets. Especificar o nome do grupo de destino do `default`. O proxy do RDS cria automaticamente um grupo de destino com este nome ao criar cada proxy.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

API do RDS

Para criar um proxy do RDS , chame a operação CreateDBProxy da API do Amazon RDS. Passe um parâmetro com a estrutura de dados AuthConfig.

O proxy do RDS cria automaticamente um grupo de destino chamado `default` ao criar cada proxy. Associe um cluster de bancos de dados do Aurora ao grupo de destino chamando a função RegisterDBProxyTargets.

> **Important**
>
> Quando você seleciona a autenticação do IAM para o esquema de autenticação padrão:
>
> - Você deve habilitar a autenticação do banco de dados do IAM em suas instâncias ou clusters de banco de dados de destino para que o proxy possa se conectar com êxito.
> - Se você escolher Criar perfil do IAM, o campo Contas de banco de dados para autenticação do IAM será obrigatório.
> - Se você selecionar um perfil do IAM existente, o console não atualizará automaticamente o perfil com as permissões de conexão do banco de dados. Confira se o perfil tem as permissões `rds-db:connect` necessárias.

Visualizar um proxy

Depois de criar um ou mais proxies do RDS, você pode visualizá-los e gerenciá-los no Console de gerenciamento da AWS, na AWS CLI ou na API do RDS. É possível analisar seus detalhes de configuração, monitorar a performance e determinar quais proxies devem ser modificados ou excluídos, conforme necessário.

Para possibilitar que as aplicações de banco de dados direcionem o tráfego por meio de um proxy, é necessário especificar o endpoint do proxy na string de conexão.

Console

Como visualizar um proxy no console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Selecione o nome do proxy para visualizar os detalhes.
4. Na página de detalhes, a seção Grupos de destino mostra como o proxy está vinculado a um cluster de bancos de dados específico do Aurora. É possível acessar a página do grupo de destino padrão para ter uma visão mais detalhada dessa associação, incluindo as configurações definidas durante a criação do proxy. Essas configurações incluem a porcentagem máxima de conexão, o tempo-limite de empréstimo da conexão, a família de mecanismos e os filtros de fixação de sessão.

CLI

Para visualizar o proxy usando a CLI, use o comando describe-db-proxies. Por padrão, a solicitação exibe todos os proxies pertencentes à sua conta da AWS. Para ver detalhes de um único proxy, especifique o nome dele com o parâmetro `--db-proxy-name`.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

Para visualizar as outras informações associadas ao proxy, use os comandos a seguir.

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

Use a seguinte sequência de comandos para ver mais detalhes sobre as coisas associadas ao proxy:

1. Para obter uma lista de proxies, execute describe-db-proxies.
2. Para mostrar parâmetros de conexão, como a porcentagem máxima de conexões que o proxy pode usar, execute describe-db-proxy-target-groups `--db-proxy-name`. Use o nome do proxy como o valor do parâmetro.
3. Para ver os detalhes do cluster de bancos de dados do Aurora associado ao grupo de destino exibido, execute describe-db-proxy-targets.

API do RDS

Para visualizar seus proxies usando a API do RDS, use a operação DescribeDBProxies. Ela retorna valores do tipo de dados DBProxy.

Para ver detalhes das configurações de conexão do proxy, use os identificadores de proxy desse valor de retorno com a operação DescribeDBProxyTargetGroups. Ela retorna valores do tipo de dados DBProxyTargetGroup.

Para ver a instância do RDS ou o cluster de bancos de dados Aurora associado ao proxy, use a operação DescribeDBProxyTargets. Ela retorna valores do tipo de dados DBProxyTarget.

Conectar-se a um banco de dados pelo RDS Proxy

Você se conecta a um cluster de banco de dados do Aurora ou a um cluster que usa o Aurora Serverless v2 por meio de um proxy, em geral da mesma maneira como se conecta diretamente ao banco de dados. A principal diferença é que você especifica o endpoint do proxy em vez do endpoint do cluster. Por padrão, todas as conexões de proxy têm capacidade de leitura/gravação e usam a instância de gravação. Se você normalmente usa o endpoint do leitor para conexões somente leitura, poderá criar um endpoint adicional somente leitura para o proxy. Você pode usar esse endpoint da mesma forma. Para obter mais informações, consulte Visão geral dos endpoints de proxy.

Tópicos

- Conectar-se a um banco de dados usando credenciais de banco de dados
- Conectar-se a um banco de dados usando autenticação do IAM
- Considerações sobre como se conectar ao PostgreSQL

Conectar-se a um banco de dados usando credenciais de banco de dados

Use as seguintes etapas para se conectar a um proxy usando as credenciais de banco de dados:

1. Localize o endpoint do proxy. No Console de gerenciamento da AWS, você pode encontrar o endpoint na página de detalhes do proxy correspondente. Com a AWS CLI, é possível usar o comando describe-db-proxies. O exemplo a seguir mostra como.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
```

```
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. Especifique o endpoint como o parâmetro host na string de conexão da aplicação cliente. Por exemplo, especifique o endpoint do proxy como o valor da opção `mysql -h` ou da opção `psql -h`.
3. Forneça o mesmo nome de usuário e senha do banco de dados como o faz normalmente.

Conectar-se a um banco de dados usando autenticação do IAM

Ao usar a autenticação do IAM com o RDS Proxy, você tem duas opções de autenticação entre o cliente e o proxy:

- Configure os usuários do banco de dados para fazer a autenticação com nomes de usuário e senhas regulares. O RDS Proxy recupera as credenciais de nome de usuário e senha do Secrets Manager. A conexão do proxy do RDS com o banco de dados subjacente não passa pelo IAM.
- Você também pode usar a autenticação de ponta a ponta do IAM, que se conecta ao seu banco de dados por meio do proxy usando o IAM sem exigir credenciais de banco de dados.

Para conectar-se ao RDS Proxy usando a autenticação do IAM, siga o mesmo procedimento de conexão geral utilizado na autenticação do IAM com um cluster de banco de dados do Aurora. Para obter mais informações sobre como usar a IAM, consulte Segurança no Amazon Aurora. Se você estiver usando a autenticação de ponta a ponta do IAM, forneça o plug-in de autenticação do IAM ao seu usuário de banco de dados. Consulte Criar uma conta de banco de dados usando autenticação do IAM.

As principais diferenças no uso do IAM para o proxy do RDS incluem o seguinte:

- Com a autenticação padrão do IAM, os usuários do banco de dados têm credenciais normais no banco de dados. Você configura segredos do Secrets Manager contendo esses nomes de usuário e senhas e autoriza o proxy do RDS a recuperar as credenciais do Secrets Manager. A

autenticação do IAM aplica-se à conexão entre o programa cliente e o proxy. Depois, o proxy faz a autenticação no banco de dados usando as credenciais de nome de usuário e senha recuperadas pelo Secrets Manager.

- Com a autenticação de ponta a ponta do IAM, você não precisa configurar segredos do Secrets Manager para credenciais de banco de dados. A autenticação do IAM aplica-se à conexão entre o cliente e o proxy e entre o proxy e o banco de dados.
- Em vez da instância, do cluster ou do endpoint de leitor, especifique o endpoint do proxy. Para obter detalhes sobre o endpoint do proxy, consulte Conectar-se ao cluster de banco de dados usando a autenticação do IAM.
- Certifique-se de usar os protocolos Transport Layer Security (TLS)/Secure Sockets Layer (SSL) ao se conectar a um proxy usando a autenticação do IAM.

É possível conceder a um usuário específico acesso ao proxy modificando a política do IAM. Veja a seguir um exemplo.

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

Tip

Ao configurar a autenticação do IAM para conexões do RDS Proxy, siga estas diretrizes importantes para evitar problemas de conexão:

- Não conceda o perfil `rds_iam` enquanto mantém a autenticação geral por senha para o mesmo usuário ou perfil do banco de dados.
- Lembre-se de que, embora os clientes se conectem ao RDS Proxy usando a autenticação do IAM, o RDS Proxy sempre se conecta ao banco de dados usando a autenticação por senha por meio do Secrets Manager.
- Se você tiver encerramentos e novas conexões frequentes, remova todas as concessões `rds_iam` existentes do usuário ou do perfil e use somente a autenticação por senha.
- Garanta que sua política de senha atenda aos requisitos de caracteres seguros SCRAM-SHA-256.

Combinar métodos de autenticação do IAM e senha para o mesmo usuário do banco de dados pode causar instabilidade na conexão.

Considerações sobre como se conectar ao PostgreSQL

Se você criar um usuário de banco de dados PostgreSQL para se conectar ao RDS Proxy, conceda a ele o privilégio `CONNECT` no banco de dados. Sem isso, o usuário não poderá estabelecer uma conexão. Para obter mais informações, consulte the section called “Adicionar um novo usuário de banco de dados a um banco de dados do PostgreSQL ao usar o RDS Proxy”.

Quando um cliente inicia uma conexão com um banco de dados PostgreSQL, ele envia uma mensagem de inicialização. Essa mensagem inclui pares de strings de caracteres de nome e valor do parâmetro. Para obter mais detalhes, consulte o `StartupMessage` em PostgreSQL Message Formats na documentação do PostgreSQL.

Ao se conectar por meio de um proxy do RDS, a mensagem de inicialização pode incluir os seguintes parâmetros reconhecidos atualmente:

- `user`
- `database`

A mensagem de inicialização também pode incluir os seguintes parâmetros de tempo de execução adicionais:

- `application_name`
- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

Para ter mais informações sobre sistemas de mensagens PostgreSQL, consulte Frontend/Backend Protocol na documentação do PostgreSQL.

Para o PostgreSQL, se você usar JDBC, recomendamos o seguinte para evitar a fixação:

- Defina o parâmetro de conexão JDBC `assumeMinServerVersion` como pelo menos `9.0` para evitar a fixação. Isso impede que o driver JDBC execute uma viagem de ida e volta adicional durante a inicialização da conexão quando ele executa `SET extra_float_digits = 3`.

- Defina o parâmetro de conexão JDBC `ApplicationName` como *`any/your-application-name`* para evitar a fixação. Isso impede que o driver JDBC execute uma viagem de ida e volta adicional durante a inicialização da conexão quando ele executa `SET application_name = "PostgreSQL JDBC Driver"`. Observe que o parâmetro JDBC é `ApplicationName`, mas o parâmetro `StartupMessage` do PostgreSQL é `application_name`.

Para obter mais informações, consulte Evitar a fixação de um RDS Proxy. Para ter mais informações sobre como se conectar usando JDBC, consulte Connecting to the Database na documentação do PostgreSQL.

Gerenciar um RDS Proxy

Esta seção fornece informações sobre como gerenciar a operação e a configuração do RDS Proxy. Esses procedimentos ajudam a aplicação a fazer o uso mais eficiente de conexões de banco de dados e alcançar o máximo de reutilização de conexão. Quanto mais você puder aproveitar a reutilização de conexões, poderá economizar mais sobrecarga de CPU e de memória. Isso, por sua vez, reduz a latência da aplicação e permite que o banco de dados dedique mais de seus recursos ao processamento de solicitações da aplicação.

Tópicos

- Modificar um RDS Proxy
- Adicionar um novo usuário de banco de dados ao usar o RDS Proxy
- Migrar da autenticação padrão do IAM para a autenticação de ponta a ponta do IAM para o RDS Proxy
- Considerações sobre a conexão do RDS Proxy
- Evitar a fixação de um RDS Proxy
- Excluir um RDS Proxy

Modificar um RDS Proxy

Você pode alterar determinadas configurações associadas a um proxy depois de criar o proxy. Faça isso modificando o próprio proxy, seu grupo de destino associado ou ambos. Cada proxy tem um grupo de destino associado.

Console de gerenciamento da AWS

> **Important**
>
> Os valores nos campos Client authentication type (Tipo de autenticação do cliente) e IAM authentication (Autenticação do IAM) se aplicam a todos os segredos do Secrets Manager associados a esse proxy. Para especificar valores diferentes para cada segredo, modifique seu proxy usando a AWS CLI ou a API.

Como modificar as configurações de um proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Na lista de proxies, escolha o proxy cujas configurações você deseja modificar ou vá para sua página de detalhes.
4. Para Actions (Ações), escolha Modify (Modificar).
5. Insira ou escolha as propriedades a serem modificadas. Você pode modificar o seguinte:
   - Proxy identifier (Identificador de proxy): renomeie o proxy inserindo um novo identificador.
   - Idle client connection timeout (Tempo limite da conexão do cliente ociosa): insira um período para o tempo limite da conexão do cliente ociosa.
   - IAM role (Perfil do IAM): altere o perfil do IAM usada para recuperar os segredos do Secrets Manager.

     > **Note**
     >
     > Você não poderá criar um perfil do IAM se definir o esquema de autenticação padrão como autenticação do IAM.

   - Secrets Manager secrets (Segredos do Secrets Manager): adicione ou remova segredos do Secrets Manager. Esses segredos correspondem a nomes de usuário e senhas de banco de dados.
   - Tipo de autenticação de cliente: altere o tipo de autenticação das conexões do cliente com o proxy.

- IAM Authentication (Autenticação do IAM): exija ou desabilite a autenticação do IAM para conexões com o proxy.
- Esquema de autenticação padrão: altere o esquema de autenticação padrão usado pelo proxy para conexões do cliente com o proxy e conexões do proxy com o banco de dados subjacente.
- Require Transport Layer Security (Exija Transport Layer Security): ative ou desative o requisito de segurança do Transport layer Security (TLS).
- VPC grupo de segurança (Grupos de segurança da VPC): adicione ou remova grupos de segurança da VPC para uso do proxy.
- Enable enhanced logging (Habilite o registro em log aprimorado): habilite ou desabilite o registro em log aprimorado.

6. Selecione Modify.

Se você não encontrou as configurações listadas que deseja alterar, use o procedimento a seguir para atualizar o grupo de destino para o proxy. O grupo de destino associado a um proxy controla as configurações relacionadas às conexões físicas do banco de dados. Cada proxy tem um grupo de destino associado, chamado `default`, que é criado automaticamente com o proxy. Não é possível renomear o grupo de destino padrão.

Você só pode modificar o grupo de destino na página de detalhes do proxy, não na lista da página Proxies.

Como modificar as configurações de um grupo de destino de proxy

1. Na página Proxies, acesse a página de detalhes de um proxy.
2. Em Target groups (Grupos de destino), escolha o link `default`. Atualmente, todos os proxies têm um único grupo de destino chamado `default`.
3. Na página de detalhes do grupo de destino default (padrão) escolha Modify (Modificar).
4. Escolha novas configurações para as propriedades que você pode modificar:

   - Banco de dados: escolha outro cluster do Aurora.
   - Connection pool maximum connections (Conexões máximas do grupo de conexões): ajuste a porcentagem das conexões máximas disponíveis que o proxy pode usar.
   - Session pinning filters (Filtros de fixação de sessões): (opcional) escolha um filtro de fixação de sessões. Isso contorna as medidas de segurança padrão para multiplexar conexões de

banco de dados entre conexões de clientes. No momento, a configuração não é compatível com o PostgreSQL. A única opção é `EXCLUDE_VARIABLE_SETS`.

Habilitar essa configuração pode fazer com que as variáveis da sessão de uma conexão afetem outras conexões. Isso pode causar erros ou problemas de correção se suas consultas dependerem dos valores das variáveis de sessão definidos fora da transação atual. Considere a possibilidade de usar essa opção depois de verificar se é seguro as aplicações compartilharem conexões de banco de dados entre conexões de clientes.

Os seguintes padrões podem ser considerados seguros:

- Instruções `SET` em que não há alteração no valor efetivo da variável de sessão; isto é, não há alteração na variável de sessão.
- Você altera o valor da variável de sessão e executa uma instrução na mesma transação.

Para obter mais informações, consulte Evitar a fixação de um RDS Proxy.

- Connection borrow timeout (Tempo limite do empréstimo de conexões): ajuste o intervalo do tempo limite de empréstimo de conexões. Essa configuração se aplica quando o número máximo de conexões já está sendo usado para o proxy. A configuração determina quanto tempo o proxy espera que uma conexão fique disponível antes de retornar um erro de tempo limite.
- Consulta de inicialização. (Opcional) Adicione uma consulta de inicialização ou modifique a atual. Você pode especificar uma ou mais instruções SQL para o proxy executar ao abrir cada nova conexão de banco de dados. A configuração normalmente é usada com instruções `SET` para garantir que cada conexão tenha configurações idênticas. Verifique se a consulta adicionada é válida. Para incluir diversas variáveis em uma única instrução `SET`, use vírgulas como separadores. Por exemplo:

```
SET variable1=value1, variable2=value2
```

Para várias instruções, use um ponto e vírgula como separador.

Não é possível alterar determinadas propriedades, como o identificador do grupo de destino e o mecanismo de banco de dados.

5. Escolha Modify target group (Modificar grupo de destino).

AWS CLI

Para modificar um proxy usando a AWS CLI, use os comandos modify-db-proxy, modify-db-proxy-target-group, deregister-db-proxy-targets e register-db-proxy-targets.

Com o comando `modify-db-proxy`, é possível alterar propriedades como as seguintes:

- O conjunto de segredos do Secrets Manager usados pelo proxy.
- Se o TLS é necessário.
- O tempo limite do cliente ocioso.
- Se deseja registrar em log informações adicionais de instruções SQL para depuração.
- O perfil do IAM usada para recuperar segredos do Secrets Manager.
- Os grupos de segurança usados pelo proxy.
- O esquema de autenticação padrão associado ao proxy.

O exemplo a seguir mostra como renomear um proxy existente.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

Para modificar configurações relacionadas à conexão ou renomear o grupo de destino, use o comando `modify-db-proxy-target-group`. Atualmente, todos os proxies têm um único grupo de destino chamado `default`. Ao trabalhar com esse grupo de destino, especifique o nome do proxy e `default` para o nome do grupo de destino. Não é possível renomear o grupo de destino padrão.

O exemplo a seguir mostra como verificar primeiro a configuração de `MaxIdleConnectionsPercent` de um proxy e alterá-la usando o grupo de destino.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
```

```
                "ConnectionBorrowTimeout": 120,
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

Com os comandos `deregister-db-proxy-targets` e `register-db-proxy-targets`, você altera a quais clusters de bancos de dados do Aurora o proxy está associado por meio de seu grupo de destino. No momento, cada proxy pode se conectar a um cluster de banco de dados do Aurora. O grupo de destino controla os detalhes da conexão de todas as instâncias de banco de dados em um cluster do Aurora.

O exemplo a seguir começa com um proxy associado a um cluster do Aurora MySQL chamado `cluster-56-2020-02-25-1399`. O exemplo mostra como alterar o proxy para que ele possa se conectar a outro cluster chamado `provisioned-cluster`.

Ao trabalhar com um cluster de bancos de dados Aurora, você especifica a opção `--db-cluster-identifier`.

O exemplo a seguir modifica um proxy Aurora MySQL. Um proxy Aurora PostgreSQL tem a porta 5432.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}
```

```
aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

API do RDS

Para modificar um proxy utilizando a API do RDS, utilize as operações ModifyDBProxy, ModifyDBProxyTargetGroup, DeregisterDBProxyTargets e RegisterDBProxyTargets.

Com `ModifyDBProxy`, é possível alterar propriedades como as seguintes:

- O conjunto de segredos do Secrets Manager usados pelo proxy.
- Se o TLS é necessário.

- O tempo limite do cliente ocioso.
- Se deseja registrar em log informações adicionais de instruções SQL para depuração.
- O perfil do IAM usada para recuperar segredos do Secrets Manager.
- Os grupos de segurança usados pelo proxy.

Com `ModifyDBProxyTargetGroup`, é possível modificar as configurações relacionadas à conexão. Atualmente, todos os proxies têm um único grupo de destino chamado `default`. Ao trabalhar com esse grupo de destino, especifique o nome do proxy e `default` para o nome do grupo de destino. Não é possível renomear o grupo de destino padrão.

Com `DeregisterDBProxyTargets` e `RegisterDBProxyTargets`, altere a qual cluster do Aurora o proxy está associado por meio do grupo de destino. Atualmente, cada proxy pode se conectar a um cluster de bancos de dados do Aurora. O grupo de destino monitora os detalhes da conexão das instâncias de banco de dados em um cluster do Aurora.

Adicionar um novo usuário de banco de dados ao usar o RDS Proxy

Em alguns casos, é possível adicionar um novo usuário de banco de dados a um cluster do Aurora com associação a um proxy. Prossiga dependendo se você estiver usando a autenticação padrão com segredos do Secrets Manager ou a autenticação de ponta a ponta do IAM.

Se você estiver usando a autenticação padrão do IAM, siga estas instruções:

1. Crie um segredo do Secrets Manager usando o procedimento descrito em Configurar credenciais de banco de dados para RDS Proxy.
2. Atualize o perfil do IAM para conceder ao proxy do RDS acesso ao novo segredo do Secrets Manager. Para fazer isso, atualize a seção de recursos da política de perfil do IAM.
3. Modifique o RDS Proxy para adicionar o novo segredo do Secrets Manager em Segredos do Secrets Manager.
4. Se o novo usuário tomar o lugar de um existente, atualize as credenciais armazenadas no segredo do Secrets Manager do proxy do usuário existente.

Se você estiver usando a autenticação de ponta a ponta do IAM, precisará criar o usuário do banco de dados e configurar as permissões do IAM. Para isso, siga estas etapas:

1. Crie um usuário em seu banco de dados que corresponda ao nome de usuário ou perfil do IAM que você deseja usar para autenticação.

2. O usuário do banco de dados deve estar configurado com o plug-in de autenticação do IAM no banco de dados. Consulte Criar uma conta de banco de dados usando autenticação do IAM.
3. Atualize a política do IAM para conceder a permissão `rds-db:connect` ao usuário ou ao perfil do IAM, conforme descrito em Criar uma política do IAM para a autenticação de ponta a ponta do IAM.
4. Seu proxy deve estar configurado para usar a autenticação do IAM como o esquema de autenticação padrão.

Com a autenticação de ponta a ponta do IAM, você não precisa gerenciar as credenciais do banco de dados nos segredos do Secrets Manager, pois as credenciais do IAM são usadas para autenticação do cliente para o proxy e do proxy para o banco de dados.

Adicionar um novo usuário de banco de dados a um banco de dados do PostgreSQL ao usar o RDS Proxy

Ao adicionar um novo usuário ao banco de dados do PostgreSQL, se você tiver executado o seguinte comando:

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

Conceda ao usuário `rdsproxyadmin` o privilégio `CONNECT` para que ele possa monitorar as conexões no banco de dados de destino.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Também é possível permitir que outros usuários do banco de dados de destino realizem verificações de integridade mudando `rdsproxyadmin` para o usuário do banco de dados no comando acima.

Alterar a senha de um usuário de banco de dados ao usar o RDS Proxy

Em alguns casos, você pode alterar a senha de um usuário de banco de dados em um cluster do Aurora com associação a um proxy. Nesse caso, atualize o segredo do Secrets Manager correspondente com a nova senha.

Se você estiver usando a autenticação de ponta a ponta do IAM, não precisará atualizar nenhuma senha nos segredos do Secrets Manager.

Migrar da autenticação padrão do IAM para a autenticação de ponta a ponta do IAM para o RDS Proxy

Se atualmente você usa a autenticação do IAM padrão para RDS Proxy, em que os clientes se autenticam no proxy usando o IAM, mas o proxy se conecta ao banco de dados usando segredos, você pode migrar para a autenticação de ponta a ponta do IAM, na qual as conexões cliente-para-proxy e proxy-para-banco de dados usam a autenticação do IAM.

Como migrar para a autenticação de ponta a ponta do IAM

1. Atualizar as permissões do perfil do IAM para RDS Proxy

   Crie uma política de permissão de proxy atualizada que inclua o Secrets Manager e permissões `rds:db-connect`:

```
# Create updated proxy permission policy
cat > updated-proxy-policy.json # EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetSecretsValue",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-1234f"
      ]
    },
    {
      "Sid": "RdsDBConnect",
      "Action": [
        "rds-db:connect"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:rds-db:us-east-1:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe"
      ]
```

```
    }
  ]
}
```

Atualize a política de perfil do proxy:

```
aws iam put-role-policy \
            --role-name RDSProxyRole \
            --policy-name UpdatedProxyPermissions \
            --policy-document file://updated-proxy-policy.json
```

2. Modificar o RDS Proxy para habilitar a autenticação de ponta a ponta do IAM

```
aws rds modify-db-proxy \
  --db-proxy-name my-database-proxy \
  --default-auth-scheme IAM_AUTH \
  --region us-east-1
```

Verifique se o status do Proxy RDS está disponível e `DefaultAuthScheme` é `IAM_AUTH` antes de continuar para garantir zero tempo de inatividade durante a migração.

```
aws rds describe-db-proxies --db-proxy-name my-database-proxy --region us-east-1
```

Saída esperada:

```
{
  "DBProxies": [
    {
      "DBProxyName": "my-database-proxy",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123456789abcdef",
      "Status": "available",
      ...
      "DefaultAuthScheme": "IAM_AUTH"
    }
  ]
}
```

3. Habilitar a autenticação do IAM no banco de dados

```
aws rds modify-db-cluster \
```

```
  --db-cluster-identifier my-database-cluster \
  --enable-iam-database-authentication \
  --region us-east-1
```

4. Configurar o usuário do banco de dados para autenticação do IAM

   Para Aurora PostgreSQL:

   ```
   GRANT rds_iam TO jane_doe;
   ```

   Para Aurora MySQL:

   ```
   ALTER USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
   ALTER USER 'jane_doe'@'%' REQUIRE SSL;
   ```

5. O código da sua aplicação cliente não precisa mudar. O processo de conexão permanece o mesmo:

   Para Aurora PostgreSQL:

   ```
   # Generate authentication token
   export PGPASSWORD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
     --port 5432 \
     --username jane_doe \
     --region us-east-1)

   # Connect to database through proxy
   psql "host=my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    port=5432 user=jane_doe dbname=postgres password=$PGPASSWORD sslmode=require
    sslrootcert=us-east-1-bundle.pem"
   ```

   Para Aurora MySQL:

   ```
   # Generate authentication token
   export MYSQL_PWD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
     --port 3306 \
     --username jane_doe \
     --region us-east-1)
   ```

```
# Connect to database through proxy
mysql -h my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com \
  -P 3306 \
  -u jane_doe \
  --ssl-ca=us-east-1-bundle.pem \
  --enable-cleartext-plugin
```

Considerações sobre a conexão do RDS Proxy

Configurar configurações de conexões

Para ajustar o grupo de conexões do RDS Proxy, você pode modificar as seguintes configurações:

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

É possível especificar por quanto tempo uma conexão de cliente pode ficar antes que o proxy a feche. O padrão é de 1.800 segundos (30 minutos).

Uma conexão de cliente é considerada ociosa quando a aplicação não envia uma nova solicitação dentro do tempo especificado após a conclusão da solicitação anterior. A conexão de banco de dados subjacente permanece aberta e é retornada ao grupo de conexões. Portanto, ela está disponível para ser reutilizada para novas conexões de cliente. Reduza o tempo limite de conexão do cliente ocioso se desejar que o proxy remova proativamente as conexões obsoletas. Se a workload estabelecer conexões frequentes com o proxy, aumente o tempo limite de conexão do cliente ocioso para reduzir o custo de estabelecimento de conexões.

Essa configuração é representada pelo Tempo limite de conexão do cliente ocioso no console do RDS e na configuração `IdleClientTimeout` na AWS CLI e a API. Para saber como alterar o valor do Tempo limite de conexão do cliente ocioso no console do RDS, consulte Console de gerenciamento da AWS. Para saber como alterar o valor da configuração `IdleClientTimeout`, use o comando da CLI modify-db-proxy ou a operação da API ModifyDBProxy.

MaxConnectionsPercent

Você pode limitar o número de conexões que um Proxy RDS pode estabelecer com o banco de dados de destino. Você especifica o limite como uma porcentagem do máximo de conexões disponíveis para o banco de dados. Essa configuração é representada pelo campo Connection pool maximum connections (Número máximo de conexões para o grupo de conexões) no console do proxy do RDS ou pelo parâmetro `MaxConnectionsPercent` na AWS CLI e na API.

O valor `MaxConnectionsPercent` é expresso como uma porcentagem da configuração `max_connections` do cluster de banco de dados do Aurora usado pelo grupo de destino. O proxy não cria todas essas conexões com antecedência. Essa configuração permite ao proxy estabelecer essas conexões conforme a necessidade da workload.

Por exemplo, para um destino de banco de dados registrado com `max_connections` definido como 1.000 e `MaxConnectionsPercent` definido como 95, o RDS Proxy define 950 conexões como o limite superior para conexões simultâneas com esse destino de banco de dados.

Um efeito colateral comum de sua workload atingir o número máximo de conexões de banco de dados permitidas é um aumento na latência geral da consulta, bem como um aumento na métrica `DatabaseConnectionsBorrowLatency`. Você pode monitorar as conexões de banco de dados usadas atualmente e o total permitido comparando as métricas `DatabaseConnections` e `MaxDatabaseConnectionsAllowed`.

Ao definir esse parâmetro, observe as seguintes práticas recomendadas:

- Permita espaço de conexão suficiente para mudanças no padrão da workload. É recomendável definir o parâmetro pelo menos 30% acima do seu uso máximo monitorado recentemente. Como o RDS Proxy redistribui as cotas de conexão do banco de dados em vários nós, as mudanças na capacidade interna podem exigir pelo menos 30% de espaço livre para conexões adicionais, a fim de evitar o aumento das latências de empréstimo.
- O RDS Proxy reserva um certo número de conexões para monitoramento ativo para comportar falhas rápidas, roteamento de tráfego e operações internas. A métrica `MaxDatabaseConnectionsAllowed` não inclui essas conexões reservadas. Ela representa o número de conexões disponíveis para atender à workload e pode ser menor do que o valor derivado da configuração `MaxConnectionsPercent`.

  Os valores mínimos `MaxConnectionsPercent` recomendados são os seguintes:

  - db.t3.small: 100
  - db.t3.medium: 55

- db.t3.large: 35
- db.r3.large ou superior: 20

Se várias instâncias de destino forem registradas com o RDS Proxy, como um cluster do Aurora com nós de leitura, defina o valor mínimo com base na menor instância registrada.

Para saber como alterar o valor Connection pool maximum connections (Número máximo de conexões para o grupo de conexões) no console do RDS, consulte Console de gerenciamento da AWS. Para saber como alterar o valor de `MaxConnectionsPercent`, consulte o comando da CLI modify-db-proxy-target-group ou a operação da API ModifyDBProxyTargetGroup.

> **Important**
>
> Se o cluster de banco de dados fizer parte de um banco de dados global com o encaminhamento de gravação ativado, reduza o valor `MaxConnectionsPercent` de seu proxy pela cota alocada para o encaminhamento de gravação. A cota de encaminhamento de gravação é definida no parâmetro do cluster de banco de dados `aurora_fwd_writer_max_connections_pct`. Para obter informações sobre o encaminhamento de gravação, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Para obter informações sobre os limites de conexões de banco de dados, consulte Número máximo de conexões com uma instância de bancos de dados Aurora MySQL e Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL.

MaxIdleConnectionsPercent

Você pode controlar o número de conexões de banco de dados ociosas que o proxy do RDS pode manter no grupo de conexões. Por padrão, o RDS Proxy considera uma conexão de banco de dados no grupo como ociosa quando não há atividade na conexão por cinco minutos.

O valor `MaxIdleConnectionsPercent` é expresso como uma porcentagem da configuração `max_connections` do grupo de destino da instância de banco de dados do RDS. O valor padrão é 50% de `MaxConnectionsPercent` e o limite superior é o valor de `MaxConnectionsPercent`. Por exemplo, se `MaxConnectionsPercent` for 80, o valor padrão de `MaxIdleConnectionsPercent` será 40.

Com um valor alto, o proxy deixa uma alta porcentagem de conexões de banco de dados ociosas abertas. Com um valor baixo, o proxy fecha uma alta porcentagem de conexões de banco de dados ociosas. Se suas workloads forem imprevisíveis, considere definir um valor alto para `MaxIdleConnectionsPercent`. Isso significa que o RDS Proxy pode acomodar picos de atividade sem abrir muitas novas conexões de banco de dados.

Essa configuração é representada pela configuração `MaxIdleConnectionsPercent` do `DBProxyTargetGroup` na AWS CLI e na API. Para saber como alterar o valor de `MaxIdleConnectionsPercent`, consulte o comando da CLI modify-db-proxy-target-group ou a operação da API ModifyDBProxyTargetGroup.

Para obter informações sobre os limites de conexões de banco de dados, consulte Número máximo de conexões com uma instância de bancos de dados Aurora MySQL e Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL.

ConnectionBorrowTimeout

É possível especificar quanto tempo o proxy do RDS espera que uma conexão de banco de dados no grupo de conexão fique disponível antes de retornar um erro de tempo limite. O padrão é 120 segundos. Esta configuração se aplica quando o número de conexões está no máximo e, portanto, nenhuma conexão está disponível no grupo de conexões. Isso também se aplica caso nenhuma instância de banco de dados apropriada esteja disponível para lidar com a solicitação, por exemplo, quando uma operação de failover está em andamento. Usando essa configuração, é possível definir o melhor período de espera para a aplicação sem precisar alterar o tempo limite da consulta no código.

Essa configuração é representada pelo campo Connection borrow timeout (Tempo limite de empréstimo de conexões) no console do RDS ou pela configuração `ConnectionBorrowTimeout` de `DBProxyTargetGroup` na AWS CLI ou API. Para saber como alterar o valor do campo Connection borrow timeout (Tempo limite de empréstimo de conexões) no console do RDS, consulte Console de gerenciamento da AWS. Para saber como alterar o valor de `ConnectionBorrowTimeout`, consulte o comando da CLI modify-db-proxy-target-group ou a operação da API ModifyDBProxyTargetGroup.

Conexões de cliente e banco de dados

As conexões do seu aplicativo com o RDS Proxy são conhecidas como conexões de cliente. As conexões de um proxy com o banco de dados são conexões de banco de dados. Ao usar o RDS Proxy, as conexões do cliente terminam no proxy, enquanto as conexões do banco de dados são gerenciadas no RDS Proxy.

O grupo de conexões do lado da aplicação pode oferecer o benefício de reduzir o estabelecimento de conexão recorrente entre a aplicação e o RDS Proxy.

Pense nos seguintes aspectos de configuração antes de implementar um grupo de conexões do lado da aplicação:

- Vida útil máxima da conexão do cliente: o RDS Proxy impõe uma vida útil máxima das conexões do cliente de 24 horas. Esse valor não é configurável. Configure o grupo com uma vida útil máxima de conexão inferior a 24 horas para evitar quedas inesperadas na conexão do cliente.
- Tempo limite de inatividade da conexão do cliente: o RDS Proxy impõe um tempo máximo de inatividade para as conexões do cliente. Configure seu pool com um tempo limite de conexão ociosa de um valor menor do que a configuração de tempo limite de inatividade da conexão do cliente para o RDS Proxy para evitar quedas inesperadas de conexão.

O número máximo de conexões de cliente configuradas no grupo de conexões do lado da aplicação não precisa ser limitado à configuração max_connections do RDS Proxy.

O grupo de conexões do cliente resulta em maior vida útil da conexão do cliente. Se suas conexões experimentarem fixação, o agrupamento de conexões de clientes pode reduzir a eficiência da multiplexação. As conexões de cliente fixas, mas ociosas, no grupo de conexões do lado da aplicação continuam mantendo uma conexão com o banco de dados e impedem que a conexão seja reutilizada por outras conexões do cliente. Analise os logs do proxy para conferir se as conexões estão sendo fixadas.

> **Note**
>
> O proxy do RDS encerra as conexões com o banco de dados depois de 24 horas, quando elas não estiverem mais em uso. O proxy executa essa ação independentemente do valor da configuração máxima de conexões ociosas.

Evitar a fixação de um RDS Proxy

A multiplexação é mais eficiente quando as solicitações de banco de dados não dependem de informações de estado de solicitações anteriores. Nesse caso, o proxy do RDS pode reutilizar uma conexão na conclusão de cada transação. Exemplos de tais informações de estado incluem a maioria das variáveis e parâmetros de configuração que você pode alterar por meio de instruções

`SET` ou `SELECT`. Por padrão, as transações SQL em uma conexão de cliente podem multiplexar entre conexões de banco de dados subjacentes.

Suas conexões com o proxy podem entrar em um estado conhecido como fixação. Quando uma conexão é fixada, cada transação posterior usa a mesma conexão de banco de dados subjacente até que a sessão termine. Outras conexões de cliente também não podem reutilizar essa conexão de banco de dados até que a sessão termine. A sessão termina quando a conexão do cliente é descartada.

O proxy do RDS automaticamente fixa uma conexão de cliente a uma conexão de banco de dados específica quando detecta uma alteração no estado de sessão que não é apropriada para outras sessões. A fixação reduz a eficácia da reutilização de conexões. Se todas ou quase todas as suas conexões forem fixadas, considere modificar o código da aplicação ou a workload para reduzir as condições que provocam a fixação.

Por exemplo, a aplicação altera uma variável de sessão ou um parâmetro de configuração. Nesse caso, as declarações posteriores poderão contar que a nova variável ou parâmetro estará em vigor. Assim, quando o proxy do RDS processa solicitações para alterar variáveis ou definições da configuração da sessão, ele fixa essa sessão para a conexão com o banco de dados. Dessa forma, o estado da sessão permanece em vigor para todas as transações posteriores na mesma sessão.

Em alguns mecanismos de banco de dados, essa regra não se aplica a todos os parâmetros que você pode definir. O proxy do RDS monitora determinadas instruções e variáveis. Portanto, o RDS Proxy não fixa a sessão quando você as modifica. Nesse caso, o proxy do RDS apenas reutiliza a conexão para outras sessões que tenham os mesmos valores para essas configurações. Para obter as listas de declarações e variáveis monitoradas do Aurora MySQL, consulte O que o RDS Proxy monitora para bancos de dados do Aurora MySQL.

O que o RDS Proxy monitora para bancos de dados do Aurora MySQL

O RDS Proxy monitora as seguintes instruções do MySQL:

- DROP DATABASE
- DROP SCHEMA
- USE

O RDS Proxy monitora as seguintes variáveis do MySQL:

- `AUTOCOMMIT`

- AUTO_INCREMENT_INCREMENT
- CHARACTER SET (or CHAR SET)
- CHARACTER_SET_CLIENT
- CHARACTER_SET_DATABASE
- CHARACTER_SET_FILESYSTEM
- CHARACTER_SET_CONNECTION
- CHARACTER_SET_RESULTS
- CHARACTER_SET_SERVER
- COLLATION_CONNECTION
- COLLATION_DATABASE
- COLLATION_SERVER
- INTERACTIVE_TIMEOUT
- NAMES
- NET_WRITE_TIMEOUT
- QUERY_CACHE_TYPE
- SESSION_TRACK_SCHEMA
- SQL_MODE
- TIME_ZONE
- TRANSACTION_ISOLATION (or TX_ISOLATION)
- TRANSACTION_READ_ONLY (or TX_READ_ONLY)
- WAIT_TIMEOUT

> **Note**
>
> O RDS Proxy rastreia as alterações nas variáveis TRANSACTION_ISOLATION e TRANSACTION_READ_ONLY quando você as define no escopo da sessão. No entanto, se você defini-las no escopo da próxima transação, o RDS Proxy fixará as conexões. Esse comportamento se aplica independentemente de você usar uma instrução SET ou uma instrução SET TRANSACTION para configurar esses valores.

Reduzir a fixação

O ajuste de performance do proxy do RDS envolve a tentativa de maximizar a reutilização de conexões em nível de transação (multiplexação) minimizando a fixação.

É possível reduzir a fixação da seguinte maneira:

- Evite solicitações de banco de dados desnecessárias que podem causar fixação.
- Defina variáveis e definições de configuração de forma consistente em todas as conexões. Dessa forma, as sessões posteriores têm mais probabilidade de reutilizar conexões que têm essas configurações específicas.

  No entanto, para a configuração do PostgreSQL, uma variável leva à fixação da sessão.
- Para um banco de dados da família de mecanismos do MySQL, aplique um filtro de fixação de sessão ao proxy. Você pode isentar certos tipos de operações de fixar a sessão se souber que isso não afeta o funcionamento correto da aplicação.
- Veja com que frequência a fixação ocorre monitorando a métrica `DatabaseConnectionsCurrentlySessionPinned` do Amazon CloudWatch. Para obter informações sobre essa e outras métricas do CloudWatch, consulte Monitorar métricas do proxy do RDS com o Amazon CloudWatch.
- Se você usar instruções `SET` para executar uma inicialização idêntica para cada conexão de cliente, poderá fazê-lo enquanto preserva a multiplexação no nível de transação. Nesse caso, você move as instruções que configuram o estado da sessão inicial para a consulta de inicialização usada por um proxy. Esta propriedade é uma string que contém uma ou mais instruções SQL, separadas por ponto e vírgula.

  Por exemplo, você pode definir uma consulta de inicialização para um proxy que define determinados parâmetros de configuração. O proxy do RDS aplica essas configurações sempre que configura uma nova conexão para esse proxy. Você pode remover as instruções `SET` correspondentes do código de sua aplicação, para que elas não interfiram na multiplexação em nível de transação.

  Para ver métricas sobre a frequência com que a fixação ocorre para um proxy, consulte Monitorar métricas do proxy do RDS com o Amazon CloudWatch.

Condições que causam fixação para todas as famílias de mecanismos

O proxy fixa a sessão à conexão atual nas seguintes situações em que a multiplexação pode causar um comportamento inesperado:

- Qualquer instrução com um tamanho de texto maior do que 16 KB faz com que o proxy fixe a sessão.

Condições que causam fixação no Aurora MySQL

Para o MySQL, as seguintes interações também geram fixação:

- As instruções explícitas de bloqueio de tabela `LOCK TABLE`, `LOCK TABLES` ou `FLUSH TABLES WITH READ LOCK` fazem com que o proxy fixe a sessão.
- A criação de bloqueios nomeados usando `GET_LOCK` faz com que o proxy fixe a sessão.
- Definir uma variável de usuário ou sistema (com algumas exceções) fixa a sessão ao proxy. Se isso limitar significativamente a reutilização da conexão, você poderá configurar as operações `SET` para evitar a fixação. Para fazer isso, ajuste a propriedade de filtros de fixação de sessão. Para obter mais informações, consulte Criar um proxy para o Amazon Aurora e Modificar um RDS Proxy.
- A criação de uma tabela temporária faz com que o proxy fixe a sessão. Dessa forma, o conteúdo da tabela temporária é preservado durante toda a sessão, independentemente dos limites de transação.
- A chamada das funções `ROW_COUNT` e `FOUND_ROWS` às vezes causa fixação.

  As circunstâncias exatas em que essas funções causam fixação podem diferir entre versões do Aurora MySQL compatíveis com o MySQL 5.7.
- As instruções preparadas fazem com que o proxy fixe a sessão. Essa regra se aplicará se a instrução preparada usar texto SQL ou o protocolo binário.
- O RDS Proxy não fixa conexões quando você usa SET LOCAL.
- A chamada de procedimentos armazenados e de funções armazenadas não causa fixação. O proxy do RDS não detecta nenhuma alteração de estado de sessão resultante dessas chamadas. Verifique se a aplicação não altera o estado da sessão dentro de rotinas armazenadas se você confia nesse estado de sessão para persistir entre transações. Por exemplo, no momento, o proxy do RDS não é compatível com um procedimento armazenado que cria uma tabela temporária que persiste em todas as transações.

- Consultas com comentários executáveis para MySQL (sintaxe /*! ... */) ou MariaDB (sintaxe /*M! ... */) causam fixação. O RDS Proxy não pode analisar o SQL incorporado nesses comentários para rastrear alterações no estado da sessão.

Se você tiver conhecimento especializado sobre o comportamento da aplicação, poderá ignorar o comportamento de fixação de determinadas instruções da aplicação. Para fazer isso, escolha a opção Filtros de fixação de sessão ao criar o proxy. Atualmente, é possível cancelar a fixação de sessão para definir variáveis de sessão e configurações.

Condições que causam fixação no Aurora PostgreSQL

Para o PostgreSQL, as seguintes interações também geram a fixação:

- Usar comandos `SET`.
- Usar comandos `PREPARE`, `DISCARD`, `DEALLOCATE` ou `EXECUTE` para gerenciar instruções preparadas.
- Criar sequências, tabelas ou visualizações temporárias.
- Declarar cursores.
- Descartar o estado da sessão.
- Escutar em um canal de notificação.
- Carregar um módulo de biblioteca, como `auto_explain`.
- Manipular sequências usando funções como `nextval` e `setval`.
- Interagir com bloqueios usando funções como `pg_advisory_lock` e `pg_try_advisory_lock`.

> **Note**
>
> O RDS Proxy não fixa bloqueios consultivos em nível de transação, especificamente `pg_advisory_xact_lock`, `pg_advisory_xact_lock_shared`, `pg_try_advisory_xact_lock` e `pg_try_advisory_xact_lock_shared`.

- Definir ou redefinir um parâmetro como o padrão. Especificamente, usar comandos `SET` e `set_config` para atribuir valores padrão às variáveis da sessão.
- A chamada de procedimentos armazenados e de funções armazenadas não causa fixação. O proxy do RDS não detecta nenhuma alteração de estado de sessão resultante dessas chamadas. Verifique se a aplicação não altera o estado da sessão dentro de rotinas armazenadas se você

confia nesse estado de sessão para persistir entre transações. Por exemplo, no momento, o proxy do RDS não é compatível com um procedimento armazenado que cria uma tabela temporária que persiste em todas as transações.

- Descartar estado da sessão. Se você usar bibliotecas de pool de conexões com a consulta `DISCARD ALL` configurada como uma consulta de redefinição, o RDS Proxy fixará sua conexão de cliente na liberação. Isso reduz a eficiência da multiplexação do proxy e pode levar a resultados inesperados, pois o comando `DISCARD ALL` pode interferir no gerenciamento do estado da sessão.

Excluir um RDS Proxy

Será possível excluir um proxy quando não precisar mais dele. Ou poderá excluir um proxy se você tirar de serviço a instância ou o cluster de banco de dados associado.

Console de gerenciamento da AWS

Como excluir um proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Escolha o proxy a ser excluído da lista.
4. Escolha Delete Proxy (Excluir proxy).

AWS CLI

Para excluir um proxy de banco de dados, use o comando delete-db-proxy da AWS CLI. Para remover associações relacionadas, use também o comando deregister-db-proxy-targets.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

API do RDS

Para excluir um proxy de banco de dados, chame a função DeleteDBProxy da API do Amazon RDS. Para excluir itens e associações relacionados, você também chama as funções DeleteDBProxyTargetGroup e DeregisterDBProxyTargets.

Como trabalhar com endpoints do proxy do Amazon RDS

Os endpoints do RDS Proxy oferecem maneiras flexíveis e eficientes de gerenciar conexões de banco de dados, o que melhora a escalabilidade, disponibilidade e segurança. Com endpoints de proxy, é possível:

- Simplificar o monitoramento e a solução de problemas: use vários endpoints para monitorar e gerenciar conexões de diferentes aplicações de maneira independente.
- Melhorar a escalabilidade de leitura: aproveite os endpoints de leitura com clusters de banco de dados do Aurora para distribuir o tráfego de leitura com eficiência e, desse modo, otimizar a performance para workloads com muitas consultas.
- Habilitar o acesso entre VPCs: conecte bancos de dados entre VPCs usando endpoints entre VPCs, o que permite que recursos, como instâncias do Amazon EC2 em uma VPC, acessem bancos de dados em outra.

Tópicos

- Visão geral dos endpoints de proxy
- Limitações de endpoints de proxy
- Uso de endpoints de leitor com clusters do Aurora
- Acessar bancos de dados Aurora entre VPCs
- Criação de um endpoint de proxy
- Visualização dos endpoints de proxy
- Modificação de um endpoint de proxy
- Exclusão de um endpoint de proxy

Visão geral dos endpoints de proxy

Trabalhar com endpoints de proxy do RDS envolve os mesmos tipos de procedimentos usados com endpoints de leitor e cluster de bancos de dados do Aurora. Se você não tem familiaridade com os endpoints do Aurora, encontre mais informações em Conexões de endpoints do Amazon Aurora.

Quando você usa o RDS Proxy com um cluster do Aurora, o endpoint padrão comporta operações de leitura e gravação. Isso significa que ele encaminha todas as solicitações à instância de gravador do cluster, contribuindo com o respectivo limite de `max_connections`. Para distribuir melhor o tráfego, é possível criar endpoints adicionais de leitura/gravação ou somente leitura para o proxy, o que possibilita aumentar a eficiência do gerenciamento da workload e melhorar a escalabilidade.

Use um endpoint somente leitura com seu proxy para lidar com consultas de leitura, da mesma forma que você faria com um endpoint de leitor para um cluster provisionado do Aurora. Essa abordagem maximiza a escalabilidade de leitura do Aurora ao distribuir consultas em uma ou mais instâncias de banco de dados de leitor. Ao usar um endpoint somente leitura e adicionar mais instâncias de leitor conforme necessário, você pode aumentar o número de consultas simultâneas e conexões que o cluster pode processar.

> **Tip**
>
> Quando você cria um proxy para um cluster do Aurora usando o Console de gerenciamento da AWS, pode fazer com que o RDS Proxy crie automaticamente um endpoint do leitor. Para obter informações sobre os benefícios de um endpoint de leitor, consulte Uso de endpoints de leitor com clusters do Aurora.

Ao criar um endpoint de proxy, você pode associá-lo a uma nuvem privada virtual (VPC) diferente da utilizada pelo proxy. Isso possibilita que você se conecte ao proxy por meio de outra VPC, como uma usada por uma aplicação diferente em sua organização.

Para obter informações sobre limites associados aos endpoints de proxy, consulte Limitações de endpoints de proxy.

Os logs do RDS Proxy incluem como prefixo em cada entrada o nome do endpoint de proxy associado. Pode ser o nome especificado para um endpoint definido pelo usuário ou o nome especial `default` para o endpoint padrão de leitura/gravação do proxy.

Cada endpoint de proxy tem seu próprio conjunto de métricas do CloudWatch. Monitore métricas de todos os endpoints específicos, de um endpoint específico ou de todos os endpoints de leitura/

gravação ou somente leitura de um proxy. Para obter mais informações, consulte Monitorar métricas do proxy do RDS com o Amazon CloudWatch.

Um endpoint de proxy usa o mesmo mecanismo de autenticação que o proxy associado. O proxy do RDS configura automaticamente permissões e autorizações para o endpoint definido pelo usuário, consistentes com as propriedades do proxy associado.

Para saber como os endpoints de proxy funcionam para clusters de banco de dados em um banco de dados global do Aurora, consulte Como os endpoints do RDS Proxy funcionam com bancos de dados globais.

Limitações de endpoints de proxy

Os endpoints do RDS Proxy têm as seguintes limitações:

- O endpoint padrão do proxy do RDS não pode ser modificado.
- O número máximo de endpoints definidos pelo usuário para um proxy é 20. Assim, um proxy pode ter até 21 endpoints: o endpoint padrão e mais 20 que você cria.
- Quando você associa endpoints adicionais a um proxy, o proxy do RDS determina automaticamente quais instâncias de banco de dados em seu cluster usar para cada endpoint. Você não pode escolher instâncias específicas da mesma maneira que escolhe endpoints personalizados do Aurora.
- Para tipos de rede de endpoint IPv6 ou de pilha dupla, sua VPC e sub-redes devem ser configuradas para comportar o tipo de rede selecionado.

Quando você cria um proxy, o RDS cria automaticamente um endpoint da VPC para comunicação segura entre as aplicações e o banco de dados. O endpoint da VPC é visível e pode ser acessado no console da Amazon VPC.

A adição de um novo endpoint de proxy provisiona um endpoint de interface do AWS PrivateLink. Se você adicionar um ou mais endpoints ao proxy, incorrerá em cobranças adicionais. Para ter mais informações, consulte Preços do Amazon Relational Database Service Proxy.

Uso de endpoints de leitor com clusters do Aurora

Você pode criar e se conectar a endpoints somente leitura chamados endpoints de leitor quando usa o proxy do RDS com clusters do Aurora. Esses endpoints de leitor ajudam a melhorar a escalabilidade de leitura de suas aplicações com uso intensivo de consultas. Os endpoints de leitor

também ajudam a melhorar a disponibilidade de suas conexões, se uma instância de banco de dados do leitor no cluster ficar indisponível.

> **Note**
>
> Quando você especifica que um novo endpoint é somente leitura, é necessário para o proxy do RDS que o cluster do Aurora tenha uma ou mais instâncias de banco de dados do leitor. Em alguns casos, é possível alterar o destino do proxy para um cluster do Aurora que contenha um único gravador. Se fizer isso, qualquer solicitação para o endpoint do leitor falhará com um erro. As solicitações também falham se o destino do proxy for uma instância do RDS em vez de um cluster do Aurora.
> Se um cluster do Aurora tem instâncias de leitor, mas essas instâncias não estão disponíveis, o proxy do RDS aguarda para enviar a solicitação em vez de retornar um erro imediatamente. Se nenhuma instância de leitor ficar disponível dentro do tempo limite de empréstimo da conexão, a solicitação falhará com um erro.

Como os endpoints de leitor ajudam a disponibilidade das aplicações

Em alguns casos, uma ou mais instâncias de leitor no cluster podem ficar indisponíveis. Nesses casos, as conexões que usam um endpoint leitor de um proxy de banco de dados podem se recuperar mais rapidamente do que aquelas que usam o endpoint leitor do Aurora. O proxy do RDS roteia conexões somente para as instâncias de leitor disponíveis no cluster. Não há um atraso causado pelo cache de DNS quando uma instância fica indisponível.

Se a conexão for multiplexada, o proxy do RDS direciona as consultas subsequentes para uma instância de banco de dados de leitor diferente, sem qualquer interrupção na aplicação. Durante a alternância automática para uma nova instância de leitor, o proxy do RDS verifica o atraso de replicação das instâncias antigas e novas do leitor. O proxy do RDS garante que a nova instância do leitor esteja atualizada com as mesmas alterações que a instância do leitor anterior. Dessa forma, sua aplicação nunca vê dados obsoletos quando o proxy do RDS muda de uma instância de banco de dados do leitor para outra.

Se a conexão estiver fixada, a próxima consulta na conexão retornará um erro. Porém, sua aplicação pode se reconectar imediatamente ao mesmo endpoint. O proxy do RDS roteia a conexão para uma instância de banco de dados de leitor diferente que esteja no estado `available`. Quando você se reconecta manualmente, o proxy do RDS não verifica o atraso de replicação entre as instâncias antigas e novas do leitor.

Se o seu cluster do Aurora não tiver nenhuma instância de leitor disponível, o proxy do RDS verificará se essa condição é temporária ou permanente. O comportamento em cada caso é o seguinte:

- Suponha que seu cluster tenha uma ou mais instâncias de banco de dados do leitor, mas nenhuma delas esteja no estado `Available`. Por exemplo, todas as instâncias do leitor podem estar reinicializando ou com problemas. Nesse caso, as tentativas de se conectar a um endpoint de leitor aguardam até que uma instância de leitor fique disponível. Se nenhuma instância de leitor ficar disponível dentro do tempo limite de empréstimo da conexão, a tentativa de conexão falhará. Se uma instância de leitor ficar disponível, a tentativa de conexão será bem-sucedida.
- Suponha que seu cluster não tenha instâncias de banco de dados de leitor. Nesse caso, o proxy do RDS retorna um erro imediatamente se você tentar se conectar a um endpoint de leitor. Para resolver esse problema, adicione uma ou mais instâncias de leitor ao cluster antes de se conectar ao endpoint de leitor.

Como os endpoints de leitor contribuem para a escalabilidade da consulta

Os endpoints de leitor para um proxy ajudam contribuem para a escalabilidade da consulta do Aurora das seguintes maneiras:

- À medida que você adiciona instâncias de leitor ao cluster do Aurora, o proxy do RDS pode rotear novas conexões para todos os endpoints de leitor para as diferentes instâncias do leitor. Dessa forma, as consultas realizadas usando uma conexão de endpoint de leitor não retardam as consultas realizadas usando outra conexão de endpoint de leitor. As consultas são executadas em instâncias de banco de dados separadas. Cada instância de banco de dados tem seus próprios recursos de computação, cache de buffer e assim por diante.
- Onde for prático, o proxy do RDS usa a mesma instância de banco de dados de leitor para todos os problemas de consultas usando uma conexão de endpoint de leitor específica. Dessa forma, um conjunto de consultas relacionadas nas mesmas tabelas pode aproveitar o cache, a otimização do plano e assim por diante em uma instância de banco de dados específica.
- Se uma instância de banco de dados de leitor ficar indisponível, o efeito sobre a aplicação dependerá se a sessão for multiplexada ou fixada. Se a sessão for multiplexada, o proxy do RDS roteia todas as consultas subsequentes para uma instância de banco de dados de leitor diferente sem qualquer ação de sua parte. Se a sessão estiver fixada, a aplicação receberá um erro e deverá se reconectar. Você pode se reconectar ao endpoint de leitor imediatamente e o proxy do RDS roteará a conexão para uma instância de banco de dados de leitor disponível. Para obter

mais informações sobre multiplexação e fixação para sessões de proxy, consulte Visão geral dos conceitos do RDS Proxy.

- Quanto mais instâncias de banco de dados de leitor você tiver no cluster, mais conexões simultâneas você poderá fazer usando endpoints de leitor. Por exemplo, suponha que seu cluster tenha quatro instâncias de banco de dados de leitor, cada uma configurada para permitir 200 conexões simultâneas. Suponha que seu proxy esteja configurado para usar 50% das conexões máximas. Aqui, o número máximo de conexões que você pode fazer através dos endpoints de leitor no proxy é 100 (50% de 200) para o leitor 1. Também é 100 para leitor 2, e assim por diante, até um total de 400. Se você dobrar o número de instâncias de banco de dados do leitor no cluster para oito, o número máximo de conexões através dos endpoints do leitor também será duplicado para 800.

Exemplos de uso de endpoints de leitor

O exemplo do Linux a seguir mostra como você pode confirmar que está conectado a um cluster do Aurora MySQL através de um endpoint de leitor. A configuração `innodb_read_only` está definida como habilitada. Tentativas de executar operações de gravação, como `CREATE DATABASE`, falham com um erro. E você pode confirmar se você está conectado a uma instância de banco de dados do leitor ao verificar o nome da instância de banco de dados usando a variável `aurora_server_id`.

> **Tip**
>
> Não confie apenas na verificação do nome da instância de banco de dados para determinar se a conexão é leitura/gravação ou somente leitura. Lembre-se de que as instâncias de banco de dados em um cluster do Aurora podem alterar as funções entre o gravador e o leitor quando os failovers ocorrem.

```
$ mysql -h endpoint-demo-reader.endpoint.proxy-demo.us-east-1.rds.amazonaws.com -u
 admin -p
...
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
mysql> create database shouldnt_work;
```

```
ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it
 cannot execute this statement

mysql> select @@aurora_server_id;
+---------------------------------------+
| @@aurora_server_id                    |
+---------------------------------------+
| proxy-reader-endpoint-demo-instance-3 |
+---------------------------------------+
```

O exemplo a seguir mostra como sua conexão com um endpoint de leitor de proxy pode continuar funcionando mesmo quando a instância de banco de dados do leitor é excluída. Neste exemplo, o cluster do Aurora tem duas instâncias de leitor, `instance-5507` e `instance-7448`. A conexão com o endpoint de leitor começa a usar uma das instâncias do leitor. Durante o exemplo, essa instância do leitor é excluída por meio de um comando `delete-db-instance`. O proxy do RDS alterna para uma instância de leitor diferente para consultas subsequentes.

```
$ mysql -h reader-demo.endpoint.proxy-demo.us-east-1.rds.amazonaws.com
  -u my_user -p
...
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-5507      |
+--------------------+

mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+

mysql> select count(*) from information_schema.tables;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Enquanto a sessão do `mysql` continua em execução, o comando a seguir exclui a instância do leitor à qual o endpoint de leitor está conectado.

```
aws rds delete-db-instance --db-instance-identifier instance-5507 --skip-final-snapshot
```

As consultas da sessão `mysql` continuam funcionando sem a necessidade de reconexão. O proxy do RDS alterna automaticamente para uma instância de banco de dados de leitor diferente.

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-7448      |
+--------------------+

mysql> select count(*) from information_schema.TABLES;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Acessar bancos de dados Aurora entre VPCs

Por padrão, os componentes da pilha de tecnologia do Aurora estão todos na mesma Amazon VPC. Por exemplo, suponha que uma aplicação em execução em uma instância do Amazon EC2 se conecte a um cluster de banco de dados do Aurora. Nesse caso, o servidor da aplicação e o banco de dados devem estar dentro da mesma VPC.

Com o RDS Proxy, é possível configurar o acesso a um cluster de banco de dados do Aurora em uma VPC a partir de recursos em outra VPC, como instâncias do EC2. Por exemplo, sua organização pode ter várias aplicações que acessam os mesmos recursos de banco de dados. Cada aplicação pode estar em sua própria VPC.

Para habilitar o acesso entre VPCs, crie um novo endpoint para o proxy. O proxy em si reside na mesma VPC que o cluster de bancos de dados do Aurora. No entanto, o endpoint entre VPCs reside na outra VPC, juntamente com os outros recursos, como as instâncias do EC2. O endpoint entre VPC está associado a sub-redes e grupos de segurança da mesma VPC que o EC2 e outros recursos. Essas associações permitem que você se conecte ao endpoint a partir das aplicações que, de outra forma, não podem acessar o banco de dados devido às restrições da VPC.

As etapas a seguir explicam como criar e acessar um endpoint entre VPCs com o RDS Proxy:

1. Crie duas VPCs ou escolha duas VPCs que você já usa para trabalhar com o Aurora . Cada VPC deve ter os próprios recursos de rede associados, como um gateway da Internet, tabelas de roteamento, sub-redes e grupos de segurança. Se você tiver apenas uma VPC, consulte Conceitos básicos do Amazon Aurora para ver as etapas de configuração de outra VPC, a fim de usar o Aurora com êxito. Também é possível examinar a VPC existente no console do Amazon EC2 para ver quais tipos de recursos podem ser conectados.
2. Crie um proxy de banco de dados associado ao cluster de bancos de dados do Aurora a que você deseja se conectar. Siga o procedimento em Criar um proxy para o Amazon Aurora.
3. Na página Details (Detalhes) do seu proxy no console do RDS, na seção Proxy endpoints (Endpoints de proxy), escolha Create endpoint (Criar endpoint). Siga o procedimento em Criação de um endpoint de proxy.
4. Escolha se deseja criar o endpoint entre VPCs como leitura/gravação ou somente leitura.
5. Em vez de aceitar o padrão da mesma VPC que o cluster de bancos de dados do Aurora, escolha uma VPC diferente. Essa VPC deve estar na mesma região da AWS que a VPC, onde o proxy reside.
6. Agora, em vez de aceitar os padrões para sub-redes e grupos de segurança da mesma VPC que o cluster de bancos de dados do Aurora, faça novas seleções. Faça-as com base nas sub-redes e grupos de segurança da VPC que você escolheu.
7. Você não precisa alterar nenhuma das configurações para os segredos do Secrets Manager. As mesmas credenciais funcionam para todos os endpoints de proxy, independentemente da VPC em que cada endpoint esteja. Da mesma forma, ao usar a autenticação do IAM, a configuração e as permissões do IAM funcionam de forma consistente em todos os endpoints de proxy, mesmo quando os endpoints estão em VPCs diferentes. Nenhuma configuração adicional do IAM é necessária por endpoint.
8. Aguarde até que o novo endpoint alcance o estado Available (Disponível).
9. Anote o nome completo do endpoint. Esse é o valor que termina em `*Region_name*.rds.amazonaws.com`, que você fornece como parte da string de conexão para sua aplicação de banco de dados.
10. Acesse o novo endpoint por meio de um recurso na mesma VPC que o endpoint. Uma maneira simples de testar esse processo é criar uma nova instância do EC2 nessa VPC. Depois, faça login na instância do EC2 e execute os comandos `mysql` ou `psql` para se conectar usando o valor do endpoint na string de conexão.

Criação de um endpoint de proxy

Para criar um endpoint de proxy, siga estas instruções:

Console

Para criar um endpoint de proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Clique no nome do proxy para o qual você deseja criar um novo endpoint.

   A página de detalhes desse proxy é exibida.
4. Na seção Proxy endpoints (Endpoints do proxy), escolha Create proxy endpoint (Criar endpoint proxy).

   A janela Create proxy endpoint (Criar endpoint proxy) aparece.
5. em Proxy endpoint name (Nome do endpoint de proxy), insira um nome descritivo de sua escolha.
6. Em Target role (Função de destino), escolha se deseja criar o endpoint como leitura/gravação ou somente leitura.

   As conexões que usam endpoints de leitura/gravação podem executar qualquer tipo de operação: declarações DDL (linguagem de definição de dados), declarações DML (linguagem de manipulação de dados) e consultas. Esses endpoints sempre se conectam à instância principal do cluster do Aurora. Você pode usar endpoints de leitura/gravação para operações gerais de banco de dados quando você usa apenas um único endpoint em sua aplicação. Você também pode usar endpoints de leitura/gravação para operações administrativas, aplicações de processamento de transações on-line (OLTP) e trabalhos de extração de transformação de carga (ETL).

   As conexões que usam um endpoint somente leitura só podem executar consultas. Quando há várias instâncias de leitor no cluster do Aurora, o proxy do RDS pode usar uma instância de leitor diferente para cada conexão com o endpoint. Dessa forma, uma aplicação com uso intensivo de consultas pode tirar proveito da capacidade de clustering do Aurora. Você pode acrescentar mais capacidade de consulta ao cluster adicionando mais instâncias de banco de dados de leitor. Essas conexões somente leitura não impõem nenhuma sobrecarga na instância

primária do cluster. Dessa forma, suas consultas de relatórios e análises não retardam as operações de gravação de suas aplicações OLTP.

7. Em Nuvem privada virtual (VPC), selecione o padrão para acessar o endpoint das mesmas instâncias do EC2 ou outros recursos que normalmente são usados para acessar o proxy ou o banco de dados associado. Para configurar o acesso entre VPC para esse proxy, selecione uma VPC diferente da padrão. Para obter mais informações sobre acesso entre VPCs, consulte Acessar bancos de dados Aurora entre VPCs.
8. Para o tipo de rede de endpoint, escolha a versão do IP para o endpoint de proxy. As opções disponíveis são:

   - IPv4: o endpoint de proxy usa somente endereços IPv4 (padrão).
   - IPv6: o endpoint de proxy usa somente endereços IPv6.
   - Pilha dupla: o endpoint de proxy comporta endereços IPv4 e IPv6.

   Para usar IPv6 ou pilha dupla, sua VPC e sub-redes devem ser configuradas para comportar o tipo de rede selecionado.
9. Em Subnets (Sub-redes), o proxy do RDS preenche as mesmas sub-redes que o proxy associado por padrão. Para restringir o acesso ao endpoint para que apenas uma parte do intervalo de endereços da VPC possa se conectar a ele, remova uma ou mais sub-redes.
10. Em VPC grupo de segurança (Grupo de segurança da VPC), você pode selecionar um grupo de segurança existente ou criar outro. O proxy do RDS preenche os mesmos grupos de segurança que o proxy associado por padrão. Se as regras de entrada e saída para o proxy forem apropriadas para esse endpoint, deixe a escolha padrão.

    Se você optar por criar um novo grupo de segurança, especifique um nome para o grupo de segurança nessa página. Depois, edite as configurações do grupo de segurança no console do EC2.
11. Escolha Create proxy endpoint (Criar endpoint de proxy).

AWS CLI

Para criar um endpoint de proxy, use o comando da AWS CLI create-db-proxy-endpoint.

Inclua os seguintes parâmetros necessários:

- `--db-proxy-name` *value*

- `--db-proxy-endpoint-name value`
- `--vpc-subnet-ids list_of_ids`. Separe os IDs de sub-rede com espaços. Você não especifica o ID da própria VPC.

Você também pode incluir os seguintes parâmetros opcionais:

- `--target-role { READ_WRITE | READ_ONLY }`. Por padrão, esse parâmetro é `READ_WRITE`. O valor `READ_ONLY` só tem um efeito sobre os clusters do Aurora provisionados que contêm uma ou mais instâncias de banco de dados do leitor. Quando o proxy está associado a um cluster do Aurora que contém apenas uma instância de banco de dados de gravador, não é possível especificar `READ_ONLY`. Para ter mais informações sobre o uso pretendido de endpoints somente leitura com clusters do Aurora, consulte Uso de endpoints de leitor com clusters do Aurora .
- `--vpc-security-group-ids value`. Separe os IDs do grupo de segurança com espaços. Se você omitir esse parâmetro, o proxy do RDS usará o grupo de segurança padrão para a VPC. O proxy do RDS determina a VPC com base nos IDs de sub-rede que você especificar para o parâmetro `--vpc-subnet-ids`.
- `--endpoint-network-type { IPV4 | IPV6 | DUAL }`. Esse parâmetro especifica a versão do IP do endpoint de proxy. O padrão é “”. `IPV4`. Para usar `IPV6` ou `DUAL`, sua VPC e sub-redes devem ser configuradas para comportar o tipo de rede selecionado.

Example

O exemplo a seguir cria um endpoint de proxy chamado `my-endpoint`.

Para Linux, macOS ou Unix:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id \
  --endpoint-network-type DUAL
```

Para Windows:

```
aws rds create-db-proxy-endpoint ^
```

```
  --db-proxy-name my-proxy ^
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id ^
  --endpoint-network-type DUAL
```

API do RDS

Para criar um endpoint de proxy, use a ação da API do RDS CreateDBProxyEndpoint.

Visualização dos endpoints de proxy

Para visualizar os endpoints de proxy existentes, siga estas instruções:

Console

Para visualizar os detalhes de um endpoint de proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Na lista, escolha o proxy cujo endpoint você deseja visualizar. Clique no nome do proxy para visualizar sua página de detalhes.
4. Na seção Proxy endpoints (Endpoints de proxy), escolha o endpoint que você deseja visualizar. Clique no nome dele para ver a página de detalhes.
5. Examine os parâmetros cujos valores lhe interessam. Você pode verificar propriedades como as seguintes:

   - Se o endpoint é leitura/gravação ou somente leitura.
   - O endereço do endpoint que você usa em uma string de conexão de banco de dados.
   - A VPC, as sub-redes e os grupos de segurança associados a um endpoint.

AWS CLI

Para visualizar um ou mais endpoints de proxy, use o comando da AWS CLI describe-db-proxy-endpoints.

Você pode incluir os seguintes parâmetros opcionais:

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

O exemplo a seguir descreve o endpoint de proxy `my-endpoint`.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Para Windows:

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

API do RDS

Para descrever um ou mais endpoints de proxy, use a operação da API do RDS DescribeDBProxyEndpoints.

Modificação de um endpoint de proxy

Para modificar os endpoints de proxy, siga estas instruções:

Console

Para modificar um ou mais endpoints de proxy

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Proxies.
3. Na lista, escolha o proxy cujo endpoint você deseja modificar. Clique no nome do proxy para visualizar sua página de detalhes.
4. Em Proxy endpoints (Endpoints de proxy), escolha o endpoint que você deseja modificar. Você pode selecioná-lo na lista ou clicar no nome dele para visualizar a página de detalhes.
5. Na página de detalhes do proxy, na seção Proxy endpoints (Endpoints de proxy), escolha Edit (Editar). Ou na página de detalhes do endpoint de proxy, em Ações, selecione Editar.

6. Altere os valores dos parâmetros que você deseja modificar.
7. Escolha Save changes (Salvar alterações).

AWS CLI

Para modificar um endpoint de proxy, use o comando da AWS CLI modify-db-proxy-endpoint com os seguintes parâmetros obrigatórios:

- `--db-proxy-endpoint-name`

Especifique alterações nas propriedades do endpoint usando um ou mais dos seguintes parâmetros:

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. Separe os IDs do grupo de segurança com espaços.

O exemplo a seguir renomeia o endpoint de proxy `my-endpoint` para `new-endpoint-name`.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

Para Windows:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

API do RDS

Para modificar um endpoint de proxy, use a operação da API do RDS ModifyDBProxyEndpoint.

Exclusão de um endpoint de proxy

Para excluir um endpoint para o proxy, siga estas instruções:

> **Note**
>
> Não é possível excluir o endpoint de proxy padrão que o RDS Proxy cria automaticamente para cada proxy.
> Quando você exclui um proxy, o proxy do RDS exclui automaticamente todos os endpoints associados.

Console

Para excluir um endpoint de proxy usando o Console de gerenciamento da AWS

1. No painel de navegação, escolha Proxies.
2. Na lista, escolha o proxy cujo endpoint você deseja endpoint. Clique no nome do proxy para visualizar sua página de detalhes.
3. Em Proxy endpoints (Endpoints de proxy), escolha o endpoint que você deseja excluir. Você pode selecionar um ou mais endpoints na lista ou clicar no nome de um único endpoint para visualizar a página de detalhes.
4. Na página de detalhes do proxy, na seção Proxy endpoints (Endpoints de proxy), escolha Delete (Excluir). Ou, na página de detalhes do endpoint de proxy, em Ações, selecione Excluir.

AWS CLI

Para excluir um endpoint de proxy, execute o comando delete-db-proxy-endpoint com os seguintes parâmetros obrigatórios:

- `--db-proxy-endpoint-name`

O comando a seguir exclui o endpoint de proxy chamado `my-endpoint`.

Para Linux, macOS ou Unix:

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Para Windows:

```
aws rds delete-db-proxy-endpoint ^
```

```
  --db-proxy-endpoint-name my-endpoint
```

API do RDS

Para excluir um endpoint deproxy com a API do RDS, execute a operação DeleteDBProxyEndpoint. Especifique o nome do endpoint de proxy para o parâmetro `DBProxyEndpointName`.

Monitorar métricas do proxy do RDS com o Amazon CloudWatch

É possível monitorar o proxy do RDS usando o Amazon CloudWatch O CloudWatch coleta e processa dados brutos dos proxies e os transforma em métricas legíveis quase em tempo real. Para localizar essas métricas no console do CloudWatch, escolha Metrics (Métricas) e escolha RDS e Per Proxy Metrics (Métricas por Proxy). Para obter mais informações, consulte Usando métricas do Amazon CloudWatch no Guia do usuário do Amazon CloudWatch.

> **Note**
>
> O RDS publica essas métricas para cada instância subjacente do Amazon EC2 associada a um proxy. Um único proxy pode ser atendido por mais de uma instância do EC2. Use as estatísticas do CloudWatch para agregar os valores de um proxy em todas as instâncias associadas.
> Algumas dessas métricas podem não ser visíveis até depois da primeira conexão bem-sucedida por um proxy.

Nos logs do RDS Proxy, cada entrada é prefixada com o nome do endpoint de proxy associado. Esse nome pode ser o nome especificado para um endpoint definido pelo usuário ou o nome especial `default` para o endpoint padrão de um proxy que realiza solicitações de leitura/gravação.

Todas as métricas do proxy do RDS estão no grupo `proxy`.

Cada endpoint de proxy tem suas próprias métricas do CloudWatch. Você pode monitorar o uso de cada endpoint de proxy de forma independente. Para obter mais informações sobre os endpoints do proxy, consulte Como trabalhar com endpoints do proxy do Amazon RDS.

Você pode agregar os valores de cada métrica usando um dos seguintes conjuntos de dimensões. Por exemplo, usando o conjunto de dimensões `ProxyName` , você pode analisar todo o tráfego para um determinado proxy. Ao usar os outros conjuntos de dimensões, você pode dividir as métricas de

maneiras diferentes. Você pode dividir as métricas com base nos diferentes endpoints ou bancos de dados de destino de cada proxy ou no tráfego de leitura/gravação e somente leitura para cada banco de dados.

- Conjunto de dimensões 1 : `ProxyName`
- Conjunto de dimensões 2 : `ProxyName`, `EndpointName`
- Conjunto de dimensões 3 :`ProxyName`,`TargetGroup`,`Target`
- Conjunto de dimensões 4 :`ProxyName`,`TargetGroup`,`TargetRole`

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`AvailabilityPercentage`	O percentual de tempo para o qual o grupo de destino estava disponível na função indicada pela dimensão. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 4
`ClientConnections`	O número atual de conexões de cliente. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 2
`ClientConnectionsClosed`	O número de conexões de cliente fechadas. A estatísti	1 minuto e acima	Dimension set 1, Dimension set 2

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
	ca mais útil para essa métrica é Sum.		
ClientConnectionsInSetup	O número atual de conexões de cliente abertas, mas que não concluíram a configuração. Essa métrica é relatada a cada minuto. A estatística mais útil para esta métrica é Soma.	1 minuto	Dimension set 1, Dimension set 2
ClientConnectionsNoTLS	O número atual de conexões de cliente sem Transport Layer Security (TLS). Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é Sum.	1 minuto	Dimension set 1, Dimension set 2
ClientConnectionsReceived	O número de solicitações de conexão de cliente recebidas. A estatística mais útil para essa métrica é Sum.	1 minuto e acima	Dimension set 1, Dimension set 2

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`ClientConnectionsSetupFailedAuth`	O número de tentativas de conexão do cliente que falharam devido à configuração incorreta da autenticação ou do TLS. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2
`ClientConnectionsSetupSucceeded`	O número de conexões de cliente estabelecido com êxito com qualquer mecanismo de autenticação com ou sem TLS. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2
`ClientConnectionsTLS`	O número atual de conexões de cliente com TLS. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 2

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`DatabaseConnectionRequests`	O número de solicitações para criar uma conexão de banco de dados. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionRequestsWithTLS`	O número de solicitações para criar uma conexão de banco de dados com TLS. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnections`	O número atual de conexões de banco de dados. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsBorrowLatency`	O tempo, em microssegundos, que leva para o proxy que está sendo monitorado obter uma conexão de banco de dados. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`DatabaseConnectionsCurrentlyBorrowed`	O número atual de conexões de banco de dados no estado de empréstimo. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlyInTransaction`	O número atual de conexões de banco de dados em uma transação. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlySessionPinned`	O número atual de conexões de banco de dados atualmente fixadas devido a operações em solicitações de cliente que alteram o estado da sessão. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`DatabaseConnectionsSetupFailed`	O número de solicitações de conexão do banco de dados que falharam. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsSetupSucceeded`	O número de conexões de banco de dados estabelecidas com êxito com ou sem TLS. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsWithTLS`	O número atual de conexões de banco de dados com TLS. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`MaxDatabaseConnectionsAllowed`	O número máximo de conexões de banco de dados permitidas. Essa métrica é relatada a cada minuto. A estatística mais útil para essa métrica é `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`QueryDatabaseResponseLatency`	O tempo, em microssegundos, que o banco de dados levou para responder à consulta. A estatística mais útil para essa métrica é `Average`.	1 minuto e acima	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4
`QueryRequests`	O número de consultas recebidas. Uma consulta incluindo várias instruções é contada como uma consulta. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2
`QueryRequestsNoTLS`	O número de consultas recebidas de conexões não TLS. Uma consulta incluindo várias instruções é contada como uma consulta. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2

Métrica	Descrição	Período válido	Conjunto de dimensões do CloudWatch
`QueryRequestsTLS`	O número de consultas recebidas de conexões TLS. Uma consulta incluindo várias instruções é contada como uma consulta. A estatística mais útil para essa métrica é `Sum`.	1 minuto e acima	Dimension set 1, Dimension set 2
`QueryResponseLatency`	O tempo, em microssegundos, entre a obtenção de uma solicitação de consulta e a resposta do proxy a ela. A estatística mais útil para essa métrica é `Average`.	1 minuto e acima	Dimension set 1, Dimension set 2

Você pode encontrar os logs de atividades do proxy do RDS sob CloudWatch no Console de gerenciamento da AWS. Cada proxy tem uma entrada na página Log groups (Grupos de logs).

Important

Esses logs são destinados ao consumo humano para fins de solução de problemas e não para acesso programático. O formato e o conteúdo dos logs estão sujeitos a alterações. Em particular, os logs mais antigos não contêm nenhum prefixo indicando o endpoint para cada solicitação. Nos logs mais recentes, cada entrada é prefixada com o nome do endpoint do proxy associado. Esse nome pode ser o nome que você especificou para um endpoint

definido pelo usuário ou o nome especial `default` para solicitações que usam o endpoint padrão de um proxy.

Trabalhar com eventos do RDS Proxy

Um evento indica uma alteração em um ambiente, como um ambiente da AWS, um serviço ou uma aplicação de um parceiro de software como serviço (SaaS). Ou pode ser uma de suas próprias aplicações ou serviços personalizados. Por exemplo, o Amazon Aurora gera um evento quando você cria ou modifica um RDS Proxy. O Amazon Aurora entrega eventos ao Amazon EventBridge quase em tempo real. A seguir, você pode encontrar uma lista de eventos do proxy do RDS que você pode assinar e um exemplo de um evento do RDS Proxy.

Para obter mais informações sobre como trabalhar com eventos, consulte o seguinte no:

- Para obter instruções sobre como visualizar eventos usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS, consulte Visualizar eventos do Amazon RDS.
- Para saber como configurar o Amazon Aurora para enviar eventos ao EventBridge, consulte Criar uma regra que é acionada em um evento do Amazon Aurora.

Eventos do RDS Proxy

As tabelas a seguir mostram a categoria de evento e uma lista de eventos quando um proxy do RDS é o tipo de fonte.

Categoria	ID do evento do RDS	Message	Observações
alteração de configuração	RDS-EVENT-0204	Proxy de banco de dados *nome* modificado pelo RDS.	Nenhum
alteração de configuração	RDS-EVENT-0207	O RDS modificou o endpoint do proxy de banco de dados *nome*.	Nenhum
alteração de configuração	RDS-EVENT-0213	O RDS detectou a adição da instância de banco	Nenhum

Categoria	ID do evento do RDS	Message	Observações
		de dados e a adicionou automaticamente ao grupo de destino do *nome* do proxy de banco de dados.	
alteração de configuração	RDS-EVENT-0214	O RDS detectou a exclusão da instância de banco de dados *nome* e a removeu automaticamente do grupo de destino *nome* do proxy de banco de dados *nome*.	Nenhum
alteração de configuração	RDS-EVENT-0215	O RDS detectou a exclusão do cluster de banco de dados *nome* e o removeu automaticamente do grupo de destino *nome* do proxy de banco de dados *nome*.	Nenhum
criação	RDS-EVENT-0203	O RDS criou o proxy de banco de dados *nome*.	Nenhum
criação	RDS-EVENT-0206	O RDS criou o endpoint para *nome* para o proxy do banco de dados *nome*.	Nenhum
exclusão	RDS-EVENT-0205	O RDS excluiu o proxy do banco de dados *nome*.	Nenhum
exclusão	RDS-EVENT-0208	O RDS excluiu o endpoint *nome* para o proxy do banco de dados *nome*.	Nenhum

Categoria	ID do evento do RDS	Message	Observações
falha	RDS-EVENT-0243	O RDS não conseguiu provisionar capacidad e para o *nome* do proxy porque não há endereços IP suficientes disponíveis em suas sub-redes: *nome*. Para resolver o problema, as sub-redes devem ter o número mínimo de endereços IP não usados, conforme recomendado na documentação do proxy do RDS.	Para determinar o número recomendado para sua classe de instância , consulte Planejar a capacidade de endereços IP.
falha	RDS-EVENT-0275	O RDS limitou algumas conexões com o proxy de banco de dados *nome*. O número de solicitações de conexão simultâneas do cliente para o proxy excedeu o limite.	Nenhum

Veja a seguir um exemplo de um evento proxy do RDS no formato JSON. O evento mostra que o RDS modificou o endpoint chamado `my-endpoint` do proxy do RDS chamado `my-rds-proxy`. O ID do evento é RDS-EVENT-0207.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
```

```
    "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

Exemplos de linha de comando do RDS Proxy

Para ver como as combinações de comandos de conexão e instruções SQL interagem com o RDS Proxy, consulte os exemplos a seguir.

Exemplos

- Preserving Connections to a MySQL Database Across a Failover
- Adjusting the max_connections Setting for an Aurora DB Cluster

Example Preservar conexões com um banco de dados MySQL em um failover

Este exemplo do MySQL demonstra como as conexões abertas continuam funcionando durante um failover. Um exemplo é quando você reinicializa um banco de dados ou quando ele se torna indisponível devido a um problema. Este exemplo usa um proxy chamado `the-proxy` e um cluster de bancos de dados Aurora com instâncias de banco de dados `instance-8898` e `instance-9814`. Quando você executa o comando `failover-db-cluster` na linha de comando do Linux, a instância de gravador à qual o proxy está conectado muda para uma instância de banco de dados diferente. Você pode ver que a instância de banco de dados associada ao proxy muda enquanto a conexão permanece aberta.

```
$ mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p
Enter password:
...
```

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ # Initially, instance-9814 is the writer.
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-8898 is the writer.
$ fg
mysql -h the-proxy.proxy-demo.us.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-8898      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-9814 is the writer again.
$ fg
mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)
```

```
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| hostname      | ip-10-1-3-178 |
+---------------+---------------+
1 row in set (0.02 sec)
```

Example Ajustar a configuração de max_connections para um cluster de bancos de dados Aurora.

Este exemplo demonstra como você pode ajustar a configuração de `max_connections` para um cluster de bancos de dados Aurora MySQL. Para fazer isso, crie um grupo de parâmetros de cluster de banco de dados com base nas configurações de parâmetros padrão para clusters compatíveis com o MySQL 5.7. Você especifica um valor para a configuração de `max_connections` substituindo a fórmula que define o valor padrão. Associe o grupo de parâmetros do cluster de banco de dados a seu cluster de banco de dados.

```
export REGION=us-east-1
export CLUSTER_PARAM_GROUP=rds-proxy-mysql-57-max-connections-demo
export CLUSTER_NAME=rds-proxy-mysql-57

aws rds create-db-parameter-group --region $REGION \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-parameter-group-name $CLUSTER_PARAM_GROUP \
  --description "Aurora MySQL 5.7 cluster parameter group for RDS Proxy demo."

aws rds modify-db-cluster --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP

echo "New cluster param group is assigned to cluster:"
aws rds describe-db-clusters --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --query '*[*].{DBClusterParameterGroup:DBClusterParameterGroup}'

echo "Current value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"

echo -n "Enter number for max_connections setting: "
read answer
```

```
aws rds modify-db-cluster-parameter-group --region $REGION --db-cluster-parameter-
group-name $CLUSTER_PARAM_GROUP \
  --parameters "ParameterName=max_connections,ParameterValue=$
$answer,ApplyMethod=immediate"

echo "Updated value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

Solução de problemas do RDS Proxy

A seguir, é possível encontrar ideias de solução de problemas para alguns problemas comuns do proxy do RDS e informações sobre logs do CloudWatch para o RDS Proxy.

Nos logs do RDS Proxy, cada entrada é prefixada com o nome do endpoint de proxy associado. Esse nome pode ser o nome que você especificou para um endpoint definido pelo usuário. Ou pode ser o nome especial `default` para o endpoint padrão de um proxy que realiza solicitações de leitura/gravação. Para obter mais informações sobre os endpoints do proxy, consulte Como trabalhar com endpoints do proxy do Amazon RDS.

Tópicos

- Verificar a conectividade para um proxy
- Problemas e soluções comuns
- Solução de problemas do RDS Proxy com o RDS para MySQL
- Solução de problemas do RDS Proxy com o RDS para PostgreSQL

Verificar a conectividade para um proxy

É possível usar os comandos a seguir para verificar se todos os componentes, como proxy, banco de dados e instâncias de computação na conexão, podem se comunicar entre si.

Examine o próprio proxy usando o comando describe-db-proxies. Examine também o grupo de destino associado usando o comando describe-db-proxy-target-groups. Confira se os detalhes dos destinos correspondem ao cluster do Aurora que você pretende associar ao proxy. Use comandos como os seguintes.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

Para confirmar se o proxy pode se conectar ao banco de dados subjacente, examine os destinos especificados nos grupos de destino usando o comando describe-db-proxy-targets. Use um comando como o seguinte.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

A saída do comando describe-db-proxy-targets inclui um campo `TargetHealth`. É possível examinar os campos `State`, `Reason` e `Description` dentro de `TargetHealth` para verificar se o proxy pode se comunicar com a instância de banco de dados subjacente.

- Um valor `State` de `AVAILABLE` indica que o proxy pode se conectar à instância de banco de dados.
- Um valor `State` de `UNAVAILABLE` indica um problema de conexão temporário ou permanente. Nesse caso, examine os campos `Reason` e `Description`. Por exemplo, se `Reason` tiver um valor de `PENDING_PROXY_CAPACITY`, tente se conectar novamente depois que o proxy terminar sua operação de escalabilidade. Se `Reason` tiver um valor de `UNREACHABLE`, `CONNECTION_FAILED` ou `AUTH_FAILURE`, use a explicação do campo `Description` para ajudá-lo a diagnosticar o problema.
- O campo `State` pode ter um valor de `REGISTERING` por um breve período antes de alterar para `AVAILABLE` ou `UNAVAILABLE`.

Se o comando do Netcat a seguir (`nc`) for bem-sucedido, você poderá acessar o endpoint do proxy na instância do EC2 ou em outro sistema em que esteja conectado. Esse comando relata uma falha se você não estiver na mesma VPC que o proxy e o banco de dados associado. Você pode fazer login diretamente no banco de dados sem estar na mesma VPC. No entanto, não é possível fazer login no proxy a menos que você esteja na mesma VPC.

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

Você pode usar os comandos a seguir para garantir que sua instância do EC2 tenha as propriedades necessárias. Em particular, a VPC da instância do EC2 deve ser a mesma que a VPC da instância de banco de dados do RDS ou do cluster de bancos de dados Aurora ao qual o proxy se conecta.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

Examine os segredos do Secrets Manager usados para o proxy.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

O campo `SecretString` exibido por `get-secret-value` deve estar codificado como uma string JSON que inclua os campos `username` e `password`. O exemplo a seguir mostra o formato do campo `SecretString`.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

Ao solucionar problemas de autenticação do IAM, verifique o seguinte:

- O banco de dados tem a autenticação do IAM habilitada.
- O proxy está configurado com o esquema de autenticação correto.
- As políticas do IAM no perfil do IAM fornecido ao proxy concedem a permissão `rds-db:connect` necessária para o banco de dados apropriado e seu nome de usuário.
- Para a autenticação de ponta a ponta do IAM, existem usuários de banco de dados que correspondem aos nomes de usuário ou perfil do IAM.
- O SSL/TLS está habilitado para a conexão.

Problemas e soluções comuns

Esta seção descreve alguns problemas comuns e possíveis soluções ao usar o Proxy RDS.

Depois de executar o comando `aws rds describe-db-proxy-targets` da CLI, se a descrição de `TargetHealth` indicar `Proxy does not have any registered credentials`, verifique o seguinte:

- Há credenciais registradas para que o usuário acesse o proxy.
- O perfil do IAM para acessar o segredo do Secrets Manager usado pelo proxy é válido.

Você pode encontrar os eventos de RDS a seguir ao criar ou se conectar a um proxy de banco de dados.

Categoria	ID do evento do RDS	Descrição
falha	RDS-EVENT-0243	O RDS não conseguiu provisionar capacidade para o proxy porque não há endereços IP suficient es disponíveis em suas sub-redes. Para resolver o problema, suas sub-redes devem ter o número mínimo de endereços IP não usados. Para determinar o número recomendado para sua classe de instância, consulte Planejar a capacidade de endereços IP.
falha	RDS-EVENT-0275	O RDS limitou algumas conexões com o proxy de banco de dados *nome*. O número de solicitações de conexão simultâneas do cliente para o proxy excedeu o limite.

Você pode encontrar os seguintes problemas ao criar um proxy ou ao se conectar a um proxy.

Erro	Causas ou soluções alternativas
`403: The security token included in the request is invalid`	Selecione uma função existente do IAM em vez de optar por criar uma.

Solução de problemas do RDS Proxy com o RDS para MySQL

Você pode encontrar os problemas a seguir ao se conectar a um proxy do MySQL.

Erro	Causas ou soluções alternativas
`ERROR 1040 (HY000): Connections rate limit exceeded (`*`limit_value`*` )`	A taxa de solicitações de conexão do cliente para o proxy excedeu o limite.
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	O número de solicitações simultâneas com autenticação do IAM do cliente para o proxy excedeu o limite.
`ERROR 1040 (HY000): Number simultaneous connections exceeded (`*`limit_value`*` )`	O número de solicitações de conexão simultâneas do cliente para o proxy excedeu o limite.
`ERROR 1045 (28000): Access`	O segredo do Secrets Manager usado pelo proxy não corresponde ao nome de usuário e à senha de um usuário de banco de dados existente.

Erro	Causas ou soluções alternativas
`denied for user 'DB_USER'@'%' (usi password: YES)`	Atualize as credenciais no segredo do Secrets Manager ou verifique se o usuário do banco de dados existe e tem a mesma senha do segredo.
`ERROR 1105 (HY000): Unknown error`	Ocorreu um erro desconhecido.
`ERROR 1231 (42000): Variable ''charact er_set_cl ient'' can't be set to the value of value`	O valor definido para o parâmetro `character_set_client` não é válido. Por exemplo, o valor `ucs2` não é válido porque ele pode travar o servidor do MySQL.
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	Você habilitou a configuração Exigir Transport Layer Security no proxy, mas a conexão incluiu o parâmetro `ssl-mode=DISABLED` no cliente MySQL. Realize um dos procedimentos a seguir: • Desabilite a configuração Exigir Transport Layer Security para o proxy. • Conectar-se ao banco de dados usando a configuração mínima de `ssl-mode=REQUIRED` no cliente do MySQL.
`ERROR 2026 (HY000): SSL connection error: Internal Server Error`	Falha no handshake TLS para o proxy. Algumas razões possíveis incluem: • O SSL é necessário, mas o servidor não oferece suporte a ele. • Ocorreu um erro interno do servidor. • Ocorreu um handshake ruim.

Erro	Causas ou soluções alternativas
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	O tempo limite do proxy foi atingido enquanto aguardava para adquirir uma conexão de banco de dados. Algumas razões possíveis incluem: • O proxy não consegue estabelecer uma conexão de banco de dados porque o máximo de conexões máximas foi atingido • O proxy não consegue estabelecer uma conexão de banco de dados porque o banco de dados não está disponível.

Solução de problemas do RDS Proxy com o RDS para PostgreSQL

Você pode encontrar os problemas a seguir ao se conectar a um proxy do PostgreSQL.

Erro	Causa	Solução
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	O usuário tentou se conectar ao banco de dados usando a autenticação do IAM com a configuração `sslmode=disable` no cliente do PostgreSQL.	O usuário precisa se conectar ao banco de dados usando a configuração mínima de `sslmode=require` no cliente do PostgreSQL. Para obter mais informações, consulte a documentação PostgreSQL SSL Support.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	Não há segredo do Secrets Manager para essa função.	Adicione um segredo do Secrets Manager para essa função. Para obter mais informações, consulte Configurar a autenticação do IAM para o RDS Proxy.
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	O cliente de banco de dados que está sendo usado para se conectar ao proxy está usando um mecanismo de autentica	Se você não estiver usando a autenticação do IAM, use a autenticação de senha MD5 ou SCRAM.

Erro	Causa	Solução
	ção que não é compatível proxy no momento.	
`ERROR 28000: A user name is missing from the connectio n startup packet. Provide a user name for this connection.`	O cliente de banco de dados que está sendo usado para se conectar ao proxy não está enviando um nome de usuário ao tentar estabelecer uma conexão.	Defina um nome de usuário ao configurar uma conexão com o proxy usando o cliente PostgreSQL de sua escolha.
`ERROR 28000: IAM is allowed only with SSL connections.`	Um cliente tentou se conectar usando a autenticação do IAM, mas o SSL não estava habilitado.	Habilite o SSL no cliente PostgreSQL.
`ERROR 28000: This RDS Proxy requires TLS connections.`	O usuário habilitou a opção Exigir Transport Layer Security, mas tentou se conectar com `sslmode=d isable` no cliente do PostgreSQL.	Para corrigir esse erro, execute um dos seguintes procedimentos: • Desabilite a opção Exigir Transport Layer Security do proxy. • Conectar-se ao banco de dados usando a configura ção mínima de `sslmode=a llow` no cliente do PostgreSQL.

Erro	Causa	Solução
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	Esse erro pode ocorrer devido aos seguintes motivos: • O cliente forneceu o nome de usuário do IAM incorreto. • O cliente forneceu um token de autorização do IAM incorreto para o usuário. • O cliente está usando uma política do IAM que não tem as permissões necessárias. • O cliente forneceu um token de autorização do IAM expirado para o usuário.	Para corrigir esse erro, faça o seguinte: 1. Confirme se o usuário do IAM fornecido existe. 2. Confirme se o token de autorização do IAM pertence ao usuário do IAM fornecido. 3. Confirme se a política do IAM tem as permissões adequadas para o RDS. 4. Verifique a validade do token de autorização do IAM usado.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	A senha para essa função não corresponde ao segredo do Secrets Manager.	Verifique o segredo dessa função no Secrets Manager para ver se a senha é a mesma que está sendo usada no cliente PostgreSQL.
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	Há um problema com o token do IAM usado para a autentica ção do IAM.	Gere outro token de autentica ção e use-o em uma nova conexão.

Erro	Causa	Solução
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	O cliente PostgreSQL usado para se conectar ao proxy usa um protocolo mais antigo que 3.0.	Use um cliente PostgreSQL mais recente que ofereça suporte ao protocolo de sistema de mensagens 3.0. Se você estiver usando a CLI `psql` do PostgreSQL, use uma versão posterior ou igual a 7.4.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	O cliente PostgreSQL usado para se conectar ao proxy está tentando usar o modo de replicação de streaming, que não é compatível com o proxy do RDS no momento.	Desative o modo de replicação de streaming no cliente PostgreSQL que está sendo usado para estabelecer a conexão.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	Pela mensagem de inicialização, o cliente PostgreSQL usado para se conectar ao proxy está solicitando uma opção que não é compatível com o proxy do RDS no momento.	Desative a opção que está sendo mostrada como não compatível na mensagem acima no cliente PostgreSQL que está sendo usado para se conectar.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	O número de solicitações simultâneas com autenticação do IAM do cliente para o proxy excedeu o limite.	Reduza a taxa na qual as conexões que usam a autenticação do IAM de um cliente PostgreSQL são estabelecidas.
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	O número de solicitações de conexão simultâneas do cliente para o proxy excedeu o limite.	Reduza o número de conexões ativas de clientes PostgreSQL para esse proxy do RDS .

Erro	Causa	Solução
`ERROR 53300: Rate of connection to proxy exceeded` *`number_value`* `.`	A taxa de solicitações de conexão do cliente para o proxy excedeu o limite.	Reduza a taxa na qual as conexões de um cliente PostgreSQL são estabelecidas.
`ERROR XX000: Unknown error.`	Ocorreu um erro desconhecido.	Entre em contato com o AWS Support para investigar o problema.
`ERROR 08000: Timed-out waiting to acquire database connection.`	O proxy atingiu o tempo limite enquanto aguardava para obter uma conexão de banco de dados dentro do período especificado pela configuração `ConnectionBorrowTimeout`. Algumas razões possíveis incluem: • O proxy não consegue estabelecer uma conexão de banco de dados porque o máximo de conexões foi atingido. • O proxy não consegue estabelecer uma conexão de banco de dados porque o banco de dados não está disponível ou quando o estabelecimento da conexão com o banco de dados demora mais do que o `ConnectionBorrowTimeout` configurado.	As possíveis soluções são as seguintes: • Evite fixar conexões de proxy. Consulte Evitar a fixação de um RDS Proxy. • Examine as configurações `ConnectionBorrowTimeout` e `MaxConnectionsPercent`. Consulte Considerações sobre a conexão do RDS Proxy. • Examine a disponibilidade do destino. Consulte `AvailabilityPercentage` em Monitorar métricas do proxy do RDS com o Amazon CloudWatch.

Erro	Causa	Solução
`ERROR XX000: Request returned an error: database_error .`	A conexão de banco de dados estabelecida pelo proxy retornou um erro.	A solução depende do erro específico do banco de dados. Um exemplo é: `Request returned an error: database "your-database-name" does not exist`. Isso significa que o nome do banco de dados especificado não existe no servidor do banco de dados. Ou isso significa que o nome de usuário usado como um nome de banco de dados (se não for especificado um nome de banco de dados) não existe no servidor.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	O número de solicitações simultâneas com autenticação do IAM do cliente para o proxy excedeu o limite.	Reduza a taxa na qual as conexões que usam a autenticação do IAM de um cliente PostgreSQL são estabelecidas.

Erro	Causa	Solução
`ERROR 28000: Enable IAM authentication for the client connection to the proxy and try again.`	O RDS Proxy não pode se conectar ao banco de dados porque a autenticação do IAM não está habilitada para a conexão do cliente com o proxy. Isso ocorre quando o parâmetro `DefaultAuthScheme` do proxy está definido como `IAM_AUTH` com um usuário registrado, mas o cliente está usando a autenticação por senha em vez da autenticação do IAM.	Habilite a autenticação do IAM para a conexão do cliente com o proxy e tente novamente.
`ERROR 28000: Configure IAM authentication as the DefaultAuthScheme in your proxy and try again.`	O RDS Proxy não consegue se conectar ao banco de dados porque `DefaultAuthScheme` não está definido como `IAM_AUTH`. O parâmetro `DefaultAuthScheme` do proxy está definido como `NONE`, mas o cliente está tentando usar a autenticação do IAM.	Defina `DefaultAuthScheme` como `IAM_AUTH` para seu proxy e tente novamente.

Solução de problemas de exclusão do banco de dados **`postgres`**

Se você excluir acidentalmente o banco de dados `postgres` da sua instância, precisará restaurá-lo para restabelecer a conectividade com sua instância. Execute os seguintes comandos em sua instância de banco de dados:

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Usar o proxy do RDS com o AWS CloudFormation

É possível usar o proxy do RDS com o AWS CloudFormation. Isso ajuda a criar grupos de recursos relacionados. Esse grupo pode incluir um proxy que pode se conectar a uma instância de banco de dados do Amazon RDS ou a um cluster de bancos de dados do Aurora recém-criado. A compatibilidade do proxy do RDS no CloudFormation envolve dois novos tipos de registro: `DBProxy` e `DBProxyTargetGroup`.

A listagem a seguir mostra um modelo de exemplo do CloudFormation para o RDS Proxy.

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
   Properties:
     DBProxyName: CanaryProxy
     EngineFamily: MYSQL
     RoleArn:
      Fn::ImportValue: SecretReaderRoleArn
     Auth:
       - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
     VpcSubnetIds:
       Fn::Split: [",", "Fn::ImportValue": SubnetIds]

 ProxyTargetGroup:
   Type: AWS::RDS::DBProxyTargetGroup
   Properties:
     DBProxyName: CanaryProxy
     TargetGroupName: default
     DBInstanceIdentifiers:
       - Fn::ImportValue: DBInstanceName
   DependsOn: DBProxy
```

Para obter mais informações sobre os recursos neste exemplo, consulte `DBProxy` e `DBProxyTargetGroup`.

Para obter mais informações sobre os recursos do e do que você pode criar usando o , consulte Referência de tipo de recurso do RDS.

Usa o RDS Proxy com bancos de dados globais do Aurora

Um banco de dados global do Aurora é um banco de dados único que abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e recuperação de desastres de qualquer interrupção de uma região inteira. Ele fornece tolerância a falhas incorporada para sua implantação porque a instância de banco de dados não depende de uma única Região da AWS, mas de várias regiões e zonas de disponibilidade diferentes. Para obter mais informações, consulte Usar o Amazon Aurora Global Database.

É possível usar o RDS Proxy com qualquer cluster de banco de dados em um banco de dados global do Aurora. Antes de começar a usar esses recursos juntos, leia as informações a seguir.

> **Important**
>
> Se o cluster de banco de dados fizer parte de um banco de dados global com o encaminhamento de gravação ativado, reduza o valor `MaxConnectionsPercent` de seu proxy pela cota alocada para o encaminhamento de gravação. A cota de encaminhamento de gravação é definida no parâmetro do cluster de banco de dados `aurora_fwd_writer_max_connections_pct`. Para obter informações sobre o encaminhamento de gravação, consulte Como usar o encaminhamento de gravação em um banco de dados global Amazon Aurora.

Limitações do RDS Proxy com bancos de dados globais

Quando o cluster de banco de dados do Aurora tem o encaminhamento de gravação ativado, o RDS Proxy não é compatível com o valor `SESSION` da variável `aurora_replica_read_consistency`. Definir esse valor pode causar um comportamento inesperado.

Como os endpoints do RDS Proxy funcionam com bancos de dados globais

Quando você entende como os endpoints do RDS Proxy funcionam com bancos de dados globais, você pode gerenciar melhor suas aplicações que usam bancos de dados do Aurora com esses dois recursos.

Para um proxy com o cluster primário de um banco de dados global como destino registrado, os endpoints do proxy funcionam da mesma forma que em qualquer cluster de banco de dados do Aurora. Os endpoints de leitura/gravação do proxy enviam todas as solicitações à instância

do gravador do cluster. Os endpoints somente leitura do proxy enviam todas as solicitações às instâncias de leitura. Se um leitor ficar indisponível enquanto a conexão estiver aberta, o RDS Proxy redirecionará as consultas subsequentes na conexão para outra instância do leitor. Para um proxy com um cluster secundário como destino registrado, as solicitações enviadas aos endpoints somente leitura do proxy também são enviadas para as instâncias de leitor. Como o cluster não tem instâncias de gravador, as solicitações enviadas aos endpoints de leitura/gravação falham com o erro "The `target group doesn't have any associated read/write instances`".

As operações de transição e failover de banco de dados global envolvem uma troca de função entre o cluster de banco de dados primário e um dos secundários. Quando o cluster secundário selecionado se torna o novo primário, uma de suas instâncias de leitor é promovida a um gravador. Essa instância de banco de dados agora é a nova instância de gravador para o cluster global. Redirecione as operações de gravação de sua aplicação para o endpoint de leitura/gravação apropriado do proxy associado ao novo cluster primário. Esse endpoint do proxy pode ser o endpoint padrão ou um endpoint de leitura/gravação personalizado.

O RDS Proxy coloca na fila todas as solicitações por meio de endpoints de leitura/gravação e as envia à instância de gravador do novo cluster primário assim que ele estiver disponível. Isso ocorre independentemente de a operação de transição ou failover ter sido concluída. Durante a transição ou o failover, o endpoint padrão do proxy para o antigo cluster primário ainda aceita operações de gravação. No entanto, assim que esse cluster se torna um cluster secundário, todas as operações de gravação falham. Para saber como e quando executar tarefas específicas de transição ou failover global, consulte os seguintes tópicos:

- Transição de banco de dados: Realizar transições para o Amazon Aurora Global Database
- Failover de banco de dados global: Recuperar um banco de dados global Amazon Aurora de uma interrupção não planejada

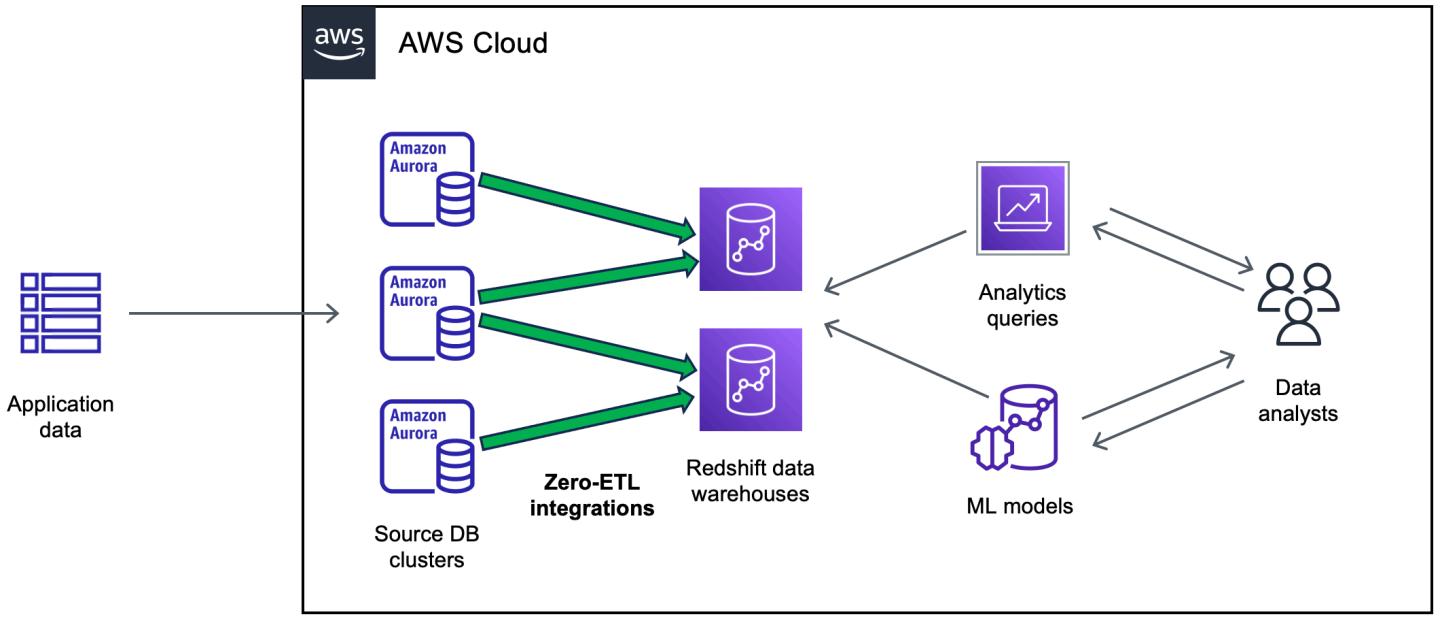
Integrações ETL zero do Aurora

Uma Integração ETL zero do Aurora com o Amazon Redshift e o Amazon SageMaker AI permite analytics e machine learning (ML) quase em tempo real usando dados do Aurora. Essa é uma solução totalmente gerenciada para disponibilizar dados transacionais no destino de analytics depois que eles são gravados em um cluster de banco de dados do Aurora. Extração, transformação e carregamento (ETL) é o processo de combinar dados de várias fontes em um grande data warehouse central.

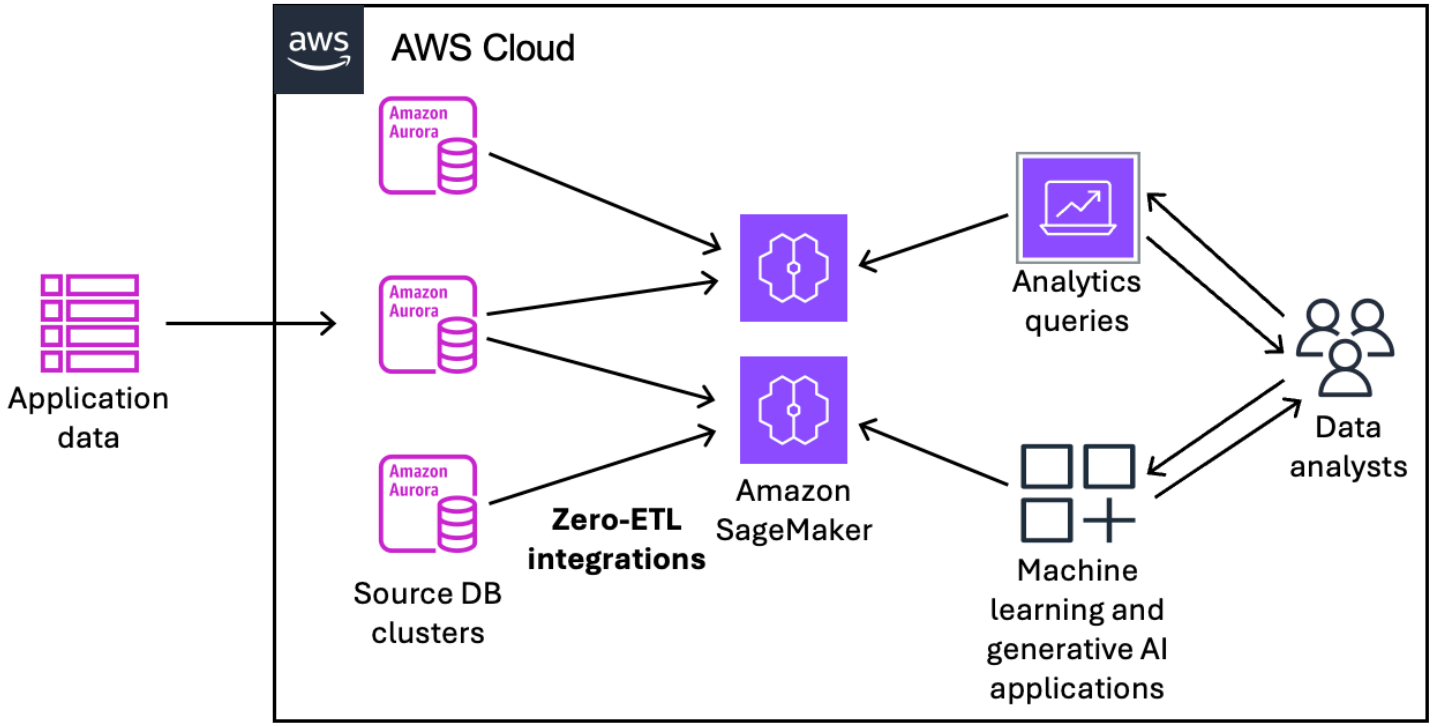
Uma Integração ETL zero torna os dados no cluster de banco de dados do Aurora disponíveis no Amazon Redshift ou em um lakehouse do Amazon SageMaker AI quase em tempo real. Quando esses dados estiverem no data warehouse ou data lake de destino, você poderá potencializar suas workloads de analytics, ML e IA usando os recursos integrados, como machine learning, visões materializadas, compartilhamento de dados, acesso federado a vários datastores e data lakes e integrações ao Amazon SageMaker AI, ao Quick e a outros Serviços da AWS.

Para criar uma Integração ETL zero, especifique um cluster de banco de dados do Aurora como a origem e um data warehouse ou lakehouse compatível como o destino. A integração replica os dados do banco de dados de origem no data warehouse ou lakehouse de destino.

O seguinte diagrama mostra essa funcionalidade de Integração ETL zero com o Amazon Redshift:



O seguinte diagrama mostra essa funcionalidade para uma Integração ETL zero com um lakehouse do Amazon SageMaker AI:



A integração monitora a integridade do pipeline de dados e se recupera de problemas quando possível. É possível criar integrações de vários clusters de banco de dados do Aurora em um único data warehouse ou lakehouse de destino, o que permite que você obtenha insights em várias aplicações.

Para ter informações sobre preços de Integrações ETL zero, consulte Definição de preço do Amazon Aurora e Preço do Amazon Redshift.

Tópicos

- Benefícios
- Principais conceitos
- Limitações
- Cotas
- Regiões com suporte
- Conceitos básicos sobre Integrações ETL zero do Aurora
- Criar integrações ETL zero do Amazon Aurora com o Amazon Redshift
- Criar Integrações ETL zero entre o Aurora e um lakehouse do Amazon SageMaker
- Filtragem de dados para Integrações ETL zero do Aurora
- Adicionar dados a um cluster de banco de dados do Aurora de origem e consultá-los

- Visualizar e monitorar Integrações ETL zero do Aurora
- Modificar Integrações ETL zero do Aurora
- Excluir Integrações ETL zero do Aurora
- Solução de problemas em Integrações ETL zero do Aurora

Benefícios

As Integrações ETL zero do Aurora oferecem os seguintes benefícios:

- Ajudam você a obter insights holísticos de várias fontes de dados.
- Eliminam a necessidade de criar e manter canais de dados complexos que executam operações de extração, transformação e carregamento (ETL). As integrações ETL zero eliminam os desafios que surgem com a criação e o gerenciamento de pipelines, provisionando-os e gerenciando-os para você.
- Reduzem a carga e os custos operacionais para que você possa se concentrar em melhorar as aplicações.
- Permitem que você aproveite os recursos de analytics e ML do destino pretendido para obter insights de dados transacionais e outros dados e responder de forma eficaz a eventos críticos e urgentes.

Principais conceitos

Ao começar a usar integrações ETL zero, considere os seguintes conceitos:

Integração

Um pipeline de dados totalmente gerenciado que replica automaticamente dados e esquemas transacionais de um cluster de banco de dados do Aurora em um data warehouse ou catálogo.

Cluster de banco de dados de origem

O cluster de banco de dados do Aurora do qual os dados são replicados. É possível especificar um cluster de banco de dados que use instâncias de bancos de dados provisionadas ou instâncias de banco de dados do Aurora Serverless v2 como origem.

Target

O data warehouse ou lakehouse para o qual os dados são replicados. Há dois tipos de data warehouse: um data warehouse de cluster provisionado e um data warehouse sem servidor. Um data warehouse de cluster provisionado é um conjunto de recursos computacionais chamados nós, que são organizados em um grupo chamado cluster. Um data warehouse sem servidor é composto por um grupo de trabalho que armazena recursos computacionais e um namespace que abriga os objetos e usuários do banco de dados. Ambos os data warehouses executam um mecanismo de analytics e contêm um ou mais bancos de dados.

Um lakehouse de destino consiste em catálogos, bancos de dados, tabelas e visualizações. Para ter mais informações sobre arquitetura de lakehouse, consulte SageMaker Lakehouse components no Guia do usuário do Amazon SageMaker AI Unified Studio.

Vários clusters de banco de dados de origem podem gravar no mesmo destino.

Para obter mais informações, consulte Arquitetura do sistema de data warehouse no Guia do desenvolvedor do Amazon Redshift.

Limitações

As limitações a seguir se aplicam às Integrações ETL zero do Aurora.

Tópicos

- Limitações gerais
- Limitações do Aurora MySQL
- Limitações do Aurora PostgreSQL
- Limitações do Amazon Redshift
- Amazon SageMaker AILimitações referentes a lakehouses do

Limitações gerais

- O cluster de banco de dados de origem deve estar na mesma região que o destino.
- Não será possível renomear um cluster de banco de dados ou qualquer uma de suas instâncias se o cluster tiver integrações existentes.
- Não é possível criar várias integrações entre os mesmos bancos de dados de origem e destino.

- Você não pode excluir um cluster de banco de dados que tenha integrações existentes. É necessário excluir todas as integrações correspondentes primeiro.
- Se você interromper o cluster de banco de dados de origem, as últimas transações provavelmente não serão replicadas no destino enquanto você não retomar o cluster.
- Se o cluster for a origem de uma implantação azul/verde, os ambientes azul e verde não poderão ter integrações ETL zero existentes durante a transição. Você deve excluir a integração primeiro, alternar e, depois, recriá-la.
- Um cluster de banco de dados deve conter pelo menos uma instância de banco de dados para ser a origem de uma integração.
- Você não pode criar uma integração para um cluster de banco de dados de origem que seja um clone entre contas, como aqueles compartilhados usando AWS Resource Access Manager (AWS RAM).
- Se o cluster de origem for o cluster de banco de dados primário em um banco de dados global do Aurora e fizer o failover em um de seus clusters secundários, a integração se tornará inativa. Você precisa excluir e recriar a integração.
- Não é possível criar uma integração para um banco de dados de origem que tenha outra integração sendo criada ativamente.
- Quando você cria inicialmente uma integração ou quando uma tabela está sendo ressincronizada, a propagação de dados da origem para o destino pode levar de 20 a 25 minutos ou mais, dependendo do tamanho do banco de dados de origem. Esse atraso pode levar a um maior atraso na réplica.
- Alguns tipos de dados não são compatíveis. Para obter mais informações, consulte the section called “Diferenças dos tipos de dados”.
- As tabelas do sistema, tabelas temporárias e visualizações não são replicadas nos warehouses de destino.
- A execução de comandos de DDL (por exemplo, `ALTER TABLE`) em uma tabela de origem pode acionar uma ressincronização da tabela, tornando a tabela indisponível para consulta durante a ressincronização. Para obter mais informações, consulte the section called “Uma ou mais das minhas tabelas do Amazon Redshift exigem ressincronização.”.

Limitações do Aurora MySQL

- Seu cluster de banco de dados de origem deve estar executando uma versão compatível do Aurora MySQL. Para conferir uma lista de versões compatíveis, consulte the section called “Integrações ETL zero ”.
- As integrações ETL zero dependem do registro em log binário (binlog) do MySQL para capturar alterações contínuas de dados. Não use a filtragem de dados baseada em log binário, pois isso pode causar inconsistências de dados entre os bancos de dados de origem e de destino.
- As integrações ETL zero são compatíveis apenas com bancos de dados configurados para usar o mecanismo de armazenamento InnoDB.
- Referências de chave externa com atualizações de tabelas predefinidas não são compatíveis. Especificamente, as regras `ON DELETE` e `ON UPDATE` não são compatíveis com as ações `CASCADE`, `SET NULL` e `SET DEFAULT`. A tentativa de criar ou atualizar uma tabela com essas referências a outra tabela colocará a tabela em um estado de falha.
- As transações XA realizadas no cluster de banco de dados de origem fazem com que a integração entre no estado `Syncing`.

Limitações do Aurora PostgreSQL

- O cluster de banco de dados de origem executar uma versão compatível do Aurora PostgreSQL. Para conferir uma lista de versões compatíveis, consulte the section called “Integrações ETL zero ”.
- Se você selecionar um cluster de banco de dados de origem do Aurora PostgreSQL, deverá especificar pelo menos um padrão de filtro de dados. O padrão deve incluir no mínimo um banco de dados único (*`database-name`*`.*.*`) para replicação no warehouse de destino. Para obter mais informações, consulte the section called “Filtragem de dados para integrações ETL zero”.
- Todos os bancos de dados criados no cluster de banco de dados do Aurora PostgreSQL de origem devem usar a codificação UTF-8.
- Ao usar o particionamento declarativo, as partições da tabela serão replicadas no Amazon Redshift. No entanto, a tabela particionada em si não é replicada no Amazon Redshift.
- Transações em duas fases não são aceitas.
- Se você excluir todas as instâncias de banco de dados de um cluster de banco de dados que seja a origem de uma integração, depois adicionar novamente uma instância de banco de dados, a replicação será interrompida entre os clusters de origem e de destino.

- O cluster de banco de dados de origem não pode usar o Aurora Limitless Database.
- As chaves primárias são necessárias em todas as tabelas presentes no filtro de dados. Qualquer tabela sem uma chave primária será colocada no estado de falha.

Limitações do Amazon Redshift

Para conferir uma lista das limitações do Amazon Redshift relacionadas às integrações ETL zero, consulte Considerações ao usar integrações ETL zero com o Amazon Redshift no Guia de gerenciamento do Amazon Redshift.

Amazon SageMaker AILimitações referentes a lakehouses do

A seguir é apresentada uma limitação para Integrações ETL zero com lakehouses do Amazon SageMaker AI.

- Os nomes de catálogo podem ter no máximo 19 caracteres de comprimento.

Cotas

Sua conta tem as cotas a seguir relacionadas a Integrações ETL zero do Aurora. Salvo indicação em contrário, cada cota aplica-se por região.

Nome	Padrão	Descrição
Integrações	100	O número total de integrações em uma Conta da AWS.
Integrações por destino	50	O número de integrações que enviam dados a um único data warehouse ou lakehouse de destino.
Integrações por cluster de origem	5	O número de integrações que enviam dados de um único cluster de banco de dados de origem.

Além disso, o warehouse de destino impõe certos limites ao número de tabelas permitidas em cada instância de banco de dados ou nó de cluster. Para ter mais informações sobre cotas e limites do Amazon Redshift, consulte Cotas e limites no Amazon Redshift no Guida de gerenciamento do Amazon Redshift.

Regiões com suporte

As Integrações ETL zero do Aurora estão disponíveis em um subconjunto de Regiões da AWS. Para obter uma lista de regiões compatíveis, consulte the section called “Integrações ETL zero ”.

Conceitos básicos sobre Integrações ETL zero do Aurora

Antes de criar uma Integração ETL zero, configure o cluster de banco de dados do Aurora e o data warehouse com as permissões e os parâmetros necessários. Durante a configuração, você realizará as seguintes etapas:

1. Criar um grupo de parâmetros de cluster de banco de dados personalizado.
2. Criar um cluster de banco de dados de origem.
3. Criar um data warehouse de destino para o Amazon Redshift ou Criar um lakehouse de destino do Amazon SageMaker.

Depois de concluir essas tarefas, prossiga para the section called “Criar Integrações ETL zero com o Amazon Redshift” ou the section called “Criar Integrações ETL zero com um lakehouse do Amazon SageMaker”.

É possível usar os SDKs da AWS para automatizar o processo de configuração. Para obter mais informações, consulte the section called “Configurar uma integração usando SDKs da AWS”.

> **Tip**
>
> É possível fazer com que o RDS conclua essas etapas de configuração para você enquanto cria a integração, em vez de realizá-las manualmente. Para começar imediatamente a criar uma integração, consulte the section called “Criar Integrações ETL zero com o Amazon Redshift”.

Para a Etapa 3, você pode optar por criar um data warehouse de destino (Etapa 3a) ou um lakehouse de destino (Etapa 3b), dependendo de suas necessidades:

- Escolha um data warehouse se precisar de recursos tradicionais de armazenamento de dados com analytics baseada em SQL.

- Escolha um lakehouse do Amazon SageMaker se precisar de recursos de machine learning e quiser usar os recursos do lakehouse para fluxos de trabalho de ciência de dados e ML.

Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado

As Integrações ETL zero do Aurora exigem valores específicos para os parâmetros de cluster de banco de dados do Aurora que controlam a replicação. Mais especificamente, o Aurora MySQL requer o log binário aprimorado (`aurora_enhanced_binlog`) e o Aurora PostgreSQL requer replicação lógica aprimorada (`aurora.enhanced_logical_replication`).

Para configurar o registro em log binário ou a replicação lógica, primeiro é necessário criar um grupo de parâmetros de cluster de banco de dados personalizado e, depois, associá-lo ao cluster de banco de dados de origem.

Aurora MySQL (família aurora-mysql):

- `aurora_enhanced_binlog=1`
- `binlog_backup=0`
- `binlog_format=ROW`
- `binlog_replication_globaldb=0`
- `binlog_row_image=full`
- `binlog_row_metadata=full`

Além disso, verifique se o parâmetro `binlog_transaction_compression` não está definido como `ON` e se o parâmetro`binlog_row_value_options` não está definido como `PARTIAL_JSON`.

Para ter mais informações sobre o log binário aprimorado do Aurora MySQL, consulte the section called “Configurar o log binário avançado”.

Aurora PostgreSQL (família aurora-postgresql):

- `rds.logical_replication=1`
- `aurora.enhanced_logical_replication=1`
- `aurora.logical_replication_backup=0`
- `aurora.logical_replication_globaldb=0`

Habilitar a replicação lógica aprimorada (`aurora.enhanced_logical_replication`) sempre gravará todos os valores de coluna no log de gravação antecipada (WAL), mesmo que `REPLICA IDENTITY FULL` não esteja habilitada. Isso pode aumentar as IOPS do cluster de banco de dados de origem.

> **Important**
>
> Se você desabilitar o parâmetro de cluster de banco de dados `aurora.enhanced_logical_replication` a instância de banco de dados primária invalidará todos os slots de replicação lógica. Isso interrompe a replicação da origem no destino, e você deve recriar os slots de replicação na instância de banco de dados primária. Para evitar interrupções, mantenha o estado do parâmetro consistente durante a replicação.

Etapa 2: Selecionar ou criar um cluster de banco de dados de origem

Depois de criar um grupo de parâmetros de cluster de banco de dados personalizado, escolha ou crie um cluster de banco de dados do Aurora. Esse cluster será a origem da replicação de dados para o data warehouse de destino. Você pode especificar um cluster de banco de dados que usa instâncias de banco de dados provisionadas ou instâncias de banco de dados do Aurora Serverless v2. Consulte instruções para criar um cluster de banco de dados, consulte the section called “Criar um cluster de banco de dados” ou the section called “Criar um cluster de banco de dados do Aurora Serverless v2”.

O banco de dados deve estar executando uma versão de mecanismo de banco de dados compatível. Para conferir uma lista de versões compatíveis, consulte the section called “Integrações ETL zero ”.

Ao criar o banco de dados, em Configuração adicional, altere o grupo de parâmetros de cluster de banco de dados padrão para o grupo de parâmetros personalizado que você criou na etapa anterior.

> **Note**
>
> Se você associar o grupo de parâmetros ao cluster de banco de dados depois que o cluster for criado, deverá reinicializar a instância de banco de dados primária no cluster para aplicar as alterações antes de criar uma Integração ETL zero. Para receber instruções, consulte the section called “Reinicializar um cluster ou instância de banco de dados do Aurora”.

Etapa 3a: criar um data warehouse de destino

Depois de criar o cluster de banco de dados, será necessário criar e configurar um data warehouse de destino. O data warehouse deve cumprir os seguintes requisitos:

- Usar um tipo de nó RA3 com pelo menos dois nós ou o Redshift sem servidor.
- Ser criptografado (se estiver usando um cluster provisionado). Para obter mais informações, consulte Criptografia de bancos de dados no Amazon Redshift.

Para obter instruções sobre como criar um data warehouse, consulte Criar um cluster para clusters provisionados ou Criar um grupo de trabalho com um namespace para o Redshift Serverless.

Ative a distinção entre maiúsculas e minúsculas no data warehouse

Para que a integração seja bem-sucedida, o parâmetro de diferenciação de maiúsculas e minúsculas (`enable_case_sensitive_identifier`) deve estar ativado para o data warehouse. Por padrão, a distinção entre maiúsculas e minúsculas é desativada em todos os clusters provisionados e grupos de trabalho do Redshift Serverless.

Para ativar a distinção entre maiúsculas e minúsculas, execute as seguintes etapas, dependendo do tipo de data warehouse:

- Cluster provisionado: para ativar a distinção entre maiúsculas e minúsculas em um cluster provisionado, crie um grupo de parâmetros personalizado com o parâmetro `enable_case_sensitive_identifier` ativado. Em seguida, associe o grupo de parâmetros ao cluster. Para obter instruções, consulte Gerenciar grupos de parâmetros usando o console ou Configurar valores de parâmetros usando a AWS CLI.

  > **Note**
  >
  > Lembre-se de reinicializar o cluster depois de associar o grupo de parâmetros personalizado a ele.

- Grupo de trabalho de tecnologia sem servidor: para ativar a distinção entre maiúsculas e minúsculas em um grupo de trabalho do Redshift Serverless, você deve usar a AWS CLI. Atualmente, o console do Amazon Redshift não é compatível com a modificação dos valores dos parâmetros do Redshift Serverless. Envie a seguinte solicitação de atualização do grupo de trabalho:

```
aws redshift-serverless update-workgroup \
  --workgroup-name target-workgroup \
  --config-parameters
 parameterKey=enable_case_sensitive_identifier,parameterValue=true
```

Não é necessário reinicializar um grupo de trabalho após modificar seus valores de parâmetros.

Configurar a autorização para o data warehouse

Depois de criar um data warehouse, você deve configurar o cluster de banco de dados do Aurora de origem como uma origem de integração autorizada. Para obter instruções, consulte Configurar a autorização para o data warehouse do Amazon Redshift.

Configurar uma integração usando SDKs da AWS

Em vez de configurar cada recurso manualmente, é possível executar o script Python a seguir para configurar automaticamente os recursos necessários. O exemplo de código usa o AWS SDK para Python (Boto3) para criar um cluster de banco de dados do Amazon Aurora de origem e um data warehouse de destino, cada um com os valores de parâmetro necessários. Depois, ele espera que os bancos de dados estejam disponíveis antes de criar uma integração ETL zero entre eles. É possível comentar diferentes funções, dependendo dos recursos que você precisa configurar.

Execute os comandos a seguir para instalar as dependências necessárias:

```
pip install boto3
pip install time
```

No script, modifique opcionalmente os nomes dos grupos de parâmetros, a origem e o destino. A função final cria uma integração denominada `my-integration` após a configuração dos recursos.

Código de exemplo em Python

Aurora MySQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
```

```
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora MySQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-mysql8.0',
        Description='For Aurora MySQL binary logging'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'aurora_enhanced_binlog',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_replication_globaldb',
```

```
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_metadata',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-mysql',
        EngineVersion='8.0.mysql_aurora.3.05.2',
        DatabaseName='myauroradb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-mysql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
```

```
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
```

```
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']
```

```
    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Aurora PostgreSQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
```

```
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora PostgreSQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-postgresql16',
        Description='For Aurora PostgreSQL logical replication'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'rds.logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.enhanced_logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])
```

```
    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-postgresql',
        EngineVersion='16.4.aurora-postgresql',
        DatabaseName='mypostgresdb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-postgresql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora PostgreSQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])
```

```
response = redshift.create_cluster(
    ClusterIdentifier=target_cluster_name,
    NodeType='ra3.4xlarge',
    NumberOfNodes=2,
    Encrypted=True,
    MasterUsername='username',
    MasterUserPassword='Password01**',
    ClusterParameterGroupName=target_param_group_name
)
print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

# Retrieve the target cluster ARN
response = redshift.describe_clusters(
    ClusterIdentifier=target_cluster_name
)
target_arn = response['Clusters'][0]['ClusterNamespaceArn']

# Retrieve the current user's account ID
response = sts.get_caller_identity()
account_id = response['Account']

# Create a resource policy specifying cluster ARN and account ID
response = redshift.put_resource_policy(
    ResourceArn=target_arn,
    Policy='''
    {
        \"Version\":\"2012-10-17\",
        \"Statement\":[
            {\"Effect\":\"Allow\",
            \"Principal\":{
                \"Service\":\"redshift.amazonaws.com\"
            },
            \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
            \"Condition\":{
                \"StringEquals\":{
                    \"aws:SourceArn\":\"%s\"}
                }
            },
            {\"Effect\":\"Allow\",
            \"Principal\":{
                \"AWS\":\"arn:aws:iam::%s:root\"},
            \"Action\":\"redshift:CreateInboundIntegration\"}
        ]
```

```
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
```

```
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Etapa 3b: criar um catálogo do AWS Glue para uma Integração ETL zero do Amazon SageMaker Lakehouse

Ao criar uma Integração ETL zero com um lakehouse do Amazon SageMaker, você deve criar um catálogo gerenciado pelo AWS Glue no AWS Lake Formation. O catálogo de destino deve ser um catálogo gerenciado pelo Amazon Redshift. Para criar um catálogo gerenciado pelo Amazon Redshift, primeiro crie um perfil vinculado ao serviço `AWSServiceRoleForRedshift`. No console do Lake Formation, adicione o `AWSServiceRoleForRedshift` como administrador somente para leitura.

Para ter mais informações sobre as tarefas anteriores, consulte os tópicos a seguir.

- Para ter informações sobre como criar um catálogo gerenciado pelo Amazon Redshift, consulte Creating an Amazon Redshift managed catalog in the AWS Glue Data Catalog no Guia do desenvolvedor do AWS Lake Formation.
- Para ter informações sobre perfis vinculados ao serviço, consulte Uso de funções vinculadas ao serviço para o Amazon Redshift no Guia de gerenciamento do Amazon Redshift.
- Para ter informações sobre as permissões de administrador somente para leitura do Lake Formation, consulte Lake Formation personas and IAM permissions reference no Guia do desenvolvedor do AWS Lake Formation.

Configurar permissões para o catálogo do AWS Glue

Antes de criar um catálogo de destino para uma Integração ETL zero, você deve criar o perfil de criação de destino do Lake Formation e o perfil de transferência de dados do AWS Glue. Use o perfil de criação de destino do Lake Formation para criar o catálogo de destino. Ao criar o catálogo de

destino, insira um perfil de transferência de dados do Glue no campo Perfil do IAM na seção Acesso por meio de mecanismos.

Perfil de criação de destino do Lake Formation

O perfil de criação de destino deve ser administrador do Lake Formation e requer as permissões a seguir.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lakeformation:RegisterResource",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutEncryptionConfiguration",
                "iam:PassRole",
                "glue:CreateCatalog",
                "glue:GetCatalog",
                "s3:PutBucketTagging",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "redshift-serverless:CreateNamespace",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
                "redshift-serverless:CreateWorkgroup"
            ],
            "Resource": [
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:catalog/*",
                "arn:aws:s3:::*",
                "arn:aws:redshift-serverless:*:111122223333:workgroup/*",
                "arn:aws:redshift-serverless:*:111122223333:namespace/*",
                "arn:aws:iam::111122223333:role/GlueDataCatalogDataTransferRole"
```

```
            ]
        }
    ]
}
```

O perfil de criação de destino deve ter a relação de confiança a seguir.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:user/Username"
          },
          "Action": "sts:AssumeRole"
        }
    ]
}
```

Perfil de transferência de dados do Glue

O perfil de transferência de dados do Glue é necessário para operações de catálogo do MySQL e deve ter as permissões a seguir.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "DataTransferRolePolicy",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "glue:GetCatalog",
                "glue:GetDatabase"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

O perfil de transferência de dados do Glue deve ter a relação de confiança a seguir.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "glue.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Próximas etapas

Com um cluster de banco de dados do Aurora de origem e um data warehouse de destino do Amazon Redshift ou um lakehouse do Amazon SageMaker, é possível criar uma Integração ETL

zero e começar a replicar dados. Para instruções, consulte the section called “Criar Integrações ETL zero com o Amazon Redshift”.

Criar integrações ETL zero do Amazon Aurora com o Amazon Redshift

Ao criar uma integração ETL zero do Aurora, especifique o cluster de banco de dados do Aurora de origem e o data warehouse de destino do Amazon Redshift. Você também pode personalizar as configurações de criptografia e adicionar etiquetas. O Aurora cria uma integração entre o cluster de banco de dados de origem e o destino. Quando a integração está ativa, todos os dados inseridos no cluster de banco de dados de origem serão replicados no destino configurado do Amazon Redshift.

Pré-requisitos

Antes de criar uma integração ETL zero, é necessário criar um cluster de banco de dados de origem e um data warehouse de destino do Amazon Redshift. Também é necessário permitir a replicação no data warehouse adicionando o cluster de banco de dados como uma origem de integração autorizada.

Para obter instruções sobre como concluir cada uma dessas etapas, consulte the section called “Conceitos básicos das integrações ETL zero”.

Permissões obrigatórias

Algumas permissões do IAM são necessárias para criar uma integração ETL zero. No mínimo, são necessárias permissões para executar as seguintes ações:

- Crie integrações ETL zero para o cluster de banco de dados do Aurora de origem.
- Visualizar e excluir todas as integrações ETL zero.
- Criar integrações de entrada no data warehouse de destino.

As políticas de exemplo a seguir demonstram as permissões de privilégio mínimo necessárias para criar e gerenciar integrações. Talvez você não precise dessas permissões exatas se o usuário ou o perfil tiver permissões mais amplas, como uma política gerenciada `AdministratorAccess`.

> **Note**
>
> Os ARNs do Amazon Redshift têm o formato a seguir. Observe o uso de uma barra (`/`) em vez de dois pontos (`:`) antes do UUID do namespace sem servidor.
>
> - Cluster provisionado: `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - Sem servidor: `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Exemplo de política para o destino do Redshift

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateIntegration",
      "Effect": "Allow",
      "Action": [
        "rds:CreateIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:db:source-db",
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "DescribeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeIntegrations"
      ],
      "Resource": [
      "arn:aws:rds:us-east-1:123456789012:integration:*"
  ]
    },
    {
```

```
      "Sid": "ChangeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DeleteIntegration",
        "rds:ModifyIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "AllowRedShiftIntegration",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:123456789012:namespace:namespace-uuid"
      ]
    }
  ]
}
```

Escolher um data warehouse de destino em uma conta diferente

Se você planeja especificar um data warehouse de destino do Amazon Redshift que esteja em outraConta da AWS, deverá criar um perfil que permita que os usuários da conta atual acessem os recursos na conta de destino. Para obter mais informações, consulte Fornecer acesso a um usuário do IAM em outra Conta da AWS de sua propriedade.

O perfil deve ter as permissões a seguir, que possibilitam ao usuário visualizar os clusters provisionados do Amazon Redshift e os namespaces do Redshift sem servidor disponíveis na conta de destino.

Permissões necessárias e política de confiança

JSON

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

O perfil deve ter a seguinte política de confiança, que especifica o ID da conta de destino.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Para obter instruções sobre como criar o perfil, consulte Criar um perfil usando políticas de confiança personalizadas.

Criar integrações ETL zero

É possível criar uma integração ETL zero usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Important**
>
> As Integrações ETL zero não permitem operações de atualização ou ressincronização. Se você encontrar problemas com uma integração após a criação, deverá excluir a integração e criar outra.

Console do RDS

Como criar uma integração ETL zero

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação à esquerda, escolha Integrações ETL zero.
3. Escolha Criar integração ETL zero.
4. Em Nome da integração, insira um nome para a integração. O nome pode ter até 60 caracteres alfanuméricos e pode incluir hifens.

   > **Important**
   >
   > Os nomes de catálogo podem ter no máximo 19 caracteres de comprimento. O identificador da integração deve atender a esse requisito se for usado como nome de catálogo.

5. Escolha Próximo.
6. Em Origem, selecione o cluster de banco de dados do Aurora de onde os dados serão originados.

   > **Note**
   >
   > O RDS notifica você se os parâmetros do cluster de banco de dados não estiverem configurados corretamente. Se você receber essa mensagem, poderá escolher Corrigir para mim ou configurá-las manualmente. Para obter instruções sobre como corrigi-los manualmente, consultethe section called “Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado”.

A modificação dos parâmetros do cluster de banco de dados requer uma reinicialização. Antes de criar a integração, a reinicialização deve ser concluída e os valores do novo parâmetro deve ser aplicado com êxito ao cluster.

7. (Opcional) Selecione Personalizar opções de filtragem de dados e adicione filtros de dados à integração. É possível usar filtros de dados para definir o escopo da replicação para o data warehouse de destino. Para obter mais informações, consulte the section called “Filtragem de dados para integrações ETL zero”.
8. Depois que o cluster de banco de dados de origem for configurado com êxito, selecione Próximo.
9. Em Destino 1, faça o seguinte:
   1. (Opcional) Para usar uma Conta da AWS diferente para o destino do Amazon Redshift, escolha Especificar uma conta diferente. Depois, insira o ID da e o nome de um perfil do IAM com permissões para exibir seus data warehouses. Para obter instruções sobre como criar um perfil do IAM, consulte the section called “Escolher um data warehouse de destino em uma conta diferente”.
   2. Em Data warehouse do Amazon Redshift, selecione o destino para os dados replicados do cluster de banco de dados de origem. Você pode escolher um cluster provisionado do Amazon Redshift ou um namespace do Redshift sem servidor como destino.

> **Note**
>
> O RDS notifica você se a política de recursos ou as configurações de diferenciação de maiúsculas e minúsculas do data warehouse especificado não estiverem configuradas corretamente. Se você receber essa mensagem, poderá escolher Corrigir para mim ou configurá-las manualmente. Para obter instruções sobre como corrigi-los manualmente, consulte Ativar a diferenciação de maiúsculas e minúsculas para seu armazém de dados e Configurar a autorização para seu armazém de dados no Guia de gerenciamento do Amazon Redshift.
>
> A modificação da distinção entre maiúsculas e minúsculas de um cluster provisionado do Redshift requer uma reinicialização. Antes de criar a integração, a reinicialização deve ser concluída e o novo valor do parâmetro deve ser aplicado com êxito ao cluster.

Se a origem e o destino selecionados estiverem em valores diferentesContas da AWS, o Amazon RDS não poderá corrigir essas configurações para você. Você deve navegar até a outra conta e corrigi-la manualmente no Amazon Redshift.

10. Depois que seu data warehouse de destino estiver configurado corretamente, escolha Avançar.
11. (Opcional) Em Etiquetas, adicione uma ou mais etiquetas à integração. Para obter mais informações, consulte the section called “Marcar recursos do Aurora e do RDS”.
12. Em Criptografia, especifique como você deseja que sua integração seja criptografada. Por padrão, o RDS criptografa todas as integrações com uma Chave pertencente à AWS. Para escolher uma chave gerenciada pelo cliente em vez disso, habilite a opção Personalizar configurações de criptografia e escolha uma chave do KMS para criptografia. Para obter mais informações, consulte the section called “Criptografar recursos do Amazon Aurora”.

    Como opção, adicione um contexto de criptografia. Para obter mais informações, consulte Contexto de criptografia no Guia do desenvolvedor AWS Key Management Service.

    > **Note**
    >
    > O Amazon RDS adiciona os seguintes pares de contexto de criptografia, além dos adicionados por você:
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > Isso reduz o número geral de pares que você pode adicionar de oito para seis e contribui com o limite geral de caracteres da restrição de concessão. Para ter mais informações, consulte Using grant constraints no Guia do desenvolvedor do AWS Key Management Service.

13. Escolha Próximo.
14. Revise suas configurações de integração e escolha Criar integração sem ETL.

    Se a criação falhar, consulte the section called “Não consigo criar uma integração ETL zero.” para conferir etapas de solução de problemas.

A integração tem um status de `Creating` enquanto está sendo criada, e o data warehouse de destino do Amazon Redshift tem um status de `Modifying`. Durante esse período, você não pode consultar o data warehouse nem alterar sua configuração.

Quando a integração tiver sido criada com sucesso, o status da integração e do data warehouse de destino do Amazon Redshift vão mudar para `Active`.

AWS CLI

Para criar uma integração sem ETL usando oAWS CLI, use o comando create-integration com as seguintes opções:

> **Note**
>
> Lembre-se de que os nomes de catálogo podem ter no máximo 19 caracteres. Escolha o nome da integração adequadamente se você pretende usá-lo como nome de catálogo.

- `--integration-name`— Especifique um nome para a integração.
- `--source-arn`: especifique o ARN do cluster de banco de dados do Aurora que será a origem da integração.
- `--target-arn`: especifique o ARN do data warehouse do Amazon Redshift que será o destino da integração.

Example

Para Linux, macOS ou Unix:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Para Windows:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

API do RDS

Para criar uma implantação azul/verde usando a API do Amazon RDS, use a operação `CreateIntegration` com os seguintes parâmetros:

> **Note**
>
> Os nomes de catálogo podem ter no máximo 19 caracteres. O parâmetro IntegrationName deve atender a esse requisito se for usado como nome de catálogo.

- `IntegrationName`— Especifique um nome para a integração.
- `SourceArn`: especifique o ARN do cluster de banco de dados do Aurora que será a origem da integração.
- `TargetArn`: especifique o ARN do data warehouse do Amazon Redshift que será o destino da integração.

Criptografar integrações com uma chave gerenciada pelo cliente

Se você especificar uma chave do KMS personalizada em vez de uma Chave pertencente à AWS ao criar uma integração, a política de chaves deverá fornecer ao serviço Amazon Redshift um acesso de entidade principal à ação `CreateGrant`. Além disso, ela deve permitir que o usuário atual execute as ações `DescribeKey` e `CreateGrant`.

O exemplo de política a seguir demonstra como conceder as permissões necessárias na política de chave. Ele inclui chaves de contexto para reduzir ainda mais o escopo das permissões.

Exemplo de política de chave

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
```

```
                    "CreateGrant"
                ]
            }
        }
    },
    {
        "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
        },
        "Action": "kms:DescribeKey",
        "Resource": "*"
    }
  ]
}
```

Para obter mais informações, consulte Criar uma política de chave no Guia do desenvolvedor do AWS Key Management Service.

Próximas etapas

Depois de criar com sucesso uma integração ETL zero, você deve criar um banco de dados de destino em seu cluster ou grupo de trabalho de destino do Amazon Redshift. Depois, você poderá começar a adicionar dados ao cluster de banco de dados do Aurora e consultá-los no Amazon Redshift. Para obter instruções, consulte Criar bancos de dados de destino no Amazon Redshift.

Criar Integrações ETL zero entre o Aurora e um lakehouse do Amazon SageMaker

Ao criar uma Integração ETL zero entre o Aurora e um lakehouse do Amazon SageMaker, você deve especificar o cluster do banco de dados do Aurora de origem e o catálogo de destino gerenciado pelo AWS Glue. Você também pode personalizar as configurações de criptografia e adicionar etiquetas. O Aurora cria uma integração entre o cluster de banco de dados de origem e o destino. Quando a integração estiver ativa, todos os dados inseridos no cluster de banco de dados de origem serão replicados no destino configurado.

Pré-requisitos

Antes de criar uma Integração ETL zero com um lakehouse do Amazon SageMaker, você deve criar um cluster de banco de dados de origem e um catálogo de destino gerenciado pelo AWS Glue. Também é necessário permitir a replicação no catálogo adicionando o cluster de banco de dados como uma origem de integração autorizada.

Para obter instruções sobre como concluir cada uma dessas etapas, consulte the section called "Conceitos básicos das integrações ETL zero".

Permissões obrigatórias

Algumas permissões do IAM são necessárias para criar uma Integração ETL zero com o lakehouse do Amazon SageMaker. No mínimo, são necessárias permissões para executar as seguintes ações:

- Crie integrações ETL zero para o cluster de banco de dados do Aurora de origem.
- Visualizar e excluir todas as integrações ETL zero.
- Criar integrações de entrada no catálogo de destino gerenciado pelo AWS Glue.
- Acessar os buckets do Amazon S3 usados pelo catálogo gerenciado pelo AWS Glue.
- Usar chaves do AWS KMS para criptografia se a criptografia personalizada estiver configurada.
- Registrar recursos com o Lake Formation.
- Inserir a política de recursos no catálogo gerenciado pelo AWS Glue para autorizar integrações de entrada.

O exemplo de política a seguir demonstra as permissões de privilégio mínimo necessárias para criar e gerenciar integrações com um lakehouse do Amazon SageMaker. Talvez você não precise dessas permissões exatas se o usuário ou o perfil tiver permissões mais amplas, como uma política gerenciada `AdministratorAccess`.

Além disso, você deve configurar uma política de recursos no catálogo de destino gerenciado pelo AWS Glue para autorizar integrações de entrada. Use o comando da AWS CLI a seguir para aplicar a política de recursos.

Exemplo de comando da AWS CLI para autorizar integrações de entrada no catálogo de destino

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
```

```
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Os nomes de recurso da Amazon (ARNs) do catálogo do Glue têm o seguinte formato:
>
> - Catálogo do Glue: `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`.

Escolher um catálogo de destino gerenciado pelo AWS Glue em uma conta diferente

Se você planeja especificar um catálogo de destino gerenciado pelo AWS Glue que esteja em outra Conta da AWS, deverá criar um perfil que permita que os usuários da conta atual acessem os recursos na conta de destino. Para obter mais informações, consulte Fornecer acesso a um usuário do IAM em outra Conta da AWS de sua propriedade.

O perfil deve ter as permissões a seguir, que possibilitam ao usuário visualizar os catálogos disponíveis do AWS Glue na conta de destino.

Permissões necessárias e política de confiança

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:GetCatalog"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

O perfil deve ter a seguinte política de confiança, que especifica o ID da conta de destino.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::111122223333:root"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Para obter instruções sobre como criar o perfil, consulte Criar um perfil usando políticas de confiança personalizadas.

Criar Integrações ETL zero com um lakehouse do Amazon SageMaker

É possível criar uma Integração ETL zero com um lakehouse do Amazon SageMakerusando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

> **Important**
>
> Em Integrações ETL zero com um lakehouse do Amazon SageMaker, não é possível usar operações de atualização ou ressincronização. Se você encontrar problemas com uma integração após a criação, deverá excluir a integração e criar outra.

Console do RDS

Como criar Integrações ETL zero com um lakehouse do Amazon SageMaker

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação à esquerda, escolha Integrações ETL zero.
3. Escolha Criar integração ETL zero.
4. Em Nome da integração, insira um nome para a integração. O nome pode ter até 60 caracteres alfanuméricos e pode incluir hifens.
5. Escolha Próximo.
6. Em Origem, selecione o cluster de banco de dados do Aurora de onde os dados serão originados.

> **Note**
>
> O RDS notifica você se os parâmetros do cluster de banco de dados não estiverem configurados corretamente. Se você receber essa mensagem, poderá escolher Corrigir para mim ou configurá-las manualmente. Para obter instruções sobre como corrigi-los manualmente, consultethe section called “Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado”.

A modificação dos parâmetros do cluster de banco de dados requer uma reinicialização. Antes de criar a integração, a reinicialização deve ser concluída e os valores do novo parâmetro deve ser aplicado com êxito ao cluster.

7. (Opcional) Selecione Personalizar opções de filtragem de dados e adicione filtros de dados à integração. É possível usar filtros de dados para definir o escopo da replicação para o lakehouse de destino do Amazon SageMaker. Para obter mais informações, consulte the section called “Filtragem de dados para integrações ETL zero”.
8. Depois que o cluster de banco de dados de origem for configurado com êxito, selecione Próximo.
9. Em Destino 1, faça o seguinte:

   1. (Opcional) Para usar uma Conta da AWS diferente para o lakehouse de destino do Amazon SageMaker, escolha Especificar uma conta diferente. Depois, insira o ARN de um perfil do IAM com permissões para exibir seus catálogos do AWS Glue. Para obter instruções sobre como criar um perfil do IAM, consulte the section called “Escolher um catálogo de destino gerenciado pelo AWS Glue em uma conta diferente”.
   2. Em Catálogo do AWS Glue, selecione o destino para os dados replicados do cluster de banco de dados de origem. Você pode escolher um catálogo existente gerenciado pelo AWS Glue como destino.
   3. O perfil do IAM de destino precisa de permissões de descrição no catálogo de destino e deve ter as seguintes permissões:

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "glue:GetCatalog",
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ]
        }
    ]
}
```

O perfil do IAM deve ter a seguinte relação de confiança:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. Você deve conceder ao perfil do IAM de destino as permissões de descrição referentes ao catálogo de destino gerenciado pelo AWS Glue com o perfil de administrador do Lake Formation criado na seção Etapa 3b: criar um catálogo do AWS Glue para uma Integração ETL zero do Amazon SageMaker Lakehouse.

> **Note**
>
> O RDS notificará você se a política de recursos ou as configurações do catálogo gerenciado pelo AWS Glue especificado não estiverem configuradas corretamente. Se você receber essa mensagem, poderá escolher Corrigir para mim ou configurá-las manualmente.
>
> Se a origem e o destino selecionados estiverem em valores diferentesContas da AWS, o Amazon RDS não poderá corrigir essas configurações para você. Você deve navegar até a outra conta e corrigi-la manualmente no SageMaker Unified Studio.

10. Depois que o catálogo de destino gerenciado pelo AWS Glue estiver configurado corretamente, escolha Próximo.
11. (Opcional) Em Etiquetas, adicione uma ou mais etiquetas à integração. Para obter mais informações, consulte the section called “Marcar recursos do Aurora e do RDS”.
12. Em Criptografia, especifique como você deseja que sua integração seja criptografada. Por padrão, o RDS criptografa todas as integrações com uma Chave pertencente à AWS. Para

escolher uma chave gerenciada pelo cliente em vez disso, habilite a opção Personalizar configurações de criptografia e escolha uma chave do KMS para criptografia. Para obter mais informações, consulte the section called “Criptografar recursos do Amazon Aurora”.

Como opção, adicione um contexto de criptografia. Para obter mais informações, consulte Contexto de criptografia no Guia do desenvolvedor AWS Key Management Service.

> **Note**
>
> O Amazon RDS adiciona os seguintes pares de contexto de criptografia, além dos adicionados por você:
>
> - `aws:glue:integration:arn` - `IntegrationArn`
> - `aws:servicename:id` - `glue`
>
> Isso reduz o número geral de pares que você pode adicionar de oito para seis e contribui com o limite geral de caracteres da restrição de concessão. Para ter mais informações, consulte Using grant constraints no Guia do desenvolvedor do AWS Key Management Service.

13. Escolha Próximo.
14. Revise suas configurações de integração e escolha Criar integração sem ETL.

    Se a criação falhar, consulte the section called “Solução de problemas em integrações ETL zero” para conferir etapas de solução de problemas.

A integração tem o status `Creating` enquanto está sendo criada, e o lakehouse de destino do Amazon SageMaker tem o status `Modifying`. Durante esse período, não é possível consultar o catálogo nem alterar sua configuração.

Quando a integração é criada com êxito, o status da integração e o lakehouse de destino do Amazon SageMaker mudam para `Active`.

AWS CLI

Para preparar um catálogo de destino gerenciado pelo AWS Glue para Integração ETL zero usando a AWS CLI, você deve primeiro usar o comando create-integration-resource-property com as seguintes opções:

- `--resource-arn`: especifique o ARN do catálogo gerenciado pelo AWS Glue que será o destino da integração.
- `--target-processing-properties`: especifique o ARN do perfil do IAM para acessar o catálogo de destino gerenciado pelo AWS Glue.

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

Para criar uma Integração ETL zero com um lakehouse do Amazon SageMaker usando a AWS CLI, use o comando create-integration com as seguintes opções:

- `--integration-name`— Especifique um nome para a integração.
- `--source-arn`: especifique o ARN do cluster de banco de dados do Aurora que será a origem da integração.
- `--target-arn`: especifique o ARN do catálogo gerenciado pelo AWS Glue que será o destino da integração.

Example

Para Linux, macOS ou Unix:

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Para Windows:

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

API do RDS

Para criar uma Integração ETL zero como Amazon SageMaker usando a API do Amazon RDS, use a operação `CreateIntegration` com os seguintes parâmetros:

> **Note**
>
> Os nomes de catálogo podem ter no máximo 19 caracteres. O parâmetro IntegrationName deve atender a esse requisito se for usado como nome de catálogo.

- `IntegrationName`— Especifique um nome para a integração.
- `SourceArn`: especifique o ARN do cluster de banco de dados do Aurora que será a origem da integração.
- `TargetArn`: especifique o ARN do catálogo gerenciado pelo AWS Glue que será o destino da integração.

Criptografar integrações com uma chave gerenciada pelo cliente

Se você especificar uma chave do KMS personalizada em vez de uma Chave pertencente à AWS ao criar uma integração com o Amazon SageMaker, a política de chave deverá fornecer à entidade principal do serviço SageMaker Unified Studio acesso à ação `CreateGrant`. Além disso, ela deve permitir que o usuário atual execute as ações `DescribeKey` e `CreateGrant`.

O exemplo de política a seguir demonstra como conceder as permissões necessárias na política de chave. Ele inclui chaves de contexto para reduzir ainda mais o escopo das permissões.

Exemplo de política de chave

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "EnablesIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
```

```
            "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "GlueServicePrincipalAddGrant",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleAddGrantKMSKey",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
```

```
                    ]
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleRetrieveKMSKeyInformation",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Para obter mais informações, consulte Criar uma política de chave no Guia do desenvolvedor do AWS Key Management Service.

Próximas etapas

Depois de criar com sucesso uma Integração ETL zero com o Amazon SageMaker, você poderá começar a adicionar dados ao cluster do Aurora de origem e consultá-los no lakehouse do Amazon SageMaker. Os dados serão replicados automaticamente e disponibilizados para workloads de analytics e machine learning.

Filtragem de dados para Integrações ETL zero do Aurora

As Integrações ETL zero do Aurora permitem filtragem de dados, o que possibilita controlar quais dados são replicados do cluster de banco de dados do Aurora de origem para o data warehouse de destino. Em vez de replicar todo o banco de dados, você pode aplicar um ou mais filtros para incluir ou excluir seletivamente tabelas específicas. Isso ajuda a otimizar o desempenho do armazenamento e das consultas, garantindo que somente os dados relevantes sejam transferidos. No momento, a filtragem é limitada aos níveis de banco de dados e tabela. Não é possível aplicar filtragem em nível de coluna e linha.

A filtragem de dados pode ser útil quando você deseja:

- Una determinadas tabelas de dois ou mais clusters de origem diferentes para não precisar de dados completos de nenhum cluster.

- Economize custos realizando análises usando apenas um subconjunto de tabelas em vez de uma frota inteira de bancos de dados.
- Filtre informações confidenciais, como números de telefone, endereços ou detalhes do cartão de crédito, de determinadas tabelas.

É possível adicionar filtros de dados a uma integração ETL zero usando o Console de gerenciamento da AWS, a AWS Command Line Interface (AWS CLI) ou a API do Amazon RDS.

Se a integração tiver um cluster provisionado como destino, o cluster deverá estar no patch 180 ou posterior para usar a filtragem de dados.

Tópicos

- Formato de um filtro de dados
- Lógica de filtros
- Precedência do filtro
- Exemplos do Aurora MySQL
- Exemplos do Aurora PostgreSQL
- Adicionar filtros de dados a uma integração
- Remover filtros de dados de uma integração

Formato de um filtro de dados

É possível definir vários filtros para uma única integração. Cada filtro inclui ou exclui qualquer tabela de banco de dados existente e futura que corresponda a um dos padrões na expressão do filtro. As integrações ETL zero do Aurora usam a sintaxe de filtro Maxwell para a filtragem de dados.

Cada filtro tem os seguintes elementos:

Elemento	Descrição
Tipo de filtro	Um tipo de filtro `Include` inclui todas as tabelas que correspondem a um dos padrões na expressão do filtro. Um tipo de filtro `Exclude` exclui todas as tabelas que correspondem a um dos padrões.

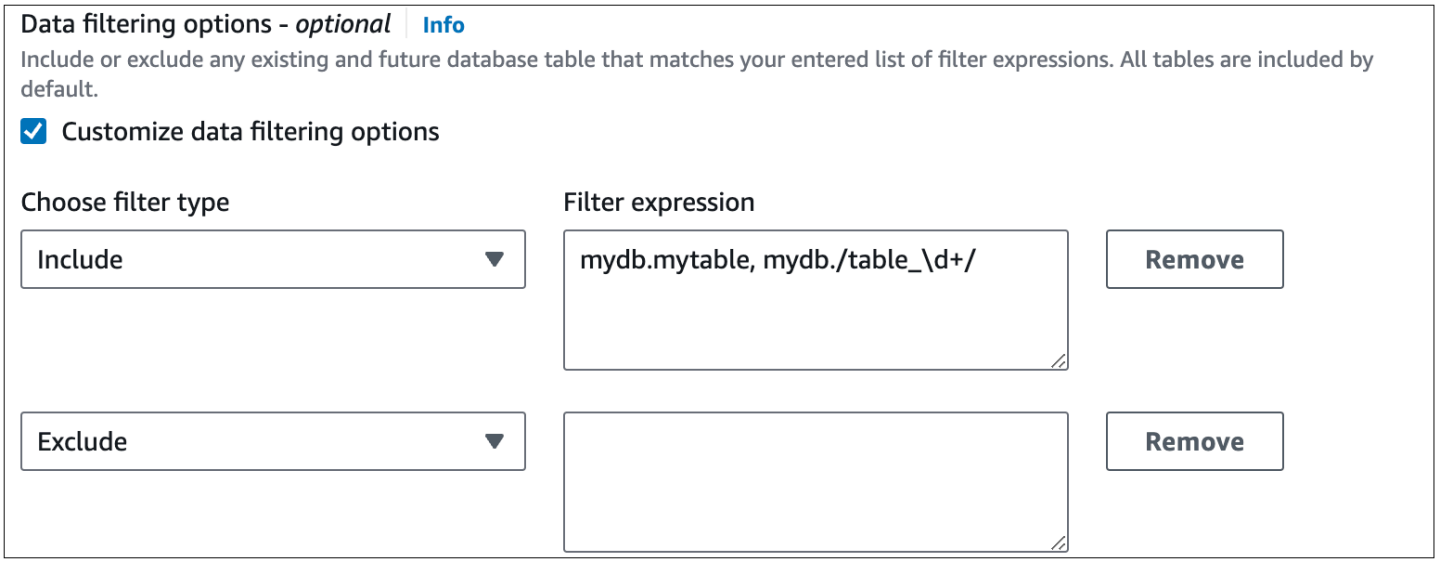
Elemento	Descrição
Expressão de filtro	Uma lista separada por vírgulas de padrões. As expressões devem usar a sintaxe de filtro Maxwell.
Pattern	Um padrão de filtro no formato *database*.*table* para o Aurora MySQL ou *database*.*schema*.*table* para o Aurora PostgreSQL. É possível especificar nomes literais ou definir expressões regulares. **Note** No Aurora MySQL, são aceitas expressões regulares em nomes de banco de dados e de tabela. No Aurora PostgreSQL, as expressões regulares são aceitas somente em nomes de esquema e de tabela, não em nomes de banco de dados. Não é possível incluir filtros ao nível da coluna nem listas de bloqueio. Uma única integração pode ter um máximo de 99 padrões no total. No console, é possível inserir padrões em uma única expressão de filtro ou distribuí-los entre várias expressões. Um único padrão não pode exceder 256 caracteres.

Important

Se você selecionar um cluster de banco de dados de origem do Aurora PostgreSQL, deverá especificar pelo menos um padrão de filtro de dados. O padrão deve incluir no mínimo

um banco de dados único (*database-name*`.*.*`) para replicação no data warehouse de destino.

A imagem a seguir mostra a estrutura dos filtros de dados do Aurora MySQL no console:



Important

Não inclua informações pessoais, confidenciais ou sigilosas em seus padrões de filtragem.

Filtros de dados na AWS CLI

Ao usar a AWS CLI para adicionar um filtro de dados, a sintaxe difere um pouco da sintaxe do console. Você deve atribuir um tipo de filtro (`Include` ou `Exclude`) a cada padrão individualmente; portanto, não é possível agrupar vários padrões sob um tipo de filtro.

Por exemplo, no console, é possível agrupar os seguintes padrões separados por vírgula em uma única instrução `Include`:

Aurora MySQL

```
mydb.mytable, mydb./table_\d+/
```

Aurora PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

No entanto, ao usar a AWS CLI, o mesmo filtro de dados deve estar no seguinte formato:

Aurora MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

Aurora PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

Lógica de filtros

Se você não especificar nenhum filtro de dados na integração, o Aurora assumirá o filtro padrão `include:*.*`, o que replica todas as tabelas no data warehouse de destino. Entretanto, se você adicionar pelo menos um filtro, a lógica padrão mudará para `exclude:*.*`, o que exclui todas as tabelas por padrão. Isso permite que você defina explicitamente quais bancos de dados e tabelas incluir na replicação.

Por exemplo, se você definir o seguinte filtro:

```
'include: db.table1, include: db.table2'
```

O Aurora avaliará o filtro da seguinte forma:

```
'exclude:*.*, include: db.table1, include: db.table2'
```

Portanto, o Aurora replica somente a `table1` e a `table2` do banco de dados denominado db para o data warehouse de destino.

Precedência do filtro

O Aurora avalia os filtros de dados na ordem em que você especificar. No Console de gerenciamento da AWS, ele processa expressões de filtro da esquerda para a direita e de cima para baixo. Um segundo filtro ou um padrão individual subsequente pode substituir o primeiro.

Por exemplo, se o primeiro filtro for `Include books.stephenking`, ele incluirá somente a tabela `stephenking` do banco de dados `books`. No entanto, se você adicionar um segundo filtro, `Exclude books.*`, ele substituirá o primeiro filtro. Isso impede que qualquer tabela do índice `books` seja replicada para o data warehouse de destino.

Quando você especifica pelo menos um filtro, a lógica primeiro assume `exclude:*.*`, o que significa que todas as tabelas são automaticamente excluídas da replicação. Como prática recomendada, defina filtros do mais amplo ao mais específico. Comece com uma ou mais instruções `Include` para especificar os dados a serem replicados e, em seguida, adicione filtros `Exclude` para remover seletivamente determinadas tabelas.

O mesmo princípio se aplica aos filtros que você define usando a AWS CLI. O Aurora avalia esses padrões de filtro na ordem em que você os especifica; portanto, um padrão pode substituir um especificado antes dele.

Exemplos do Aurora MySQL

Os seguintes exemplos demonstram como a filtragem de dados funciona para exemplos do Aurora MySQL de Integração ETL zero:

- Inclua todos os bancos de dados e todas as tabelas:

```
'include: *.*'
```

- Inclua todas as tabelas no banco de dados `books`:

```
'include: books.*'
```

- Exclua todas as tabelas chamadas `mystery`:

```
'include: *.*, exclude: *.mystery'
```

- Inclua duas tabelas específicas no banco de dados `books`:

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- Inclua todas as tabelas no banco de dados `books`, exceto aquelas que contêm a substring `mystery`:

```
'include: books.*, exclude: books./.*mystery.*/'
```

- Inclua todas as tabelas no banco de dados `books`, exceto aquelas que começam com `mystery`:

```
'include: books.*, exclude: books./mystery.*/'
```

- Inclua todas as tabelas no banco de dados `books`, exceto aquelas que terminam com `mystery`:

```
'include: books.*, exclude: books./.*mystery/'
```

- Inclua todas as tabelas no banco de dados `books` que começam com `table_`, exceto aquela chamada `table_stephen_king`. Por exemplo, `table_movies` ou `table_books` seria replicada, mas não `table_stephen_king`.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

Exemplos do Aurora PostgreSQL

Os seguintes exemplos demonstram como a filtragem de dados funciona para Integrações ETL zero do Aurora PostgreSQL:

- Inclua todas as tabelas no banco de dados `books`:

```
'include: books.*.*'
```

- Exclua todas as tabelas denominadas `mystery` no banco de dados `books`:

```
'include: books.*.*, exclude: books.*.mystery'
```

- Inclua uma tabela no banco de dados `books` no esquema `mystery` e uma tabela no banco de dados `employee` no esquema `finance`:

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- Inclua todas as tabelas no banco de dados `books` e no esquema `science_fiction`, exceto aquelas que contêm a substring `king`:

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- Inclua todas as tabelas no banco de dados `books`, exceto aquelas que têm um nome de esquema que começa com `sci`:

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- Inclua todas as tabelas no banco de dados `books`, exceto aquelas no esquema `mystery` que terminam com `king`:

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- Inclua todas as tabelas no banco de dados `books` que começam com `table_`, exceto aquela chamada `table_stephen_king`. Por exemplo, `table_movies` no esquema `fiction` e `table_books` no esquema `mystery` são replicados, mas não `table_stephen_king` em nenhum dos esquemas:

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

Adicionar filtros de dados a uma integração

É possível configurar a filtragem de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

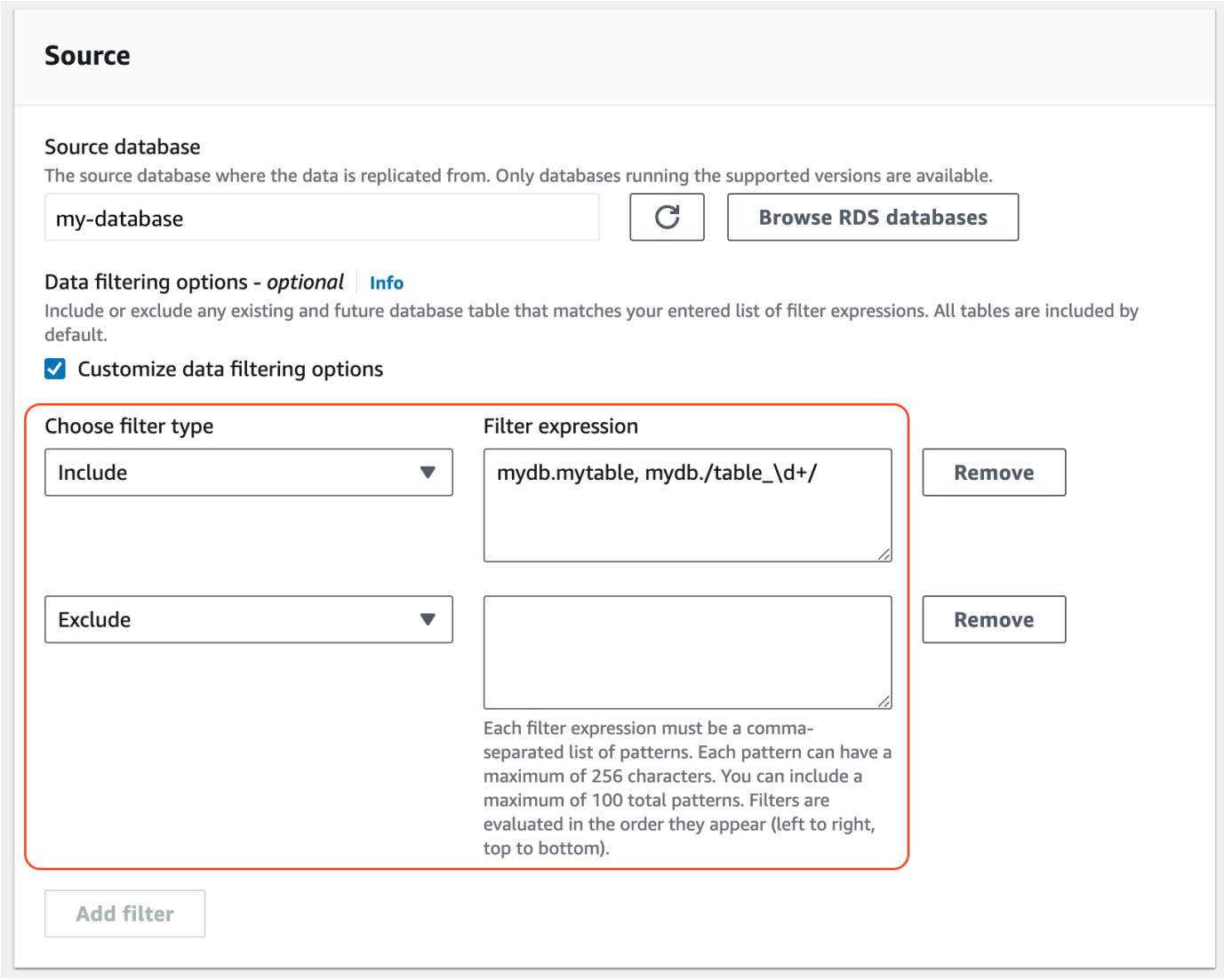
> **Important**
>
> Se você adicionar um filtro depois de criar uma integração, o Aurora o tratará como se ele sempre tivesse existido. Ele remove quaisquer dados no data warehouse de destino que não correspondam aos novos critérios de filtragem e ressincroniza todas as tabelas afetadas.

Console do RDS

Para adicionar filtros de dados para uma integração ETL zero

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Integrações ETL zero. Selecione a integração à qual você deseja adicionar filtros de dados e escolha Modificar.
3. Em Fonte, adicione uma ou mais instruções `Exclude` e `Include`.

   A seguinte imagem mostra um exemplo de filtros de dados para uma integração do MySQL:



4. Quando as alterações estiverem adequadas para você, escolha Continuar e Salvar alterações.

AWS CLI

Para adicionar filtros de dados a uma integração ETL zero usando a AWS CLI, chame o comando modify-integration. Além do identificador de integração, especifique o parâmetro `--data-filter` com uma lista separada por vírgulas dos filtros Maxwell `Include` e `Exclude`.

Example

O exemplo a seguir adiciona padrões de filtro a `my-integration`.

Para Linux, macOS ou Unix:

```
aws rds modify-integration \
```

```
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Para Windows:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

API do RDS

Para modificar uma integração ETL zero usando a API do RDS, chame a operação ModifyIntegration. Especifica o identificador de integração e fornece uma lista separada por vírgulas de padrões de filtro.

Remover filtros de dados de uma integração

Quando você remove um filtro de dados de uma integração, o Aurora reavalia os filtros restantes como se o filtro removido nunca tivesse existido. Em seguida, ele replica quaisquer dados anteriormente excluídos que agora atendem aos critérios no data warehouse de destino. Isso aciona uma ressincronização de todas as tabelas afetadas.

Adicionar dados a um cluster de banco de dados do Aurora de origem e consultá-los

Para criar uma Integração ETL zero que replica dados do Amazon Aurora no Amazon Redshift, você deve criar um banco de dados de destino pretendido.

Para conexões com o Amazon Redshift, conecte-se ao cluster ou grupo de trabalho do Amazon Redshift e crie um banco de dados com uma referência ao identificador da integração. Depois, é possível adicionar dados ao cluster de banco de dados do Aurora de origem e vê-los replicados no Amazon Redshift ou no Amazon SageMaker.

Tópicos

- Criar um banco de dados de destino
- Adicionar dados ao cluster de banco de dados de origem
- Consultar os dados do Aurora no Amazon Redshift
- Diferenças de tipos de dados entre os bancos de dados Aurora e Amazon Redshift

- Operações de DDL para o Aurora PostgreSQL

Criar um banco de dados de destino

Depois de criar uma integração, para começar a replicar dados no Amazon Redshift, você deve primeiro criar um banco de dados no data warehouse de destino. Esse banco de dados deve incluir uma referência ao identificador da integração. Você pode usar o console do Amazon Redshift ou o Editor de Consultas v2 para criar o banco de dados.

Para conferir instruções sobre como criar um banco de dados de destino, consulte Criar bancos de dados de destino no Amazon Redshift.

Adicionar dados ao cluster de banco de dados de origem

Depois de configurar a integração, você pode preencher o cluster de banco de dados do Aurora de origem com dados que você deseja replicar no data warehouse.

> **Note**
>
> Há diferenças entre os tipos de dados no Amazon Aurora e no warehouse de analytics de destino. Para obter uma tabela de mapeamentos de tipos de dados, consulte. the section called “Diferenças dos tipos de dados”

Primeiro, conecte-se ao cluster de banco de dados de origem usando o cliente do MySQL ou do PostgreSQL de sua escolha. Para instruções, consulte the section called “Conexão com um cluster de banco de dados”.

Depois, crie uma tabela e insira uma linha de dados de exemplo.

> **Important**
>
> Verifique se a tabela tem uma chave primária. Caso contrário, ela não poderá ser replicada no data warehouse de destino.

Os utilitários pg_dump e pg_restore do PostgreSQL inicialmente criam tabelas sem uma chave primária e depois as adicionam. Se você estiver usando um desses utilitários, recomendamos primeiro criar um esquema e depois carregar os dados em um comando separado.

MySQL

O exemplo a seguir usa o utilitário MySQL Workbench.

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

PostgreSQL

O exemplo a seguir usa o terminal interativo do PostgreSQL `psql`. Ao se conectar ao cluster, inclua o banco de dados nomeado que você especificou ao criar a integração.

```
psql -h mycluster.cluster-123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username
 -d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

named_db=> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

Consultar os dados do Aurora no Amazon Redshift

Depois que os dados são adicionados ao cluster de banco de dados do Aurora, eles são replicados no banco de dados de destino e podem ser consultados.

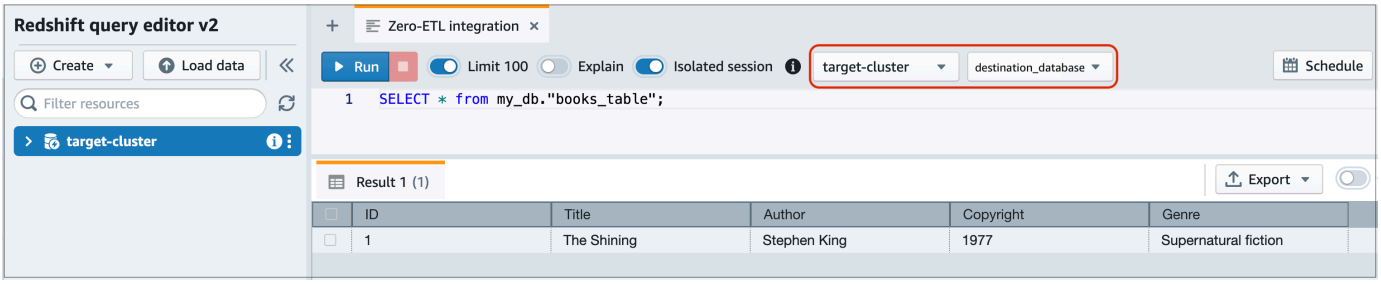
Como consultar os dados replicados

1. Navegue até o console do Amazon Redshift e escolha Editor de Consultas v2 no painel de navegação esquerdo.
2. Conecte-se ao cluster ou grupo de trabalho e escolha o banco de dados de destino (criado na integração) no menu suspenso (destination_database neste exemplo). Para conferir instruções

sobre como criar um banco de dados de destino, consulte Criar bancos de dados de destino no Amazon Redshift.

3. Use uma declaração SELECT para consultar dados. Neste exemplo, é possível executar o comando a seguir para selecionar todos os dados da tabela que você criou no cluster de banco de dados do Aurora de origem:

```
SELECT * from my_db."books_table";
```



- *my_db* é o nome do esquema do banco de dados Aurora. Essa opção só é necessária para bancos de dados do MySQL.
- *books_table* é o nome da tabela do Aurora.

Também é possível consultar os dados usando o cliente de linha de comandos. Por exemplo:

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                 Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+-----------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> Para diferenciar letras maiúsculas de minúsculas, use aspas duplas (“ ”) para nomes de esquemas, tabelas e colunas. Para obter mais informações, consulte enable_case_sensitive_identifier.

Diferenças de tipos de dados entre os bancos de dados Aurora e Amazon Redshift

Abaixo, as tabelas mostram as associações entre tipos de dados do Aurora MySQL e Aurora PostgreSQL e tipos de dados de destino correspondentes. No momento, o Amazon Aurora é compatível somente com esses tipos de dados para integrações ETL zero.

Se uma tabela no cluster de banco de dados de origem incluir um tipo de dado incompatível, a tabela ficará fora de sincronia e não poderá ser consumida no destino pretendido. O streaming da origem para o destino continuará, mas a tabela com o tipo de dados não compatível não estará disponível. Para corrigir a tabela e disponibilizá-la no destino pretendido, você deve reverter manualmente a alteração incompatível e, em seguida, atualizar a integração executando `ALTER DATABASE...INTEGRATION REFRESH`.

> **Note**
>
> Não é possível atualizar Integrações ETL zero com um lakehouse do Amazon SageMaker. Em vez disso, exclua e tente criar a integração novamente.

Tópicos

- Aurora MySQL
- Aurora PostgreSQL

Aurora MySQL

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
INT	INTEGER	Número inteiro de quatro bytes assinado	Nenhum
SMALLINT	SMALLINT	Número inteiro de dois bytes assinado	Nenhum

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
TINYINT	SMALLINT	Número inteiro de dois bytes assinado	Nenhum
MEDIUMINT	INTEGER	Número inteiro de quatro bytes assinado	Nenhum
BIGINT	BIGINT	Número inteiro de oito bytes assinado	Nenhum
INT UNSIGNED	BIGINT	Número inteiro de oito bytes assinado	Nenhum
TINYINT UNSIGNED	SMALLINT	Número inteiro de dois bytes assinado	Nenhum
MEDIUMINT UNSIGNED	INTEGER	Número inteiro de quatro bytes assinado	Nenhum
BIGINT UNSIGNED	DECIMAL(20,0)	Numérico exato com precisão selecionável	Nenhum
DECIMAL(p,s) = NUMERIC(p,s)	DECIMAL (p,s)	Numérico exato com precisão selecionável	Precisão maior que 38 e escala maior que 37 não são aceitas

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL (p,s)	Numérico exato com precisão selecionável	Precisão maior que 38 e escala maior que 37 não são aceitas
FLOAT4/REAL	REAL	Número de ponto flutuante de precisão simples	Nenhum
FLOAT4/REAL UNSIGNED	REAL	Número de ponto flutuante de precisão simples	Nenhum
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	Número de ponto flutuante de precisão dupla	Nenhum
DOUBLE/REAL/FLOAT8 UNSIGNED	DOUBLE PRECISION	Número de ponto flutuante de precisão dupla	Nenhum
BIT (n)	VARBYTE(8)	Valor binário de comprimento variável	Nenhum
BINARY(n)	VARBYTE(n)	Valor binário de comprimento variável	Nenhum

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
VARBINARY (n)	VARBYTE(n)	Valor binário de comprimento variável	Nenhum
CHAR (n)	VARCHAR(n)	Valor de string de comprimento variável	Nenhum
VARCHAR(n)	VARCHAR(n)	Valor de string de comprimento variável	Nenhum
TEXT	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
TINYTEXT	VARCHAR(255)	Valor de string de comprimento variável de até 255 caracteres	Nenhum
MEDIUMTEXT	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
LONGTEXT	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
ENUM	VARCHAR(1020)	Valor de string de comprimento variável de até 1.020 caracteres	Nenhum
SET	VARCHAR(1020)	Valor de string de comprimento variável de até 1.020 caracteres	Nenhum
DATE	DATE	Data de calendário (ano, mês, dia)	Nenhum
DATETIME	TIMESTAMP	Data e hora (sem fuso horário)	Nenhum
TIMESTAMP(p)	TIMESTAMP	Data e hora (sem fuso horário)	Nenhum
TIME	VARCHAR(18)	Valor de string de comprimento variável de até 18 caracteres	Nenhum
YEAR	VARCHAR(4)	Valor de string de comprimento variável de até 4 caracteres	Nenhum

Tipo de dados do Aurora MySQL	Tipo de dados de destino	Descrição	Limitações
JSON	SUPER	Dados ou documentos semiestruturados como valores	Nenhum

Aurora PostgreSQL

As Integrações ETL zero para o Aurora PostgreSQL não são compatíveis com tipos de dados personalizados nem tipos de dados criados por extensões.

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
array	SUPER	Dados ou documentos semiestruturados como valores	Nenhum
bigint	BIGINT	Número inteiro de oito bytes assinado	Nenhum
bigserial	BIGINT	Número inteiro de oito bytes assinado	Nenhum
bit variável (n)	VARBYTE(n)	Valor de binário de tamanho variável de até 16.777.216 bytes	Nenhum
bit(n)	VARBYTE(n)	Valor de binário de tamanho	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
		variável de até 16.777.216 bytes	
bit, bit varying	VARBYTE(16777216)	Valor de binário de tamanho variável de até 16.777.216 bytes	Nenhum
booleano	BOOLEAN	Booliano lógico (verdadeiro/ falso)	Nenhum
bytea	VARBYTE(16777216)	Valor de binário de tamanho variável de até 16.777.216 bytes	Nenhum
char(n)	CHAR (n)	Valor de string de caracteres de tamanho fixo de até 65.535 bytes	Nenhum
char varying(n)	VARCHAR(65535)	Valor de string de caracteres de comprimento variável de até 65.535 caractere s	Nenhum
cid	BIGINT	Número inteiro de oito bytes assinado	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
cidr	VARCHAR(19)	Valor de string de comprimento variável de até 19 caracteres	Nenhum
date	DATE	Data de calendário (ano, mês, dia)	Valores maiores do que 294.276 d.C. não são aceitos.
double precision	DOUBLE PRECISION	Números de ponto flutuante de precisão dupla	Valores abaixo do normal não são totalmente aceitos.
gtsvector	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
inet	VARCHAR(19)	Valor de string de comprimento variável de até 19 caracteres	Nenhum
integer	INTEGER	Número inteiro de quatro bytes assinado	Nenhum
int2vector	SUPER	Dados ou documentos semiestruturados como valores.	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
intervalo	INTERVAL	Duração	Somente os tipos de INTERVAL que especificam um qualificador de ano a mês ou de dia a segundo são aceitos.
json	SUPER	Dados ou documentos semiestruturados como valores	Nenhum
jsonb	SUPER	Dados ou documentos semiestruturados como valores	Nenhum
jsonpath	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
macaddr	VARCHAR(17)	Valor de string de comprimento variável de até 17 caracteres	Nenhum
macaddr8	VARCHAR(23)	Valor de string de comprimento variável de até 23 caracteres	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
money	DECIMAL(20,3)	Valor da moeda	Nenhum
name	VARCHAR(64)	Valor de string de comprimento variável de até 64 caracteres	Nenhum
numeric(p,s)	DECIMAL (p,s)	Valor fixo de precisão definido pelo usuário	• Valores `NaN` não são aceitos. • A precisão e a escala devem ser definidas explicitamente e não serem maiores que 38 (precisão) e 37 (escala) • Escala negativa não é aceita.
oid	BIGINT	Número inteiro de oito bytes assinado	Nenhum
oidvector	SUPER	Dados ou documentos semiestruturados como valores.	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
pg_brin_bloom_summary	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
pg_dependencies	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
pg_lsn	VARCHAR(17)	Valor de string de comprimento variável de até 17 caracteres	Nenhum
pg_mcv_list	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
pg_ndistinct	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
pg_node_tree	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
pg_snapshot	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
real	REAL	Número de ponto flutuante de precisão simples	Valores abaixo do normal não são totalmente aceitos.
refcursor	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
smallint	SMALLINT	Número inteiro de dois bytes assinado	Nenhum
smallserial	SMALLINT	Número inteiro de dois bytes assinado	Nenhum
serial	INTEGER	Número inteiro de quatro bytes assinado	Nenhum
texto	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
tid	VARCHAR(23)	Valor de string de comprimento variável de até 23 caracteres	Nenhum
hora [(p)] sem fuso horário	VARCHAR(19)	Valor de string de comprimento variável de até 19 caracteres	Os valores `Infinity` e `-Infinity` não são aceitos.
hora [(p)] com fuso horário	VARCHAR(22)	Valor de string de comprimento variável de até 22 caracteres	Os valores `Infinity` e `-Infinity` não são aceitos.
carimbo de data/hora [(p)] sem fuso horário	TIMESTAMP	Data e hora (sem fuso horário)	• Os valores `Infinity` e `-Infinity` não são aceitos. • Valores maiores do que `9999-12-31` não são aceitos. • Valores B.C. não são aceitos.

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
carimbo de data e hora [(p)] com fuso horário	TIMESTAMPTZ	Data e hora (com fuso horário)	• Os valores `Infinity` e `-Infinity` não são aceitos. • Valores maiores do que `9999-12-31` não são aceitos. • Valores B.C. não são aceitos.
tsquery	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
tsvector	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum
txid_snapshot	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum

Tipo de dados do Aurora PostgreSQL	Tipo de dados do Amazon Redshift	Descrição	Limitações
uuid	VARCHAR(36)	String de 36 caracteres de comprimento variável	Nenhum
xid	BIGINT	Número inteiro de oito bytes assinado	Nenhum
xid8	DECIMAL(20, 0)	Decimal de precisão fixa	Nenhum
xml	VARCHAR(65535)	Valor de string de comprimento variável de até 65.535 caracteres	Nenhum

Operações de DDL para o Aurora PostgreSQL

Como o Amazon Redshift é derivado do PostgreSQL, ele compartilha vários recursos com o Aurora PostgreSQL porque ambos têm em comum a arquitetura do PostgreSQL. As Integrações ETL zero utilizam essas semelhanças para simplificar a replicação de dados do Aurora PostgreSQL para o Amazon Redshift, associando bancos de dados por nome e utilizando o banco de dados compartilhado, o esquema e a estrutura de tabelas.

Pense nos seguintes pontos ao gerenciar Integrações ETL zero do Aurora PostgreSQL:

- O isolamento é gerenciado no nível do banco de dados.
- A replicação ocorre no nível do banco de dados.
- Os bancos de dados do Aurora PostgreSQL são associados aos bancos de dados do Amazon Redshift por nome, e os dados fluirão para o banco de dados do Redshift renomeado correspondente caso o original seja renomeado.

Apesar de suas semelhanças, o Amazon Redshift e o Aurora PostgreSQL têm diferenças importantes. As seções a seguir descrevem as respostas do sistema do Amazon Redshift para operações comuns de DDL.

Tópicos

- Operações do banco de dados
- Operações de esquema
- Operações de tabela

Operações do banco de dados

A tabela a seguir mostra as respostas do sistema para operações de DDL do banco de dados.

Operação de DDL	Resposta do sistema do Redshift
`CREATE DATABASE`	Nenhuma operação
`DROP DATABASE`	O Amazon Redshift descarta todos os dados no banco de dados de destino do Redshift.
`RENAME DATABASE`	O Amazon Redshift descarta todos os dados no banco de dados de destino original e os ressincroniza no novo banco de dados de destino. Se o novo banco de dados não existir, você deverá criá-lo manualmente. Para conferir instruções, consulte Criar bancos de dados de destino no Amazon Redshift.

Operações de esquema

A tabela a seguir mostra as respostas do sistema para operações de DDL do esquema.

Operação de DDL	Resposta do sistema do Redshift
`CREATE SCHEMA`	Nenhuma operação

Operação de DDL	Resposta do sistema do Redshift
`DROP SCHEMA`	O Amazon Redshift descarta o esquema original.
`RENAME SCHEMA`	O Amazon Redshift descarta o esquema original e ressincroniza os dados no novo esquema.

Operações de tabela

A tabela a seguir mostra as respostas do sistema para operações de DDL da tabela.

Operação de DDL	Resposta do sistema do Redshift
`CREATE TABLE`	O Amazon Redshift cria a tabela. Algumas operações fazem com que a criação da tabela falhe, como criar uma tabela sem uma chave primária ou realizar o particion amento declarativo. Para obter mais informaçõ es, consulte the section called “Limitações” e the section called “Solução de problemas em integrações ETL zero”.
`DROP TABLE`	O Amazon Redshift descarta a tabela.
`TRUNCATE TABLE`	O Amazon Redshift trunca a tabela.
`ALTER TABLE (RENAME... )`	O Amazon Redshift renomeia a tabela ou a coluna.
`ALTER TABLE (SET SCHEMA)`	O Amazon Redshift descarta a tabela no esquema original e a ressincroniza no novo esquema.
`ALTER TABLE (ADD PRIMARY KEY)`	O Amazon Redshift adiciona uma chave primária e ressincroniza a tabela.

Operação de DDL	Resposta do sistema do Redshift
`ALTER TABLE (ADD COLUMN)`	O Amazon Redshift adiciona uma coluna à tabela.
`ALTER TABLE (DROP COLUMN)`	O Amazon Redshift descartará a coluna se ela não for uma coluna de chave primária. Caso contrário, ele ressincronizará a tabela.
`ALTER TABLE (SET LOGGED/UNLOGGED )`	Se você alterar a tabela para ativar o registro em log, o Amazon Redshift a ressincronizará. Se você alterar a tabela para desativar o registro em log, o Amazon Redshift a descartará.

Visualizar e monitorar Integrações ETL zero do Aurora

Você pode ver os detalhes de uma integração ETL zero do Amazon Aurora para ver suas informações de configuração e status atual. Também é possível monitorar o status da integração consultando visualizações específicas do sistema no Amazon Redshift. Além disso, o Amazon Redshift publica determinadas métricas relacionadas à integração no Amazon CloudWatch, que você pode visualizar no console do Amazon Redshift.

Tópicos

- Visualizar integrações
- Monitorar integrações usando tabelas de sistema para o Amazon Redshift
- Monitorar integrações com o Amazon EventBridge para o Amazon Redshift

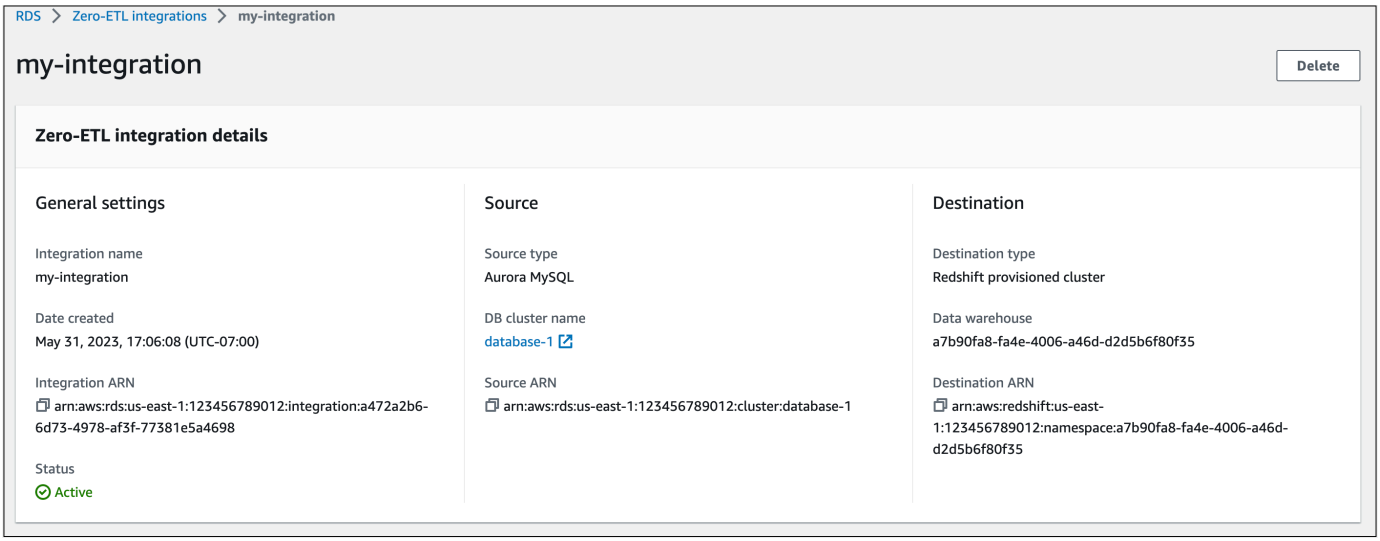
Visualizar integrações

Você pode visualizar Integrações ETL zero do Aurora usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como visualizar os detalhes de uma integração ETL zero

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação à esquerda, escolha Integrações ETL zero.
3. Selecione uma integração para ver mais detalhes sobre ela, como o cluster de banco de dados de origem e o data warehouse de destino.



Uma integração pode ter os seguintes status:

- `Creating`: a integração está sendo criada.
- `Active`: a integração está enviando dados transacionais para o data warehouse de destino.
- `Syncing`: a integração encontrou um erro recuperável e precisa reprocessar os dados. As tabelas afetadas não estarão disponíveis para consulta enquanto a ressincronização não for concluída.
- `Needs attention`: a integração encontrou um evento ou erro que requer intervenção manual para resolvê-lo. Para corrigir o problema, siga as instruções na mensagem de erro na página de detalhes da integração.
- `Failed`: a integração encontrou um evento ou erro irrecuperável que não pode ser corrigido. Você precisa excluir e recriar a integração.
- `Deleting`: a integração está sendo excluída.

AWS CLI

Para visualizar todas as integrações ETL zero na conta atual usando a AWS CLI, use o comando describe-integrations e especifique a opção `--integration-identifier`.

Example

Para Linux, macOS ou Unix:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Para Windows:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API do RDS

Para visualizar a integração ETL zero usando a API do Amazon RDS, use a operação `DescribeIntegrations` com o parâmetro `IntegrationIdentifier`.

Monitorar integrações usando tabelas de sistema para o Amazon Redshift

O Amazon Redshift tem visualizações e tabelas do sistema que contêm informações sobre como o sistema está funcionando. Você pode consultar essas visualizações e tabelas do sistema da mesma maneira como consultaria qualquer outra tabela de banco de dados. Para ter mais informações sobre visualizações e tabelas do sistema no Amazon Redshift, consulte Referência de visualizações e tabelas do sistema no Guia do desenvolvedor de banco de dados do Amazon Redshift.

Você pode consultar as seguintes visualizações e tabelas de sistema para obter informações sobre Integrações ETL zero do Aurora:

- SVV_INTEGRATION: fornece detalhes de configuração para suas integrações.
- SVV_INTEGRATION_TABLE_STATE: descreve o estado de cada tabela em uma integração.
- SYS_INTEGRATION_TABLE_STATE_CHANGE: exibe os logs de alteração do estado de uma integração.
- SYS_INTEGRATION_ACTIVITY: fornece informações sobre execuções de integração concluídas.

Todas as métricas do Amazon CloudWatch relacionadas à integração se originam no Amazon Redshift. Para ter mais informações, consulte Métricas para integrações ETL zero no Guia de gerenciamento do Amazon Redshift. No momento, o Amazon Aurora não publica nenhuma métrica relacionada à integração no Amazon CloudWatch.

Monitorar integrações com o Amazon EventBridge para o Amazon Redshift

O Amazon Redshift envia eventos relacionados à integração ao Amazon EventBridge. Para obter uma lista de eventos e seus IDs de eventos correspondentes, consulte Notificações de eventos de integração ETL zero com o Amazon EventBridge no Guia de gerenciamento do Amazon Redshift.

Modificar Integrações ETL zero do Aurora

É possível modificar somente o nome, a descrição e as opções de filtragem de dados de uma Integração ETL zero em um data warehouse compatível. Não é possível modificar a chave do AWS KMS usada para criptografar a integração ou os bancos de dados de origem ou de destino.

Se você adicionar um filtro de dados a uma integração existente, o Aurora reavaliará o filtro como se ele sempre tivesse existido. Ele remove todos os dados que no momento estão no data warehouse de destino e não correspondem aos novos critérios de filtragem. Se você remover um filtro de dados de uma integração, ele replicará todos os dados que anteriormente não correspondiam aos critérios de filtragem (mas agora correspondem) no data warehouse de destino. Para obter mais informações, consulte the section called “Filtragem de dados para integrações ETL zero”.

É possível modificar uma integração ETL zero usando o Console de gerenciamento da AWS, a AWS CLI ou a API do Amazon RDS.

Console do RDS

Para modificar uma integração ETL zero

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Integrações ETL zero e a integração que você deseja modificar.
3. Escolha Modificar e altere as configurações disponíveis.
4. Quando todas as alterações estiverem conforme desejado, escolha Modificar.

AWS CLI

Para modificar uma integração ETL zero usando a AWS CLI, chame o comando modify-integration. Junto com o `--integration-identifier`, especifique qualquer uma das seguintes opções:

- `--integration-name`: especifique um novo nome para a integração.
- `--description`: especifique uma nova descrição para a integração.
- `--data-filter`: especifique as opções de filtragem de dados para a integração. Para obter mais informações, consulte the section called “Filtragem de dados para integrações ETL zero”.

Example

A solicitação a seguir modifica uma integração existente.

Para Linux, macOS ou Unix:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Para Windows:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

API do RDS

Para modificar uma integração ETL zero usando a API do RDS, chame a operação ModifyIntegration. Especifique o identificador de integração e os parâmetros que você deseja modificar.

Excluir Integrações ETL zero do Aurora

Quando você exclui uma integração ETL zero, o Amazon Aurora a remove do cluster de banco de dados do Aurora de origem. Os dados transacionais não são excluídos do Amazon Aurora ou do destino de analytics, mas o Aurora não envia dados novos ao Amazon Redshift ou ao Amazon SageMaker.

Só será possível excluir uma integração quando ela tiver o status `Active`, `Failed`, `Syncing` ou `Needs attention`.

É possível excluir as integrações ETL zero usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Como excluir uma integração ETL zero

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação à esquerda, escolha Integrações ETL zero.
3. Selecione a integração ETL zero que você deseja excluir.
4. Escolha Ações e Excluir, depois confirme a exclusão.

AWS CLI

Para excluir uma integração ETL zero, use o comando delete-integration e especifique a opção `--integration-identifier`.

Example

Para Linux, macOS ou Unix:

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Para Windows:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API do RDS

Para excluir uma integração ETL zero usando a API do Amazon RDS, use a operação `DeleteIntegration` com o parâmetro `IntegrationIdentifier`.

Solução de problemas em Integrações ETL zero do Aurora

Você pode verificar o estado de uma Integração ETL zero consultando a tabela de sistema SVV_INTEGRATION no destino de analytics. Se a coluna `state` tiver um valor de `ErrorState`, isso significa que há algo errado. Para obter mais informações, consulte the section called “Monitorar usando tabelas do sistema”.

Use as informações a seguir para solucionar problemas comuns em Integrações ETL zero do Aurora.

> **Important**
>
> As operações de ressincronização e atualização não estão disponíveis para Integrações ETL zero com um lakehouse do Amazon SageMaker AI. Se houver problemas com uma integração, você deve excluí-la e criar outra. Não é possível atualizar ou ressincronizar uma integração existente.

Tópicos

- Não consigo criar uma integração ETL zero.
- Minha integração está travada em um estado de Syncing.
- Minhas tabelas não estão sendo replicadas para o Amazon Redshift.
- Uma ou mais das minhas tabelas do Amazon Redshift exigem ressincronização.
- Problemas de falha em Integrações ETL zero com um lakehouse do Amazon SageMaker AI
- As alterações de DDL estão no Amazon Redshift antes que a transação de DDL seja concluída para o Aurora PostgreSQL

Não consigo criar uma integração ETL zero.

Se você não conseguir criar uma integração ETL zero, verifique se as seguintes informações estão corretas para o banco de dados de origem:

- O banco de dados de origem deve estar executando uma versão de mecanismo de banco de dados compatível. Para conferir uma lista de versões compatíveis, consulte the section called “Integrações ETL zero ”.
- Você configurou corretamente os parâmetros de banco de dados. Se os parâmetros necessários forem definidos incorretamente ou não estiverem associados ao banco de dados, a criação falhará.

Consulte the section called “Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado”.

Além disso, confira se o indicado a seguir está correto para seu data warehouse de destino:

- Ter a diferenciação entre maiúsculas e minúsculas habilitada. Consulte Ativar a diferenciação entre maiúsculas e minúsculas em um data warehouse.
- Você adicionou a entidade principal autorizada e a origem de integração corretas. Consulte Configurar a autorização para o data warehouse do Amazon Redshift.
- O data warehouse é criptografado (se for um cluster provisionado). Consulte Criptografia de banco de dados do Amazon Redshift.

Minha integração está travada em um estado de **`Syncing`**.

Sua integração poderá mostrar consistentemente um status de `Syncing` se você alterar o valor de um dos parâmetros obrigatórios do banco de dados.

Para corrigir esse problema, confira os valores dos parâmetros no grupo de parâmetros associado ao cluster de banco de dados de origem e verifique se eles correspondem aos valores obrigatórios. Para obter mais informações, consulte the section called “Etapa 1: criar um grupo de parâmetros de cluster de banco de dados personalizado”.

Se você modificar algum parâmetro, reinicialize o cluster de banco de dados para aplicar as alterações.

Minhas tabelas não estão sendo replicadas para o Amazon Redshift.

Caso não veja uma ou mais tabelas refletidas no Amazon Redshift, execute um dos seguintes comandos para ressincronizá-las:

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

Para ter mais informações, consulte ALTER DATABASE na referência do Amazon Redshift SQL.

Talvez seus dados não estejam sendo replicados porque uma ou mais tabelas de origem não têm uma chave primária. O painel de monitoramento no Amazon Redshift exibe o status dessas tabelas como `Failed`, e o status da integração ETL zero geral muda para `Needs attention`. Para resolver esse problema, é possível identificar uma chave em sua tabela que pode se tornar uma

chave primária ou adicionar uma chave primária sintética. Para conhecer as soluções detalhadas, consulte os seguintes recursos:

- Handle tables without primary keys while creating Amazon Aurora MySQL or Amazon RDS for MySQL zero-ETL integrations with Amazon Redshift
- Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift

Uma ou mais das minhas tabelas do Amazon Redshift exigem ressincronização.

A execução de determinados comandos no banco de dados de origem pode exigir que suas tabelas sejam novamente sincronizadas. Nesses casos, a visualização do sistema SVV_INTEGRATION_TABLE_STATE mostra um `table_state` de `ResyncRequired`, o que significa que a integração deve recarregar completamente os dados dessa tabela específica do MySQL para o Amazon Redshift.

Quando a tabela começa a ser ressincronizada, ela entra em um estado de `Syncing`. Você não precisa realizar nenhuma ação manual para ressincronizar uma tabela. Enquanto os dados da tabela estiverem sendo sincronizados novamente, você não poderá acessá-los no Amazon Redshift.

A seguir estão alguns exemplos de operações que podem colocar uma tabela em um estado `ResyncRequired` e possíveis alternativas a serem consideradas.

Operation	Exemplo	Alternativa
Adicionar uma coluna em uma posição específica	`ALTER TABLE table_name ADD COLUMN column_name INTEGER NOT NULL first;`	O Amazon Redshift não oferece suporte à adição de colunas em posições específicas usando as palavras-chave `first` ou `afte`

Operation	Exemplo	Alternativa
		Se a ordem das colunas na tabela de destino não for crítica, adicione a coluna no final da tabela usando um comando mais simples: `ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Operation	Exemplo	Alternativa
Adicionar uma coluna de carimbo de data/hora com o CURRENT_TIMESTAMP pad	`ALTER TABLE table_name ADD COLUMN column_name TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP;`	O valor CURRENT_TIMESTAMP das linhas da tabela é calculado pelo Aurora MySQL e não pode ser simulado no Amazon Redshift sem uma nova sincronização completa dos dados da tabela. Se possível, altere o valor padrão para uma constante literal, como 2023-01-01 00:00:15, a fim de evitar latência na disponibilidade da tabela.

Operation	Exemplo	Alternativa
Executar várias operações de coluna em um único comando	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	Considere dividir o comando em duas operações separadas, `ADD` e `RENAME`, que não exigirão ressincronização.

Problemas de falha em Integrações ETL zero com um lakehouse do Amazon SageMaker AI

Se você encontrar problemas em uma Integração ETL zero existente com um lakehouse do Amazon SageMaker AI, a única solução é excluir a integração e criar outra. Diferentemente de outros serviços da AWS, as Integrações ETL zero não permitem operações de atualização ou ressincronização.

Para resolver problemas de integração:

1. Exclua a Integração ETL zero que apresenta problemas usando o console, a CLI ou a API.
2. Verifique se as configurações do banco de dados de origem e do data warehouse de destino estão corretas.
3. Crie outra Integração ETL zero com a mesma configuração ou a configuração atualizada.

Esse processo provocará a reinicialização completa do pipeline de dados, o que pode levar algum tempo, dependendo do tamanho do banco de dados de origem.

As alterações de DDL estão no Amazon Redshift antes que a transação de DDL seja concluída para o Aurora PostgreSQL

As alterações de DDL podem aparecer no Amazon Redshift antes que uma operação de DDL termine nas integrações ETL zero do Aurora PostgreSQL. Para obter mais informações, consulte Operações de DDL para o Aurora PostgreSQL.

Usar o Aurora Serverless v2

O Aurora Serverless v2 é uma configuração sob demanda e de autoescalabilidade do Amazon Aurora. O Aurora Serverless v2 ajuda a automatizar os processos de monitoramento da workload e ajustar a capacidade para seus bancos de dados. A capacidade é ajustada automaticamente com base na demanda da aplicação. Você será cobrado apenas pelos recursos que seus clusters de banco de dados consumirem. Dessa forma, o Aurora Serverless v2 pode ajudar você a ficar dentro do orçamento e evitar pagar pelos recursos computacionais não utilizados.

Esse tipo de automação é especialmente valioso para bancos de dados multilocatário, bancos de dados distribuídos, sistemas de desenvolvimento e teste e outros ambientes com workloads altamente variáveis e imprevisíveis.

Tópicos

- Casos de uso do Aurora Serverless v2
- Vantagens do Aurora Serverless v2
- Como funciona o Aurora Serverless v2
- Requisitos e limitações do Aurora Serverless v2
- Criar um cluster de banco de dados que usa o Aurora Serverless v2
- Gerenciar clusters de banco de dados do Aurora Serverless v2
- Performance e escalabilidade no Aurora Serverless v2
- Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2
- Migrar para o Aurora Serverless v2

Casos de uso do Aurora Serverless v2

O Aurora Serverless v2 é compatível com muitos tipos de workload de banco de dados Elas variam desde ambientes de desenvolvimento e teste até sites e aplicações com workloads imprevisíveis e aplicações mais exigentes e essenciais para os negócios que exigem alta escala e disponibilidade.

O Aurora Serverless v2 é especialmente útil para os seguintes casos de uso:

- Workloads variáveis: você está executando workloads que têm aumentos repentinos e imprevisíveis na atividade. Um exemplo é um site de tráfego que tem um surto de atividades quando começa a chover. Outro é um site de comércio eletrônico com maior tráfego quando

você oferece vendas ou promoções especiais. Com o Aurora Serverless v2, seu banco de dados faz escalabilidade automática da capacidade para atender às necessidades da carga de pico da aplicação e reduz a escala novamente na vertical quando o pico de atividades termina. Com o Aurora Serverless v2, não é mais necessário provisionar para capacidade de pico ou média. Você pode especificar um limite de capacidade posterior para lidar com a pior situação, e essa capacidade não é usada, a menos que seja necessária.

A granularidade da escalabilidade no Aurora Serverless v2 ajuda você a combinar a capacidade de forma próxima às necessidades do seu banco de dados. Para um cluster provisionado, o aumento da escala na vertical exige a adição de uma instância de banco de dados inteiramente nova. No caso de um Aurora Serverless v1, o aumento da escala na vertical exige dobrar o número de unidades de capacidade (ACUs) do Aurora para o cluster, por exemplo, de 16 para 32 ou de 32 para 64. Em contrapartida, o Aurora Serverless v2 pode adicionar meia ACU quando é necessário apenas um pouco mais de capacidade. Ele pode adicionar 0,5, 1, 1,5, 2 ou meia ACU com base na capacidade adicional necessária para lidar com um aumento na workload. E ele pode remover 0,5, 1, 1,5, 2 ou meia ACU quando a workload diminui e essa capacidade não é mais necessária.

- Multi-tenant applications (Aplicações multilocatários): com o Aurora Serverless v2, não é necessário gerenciar individualmente a capacidade do banco de dados para cada aplicação em sua frota. O Aurora Serverless v2 gerencia a capacidade do banco de dados individual para você.

  É possível criar um cluster para cada locatário. Dessa forma, você pode usar recursos como clonagem, restauração de snapshot e bancos de dados globais do Aurora para aprimorar a alta disponibilidade e a recuperação de desastres, conforme apropriado para cada locatário.

  Cada locatário pode ter períodos ociosos e de ocupação específicos, dependendo da hora do dia, da época do ano, dos eventos promocionais, etc. Cada cluster pode ter um amplo intervalo de capacidade. Dessa forma, os clusters com baixa atividade geram cobranças mínimas de instância de banco de dados. Qualquer cluster pode ter a escala aumentada na vertical rapidamente para lidar com períodos de alta atividade.
- Novas aplicações: você está implantando uma nova aplicação e não tem certeza sobre o tamanho da instância de banco de dados de que precisa. Com o uso do Aurora Serverless v2, é possível configurar um cluster com uma ou várias instâncias de banco de dados e realizar a escalabilidade automática do banco de dados para os requisitos de capacidade de sua aplicação.
- Aplicativos de uso misto: suponha que você tenha uma aplicação de processamento de transações on-line (OLTP), mas periodicamente ocorrem picos no tráfego de consultas. Ao especificar níveis de promoção para as instâncias de banco de dados do Aurora Serverless v2 em

um cluster, você pode configurar seu cluster para que as instâncias de banco de dados do leitor possam ser escaladas independentemente da instância de banco de dados do gravador para lidar com a carga adicional. Quando o pico de uso diminui, as instâncias de banco de dados do leitor têm a escala reduzida na vertical para corresponder à capacidade da instância de banco de dados do gravador.

- Planejamento de capacidade: suponha que você geralmente ajuste a capacidade do banco de dados ou verifique a capacidade ideal do banco de dados para sua workload, modificando as classes de todas as instâncias de banco de dados em um cluster. Com o Aurora Serverless v2, é possível evitar essa sobrecarga administrativa. É possível determinar a capacidade mínima e máxima apropriada executando a workload e verificando o quanto as instâncias de banco de dados realmente são escaladas.

  Você pode modificar as instâncias de banco de dados existentes de provisionadas para Aurora Serverless v2 ou de Aurora Serverless v2 para provisionado. Não é necessário criar um cluster nem uma instância de banco de dados nesses casos.

  Com um banco de dados global do Aurora, talvez você não precise de tanta capacidade para os clusters secundários quanto para o cluster principal. É possível usar instâncias de banco de dados do Aurora Serverless v2 nos clusters secundários. Dessa forma, será possível aumentar a escala da capacidade do cluster na vertical se uma região secundária for promovida e assumir a workload de sua aplicação.
- Desenvolvimento e teste: além de executar as aplicações mais exigentes, você também pode usar o Aurora Serverless v2 em ambientes de desenvolvimento e teste. Com o Aurora Serverless v2, você pode criar instâncias de banco de dados com uma capacidade mínima baixa em vez de usar classes de instância de banco de dados db.t* com capacidade de intermitência. É possível definir a capacidade máxima alta o suficiente para que essas instâncias de banco de dados ainda possam executar workloads substanciais sem ficar com memória insuficiente. Quando o banco de dados não está em uso, todas as instâncias de banco de dados têm a escala reduzida na vertical para evitar cobranças desnecessárias.

  > **Tip**
  >
  > Para tornar conveniente usar o Aurora Serverless v2 em ambientes de desenvolvimento e teste, o Console de gerenciamento da AWS fornece o atalho Easy create (Criação fácil) ao criar um cluster. Se escolher a opção Dev/Test (Dev/teste), o Aurora criará um cluster com

uma instância de banco de dados do Aurora Serverless v2 e um intervalo de capacidade típico para um sistema de desenvolvimento e teste.

Usar o Aurora Serverless v2 para workloads provisionadas existentes

Suponha que você já tenha um Aurora em execução em um cluster provisionado. Você pode verificar como a aplicação funcionaria com o Aurora Serverless v2 adicionando uma ou mais instâncias de banco de dados do Aurora Serverless v2 ao cluster existente como instâncias de banco de dados do leitor. Você pode conferir com que frequência a escala das instâncias de banco de dados do leitor é aumentada ou reduzida na vertical. É possível usar o mecanismo de failover do Aurora para promover uma instância de banco de dados do Aurora Serverless v2 ao gravador e conferir como ela lida com a workload de leitura/gravação Dessa forma, é possível alternar com o mínimo de tempo de inatividade e sem alterar o endpoint que suas aplicações cliente usam. Para obter detalhes sobre o procedimento para converter clusters existentes em Aurora Serverless v2, consulte Migrar para o Aurora Serverless v2.

Vantagens do Aurora Serverless v2

O Aurora Serverless v2 destina-se a workloads variáveis. Com essas workloads imprevisíveis, você pode ter dificuldade em planejar quando alterar a capacidade do banco de dados. Você também pode ter problemas para fazer alterações de capacidade com rapidez suficiente usando os mecanismos conhecidos, como adicionar instâncias de banco de dados ou alterar classes de instâncias de banco de dados. O Aurora Serverless v2 oferece as seguintes vantagens para ajudar nesses casos de uso:

- Gerenciamento de capacidade mais simples do que o provisionado: o Aurora Serverless v2 reduz o esforço para planejar tamanhos de instâncias de banco de dados e redimensioná-las conforme a workload muda. Ele também reduz o esforço para manter a capacidade consistente para todas as instâncias de banco de dados em um cluster.
- Escalabilidade mais rápida e fácil durante períodos de alta atividade: o Aurora Serverless v2 escala a capacidade computacional e de memória, conforme necessário, sem interrupções nas transações do cliente nem em sua workload geral. A capacidade de usar instâncias de banco de dados do leitor como Aurora Serverless v2 ajuda você a aproveitar a escalabilidade horizontal e vertical. Com a capacidade de usar bancos de dados globais do Aurora, você pode distribuir sua workload de leitura do Aurora Serverless v2 por vários Regiões da AWS. Esse recurso é mais

conveniente do que os mecanismos de escalabilidade de clusters provisionados. Também é mais rápido e mais granular do que os recursos de escalabilidade do Aurora Serverless v1.

- Economia durante períodos de baixa atividade: o Aurora Serverless v2 ajuda você a evitar o provisionamento excessivo de suas instâncias de banco de dados. O Aurora Serverless v2 adiciona recursos em incrementos granulares quando a escala das instâncias de banco de dados é aumentada na vertical. É possível pagar apenas pelos recursos de banco de dados consumidos. O uso de recursos do Aurora Serverless v2 é medido por segundo. Dessa forma, quando a escala de uma instância de banco de dados é reduzida na vertical, o uso reduzido de recursos é registrado imediatamente.
- Maior paridade de recursos com o provisionado: você pode usar muitos recursos do Aurora com o Aurora Serverless v2 que não estão disponíveis para o Aurora Serverless v1. Por exemplo, com o Aurora Serverless v2, é possível usar instâncias de banco de dados do leitor, bancos de dados globais, autenticação do banco de dados do AWS Identity and Access Management (IAM) e Performance Insights. Você também pode usar muito mais parâmetros de configuração do que com o Aurora Serverless v1.

  Especificamente, com o Aurora Serverless v2, é possível aproveitar os seguintes recursos de clusters provisionados:

  - Instâncias de banco de dados do leitor: o Aurora Serverless v2 pode aproveitar as instâncias de banco de dados do leitor para escalar na horizontal. Quando um cluster contém uma ou mais instâncias de banco de dados do leitor, ele pode realizar failover imediatamente em caso de problemas com a instância de banco de dados do gravador. Este é um recurso que não está disponível no Aurora Serverless v1.
  - Clusters multi-AZ: você pode distribuir as instâncias de banco de dados do Aurora Serverless v2 de um cluster em várias zonas de disponibilidade (AZs). A configuração de um cluster multi-AZ ajuda a garantir a continuidade dos negócios, mesmo no caso raro de problemas que afetam uma AZ inteira. Este é um recurso que não está disponível no Aurora Serverless v1.
  - Bancos de dados globais: é possível usar o Aurora Serverless v2 em combinação com bancos de dados globais do Aurora para criar cópias somente leitura adicionais de seu cluster em outro Regiões da AWS para fins de recuperação de desastres.
  - Proxy do RDS: é possível usar o proxy do Amazon RDS para permitir que suas aplicações agrupem e compartilhem conexões de banco de dados para melhorar sua capacidade de escalar.
- Escalabilidade mais rápida, mais granular e menos disruptiva em comparação com o Aurora Serverless v1: é possível reduzir ou aumentar a escala na vertical do Aurora Serverless v2 com

maior rapidez. A escalabilidade pode alterar a capacidade em apenas 0,5 ACUs em vez de dobrar ou reduzir pela metade o número de ACUs. A escalabilidade geralmente acontece sem nenhuma pausa no processamento. A escalabilidade não envolve um evento do qual você precise estar ciente, como acontece com o Aurora Serverless v1. A escalabilidade pode acontecer enquanto as instruções SQL estão em execução e as transações estão abertas, sem a necessidade de esperar por um ponto silencioso.

Como funciona o Aurora Serverless v2

A visão geral a seguir descreve como Aurora Serverless v2 funciona.

Tópicos

- Aurora Serverless v2Visão geral do
- Configurações para clusters de banco de dados do Aurora
- Aurora Serverless v2Capacidade do
- Aurora Serverless v2Escalabilidade do
- Aurora Serverless v2 e alta disponibilidade
- Aurora Serverless v2 e armazenamento
- Parâmetros de configuração para clusters do Aurora

Aurora Serverless v2Visão geral do

Amazon Aurora Serverless v2 O é adequado para as workloads mais exigentes e altamente variáveis. Por exemplo, o uso do banco de dados pode ser pesado por um curto período de tempo, seguido por longos períodos de atividade leve ou nenhuma atividade. Alguns exemplos são sites de varejo, jogos ou esportes com eventos promocionais periódicos e bancos de dados que produzem relatórios quando necessário. Outros são ambientes de desenvolvimento e teste e novas aplicações em que o uso pode aumentar rapidamente. Para casos como esses e muitos outros, configurar a capacidade corretamente antecipadamente nem sempre é possível com o modelo provisionado. Também pode resultar em custos mais elevados se você provisionar em excesso e tem capacidade que você não usa.

Em contrapartida, os clusters provisionados do Aurora são adequados para workloads estáveis. Com clusters provisionados, você escolhe uma classe de instância de banco de dados que tenha uma quantidade predefinida de memória, potência da CPU, largura de banda de E/S, etc. Se sua

workload for alterada, modifique manualmente a classe da instância do gravador e dos leitores. O modelo provisionado funciona bem quando você pode ajustar a capacidade antes dos padrões de consumo esperados e é aceitável ter breves interrupções enquanto você altera a classe da instância do gravador e dos leitores do cluster.

O Aurora Serverless v2 é projetado desde o início para oferecer compatibilidade com clusters de banco de dados sem servidor que são instantaneamente escaláveis. O Aurora Serverless v2 foi projetado para fornecer o mesmo grau de segurança e isolamento que com gravadores e leitores provisionados. Esses aspectos são cruciais em ambientes de nuvem sem servidor multilocatário. O mecanismo de escalabilidade dinâmica tem muito pouca sobrecarga para que ele possa responder rapidamente às alterações na workload do banco de dados. Também é poderoso o suficiente para atender a aumentos dramáticos na demanda de processamento.

Usando o Aurora Serverless v2, é possível criar um cluster de banco de dados do Aurora sem ficar limitado a uma capacidade de banco de dados específica para cada gravador e leitor. Você especifica apenas o intervalo mínimo e máximo de capacidade. O Aurora escala cada gravador ou leitor do Aurora Serverless v2 no cluster dentro desse intervalo de capacidade. Ao usar um cluster multi-AZ no qual cada gravador ou leitor possa ser escalado dinamicamente, você pode aproveitar a escalabilidade dinâmica e a alta disponibilidade.

Aurora Serverless v2O escala os recursos do banco de dados automaticamente com base nas especificações da capacidade mínima e máxima. A escalabilidade é rápida porque a maioria das operações de eventos de escalabilidade mantém o gravador ou o leitor no mesmo host. Nos raros casos em que um gravador ou leitor do Aurora Serverless v2 é movido de um host para outro, o Aurora Serverless v2 gerencia as conexões automaticamente. Você não precisa alterar o código da aplicação cliente do seu banco de dados nem suas strings de conexão do banco de dados.

Com o Aurora Serverless v2, como acontece com os clusters provisionados, a capacidade de armazenamento e a capacidade computacional são separadas. Quando nos referimos à capacidade e à escalabilidade do Aurora Serverless v2, é sempre a capacidade computacional que está sendo aumentada ou reduzida. Assim, seu cluster pode conter muitos terabytes de dados, mesmo quando a capacidade da CPU e da memória são reduzidas para níveis baixos.

Em vez de provisionar e gerenciar servidores, você especifica a capacidade do banco de dados. Para obter detalhes sobre a capacidade do Aurora Serverless v2, consulte Aurora Serverless v2Capacidade do. A capacidade real de cada gravador ou leitor do Aurora Serverless v2 varia com o tempo, dependendo da workload. Para obter detalhes sobre esse mecanismo, consulte Aurora Serverless v2Escalabilidade do.

> **Important**
>
> Com o Aurora Serverless v1, seu cluster tem uma única medida de capacidade computacional que pode ser escalada entre os valores de capacidade mínima e máxima. Com o Aurora Serverless v2, seu cluster pode conter leitores além do gravador. Todos os gravadores e leitores do Aurora Serverless v2 podem escalar entre os valores mínimo e máximo de capacidade. Dessa forma, a capacidade total de seu cluster do Aurora Serverless v2 depende do intervalo de capacidade definido para o cluster de banco de dados e do número de gravadores e leitores no cluster. Em algum momento específico, você será cobrado somente pela capacidade de Aurora Serverless v2 que está sendo usada ativamente em seu cluster de banco de dados do Aurora.

Configurações para clusters de banco de dados do Aurora

Para cada um dos clusters de banco de dados do Aurora, você pode escolher qualquer combinação de capacidade do Aurora Serverless v2, capacidade provisionada ou ambas.

É possível configurar um cluster que contenha a capacidade do Aurora Serverless v2e a capacidade provisionada, denominada cluster de configuração mista. Por exemplo, suponha que você precise de mais capacidade de leitura/gravação do que está disponível para um gravador do Aurora Serverless v2. Nesse caso, é possível configurar o cluster com um gravador provisionado muito grande. Nesse caso, você ainda pode usar o Aurora Serverless v2 para os leitores. Ou suponha que a workload de gravação do cluster varie, mas a workload de leitura permaneça estável. Nesse caso, você pode configurar o cluster com um gravador do Aurora Serverless v2 e um ou mais leitores provisionados.

Você também pode configurar um cluster de banco de dados no qual toda a capacidade seja gerenciada pelo Aurora Serverless v2. Para fazer isso, é possível criar um cluster e usar o Aurora Serverless v2 desde o início. Ou você pode substituir toda a capacidade provisionada em um cluster existente por Aurora Serverless v2. Por exemplo, alguns dos caminhos de atualização de versões mais antigas do mecanismo exigem começar com um gravador provisionado e substituí-lo por um gravador do Aurora Serverless v2. Para saber os procedimentos para criar um cluster de banco de dados com o Aurora Serverless v2 ou para alternar um cluster de banco de dados existente para o Aurora Serverless v2, consulte Criar um cluster de banco de dados do Aurora Serverless v2 e Alternar de um cluster provisionado para o Aurora Serverless v2.

Se você nunca usa o Aurora Serverless v2 em um cluster de banco de dados, todos os gravadores e leitores no cluster de banco de dados são provisionados. Esse é o tipo mais antigo e mais comum de

cluster de banco de dados que a maioria dos usuários conhece. De fato, antes do Aurora Serverless, não havia um nome especial para esse tipo de cluster de banco de dados do Aurora. A capacidade provisionada é constante. As cobranças são relativamente fáceis de prever. No entanto, você precisa prever antecipadamente de quanta capacidade precisa. Em alguns casos, suas previsões podem ser imprecisas ou suas necessidades de capacidade podem mudar. Nesses casos, seu cluster de banco de dados pode ficar subprovisionado (mais lento do que o desejável) ou superprovisionado (mais caro do que o desejável).

Aurora Serverless v2Capacidade do

A unidade de medida para Aurora Serverless v2 é a Unidade de capacidade do Aurora (ACU). A capacidade do Aurora Serverless v2 não está vinculada às classes de instância de banco de dados que você usa para clusters provisionados.

Cada ACU é uma combinação de aproximadamente 2 gigabytes (GB) de memória, CPU correspondente e rede. Você deve especificar o intervalo de capacidade do banco de dados usando essa unidade de medida. As métricas `ServerlessDatabaseCapacity` e `ACUUtilization` ajudam você a determinar quanta capacidade o banco de dados está realmente usando e em que ponto está essa capacidade no intervalo especificado.

A qualquer momento, cada gravador do Aurora Serverless v2 ou leitor de banco de dados tem uma capacidade. A capacidade é representada como um número de ponto flutuante representando ACUs. A capacidade aumenta ou diminui sempre que o gravador ou o leitor é escalado. Esse valor é medido a cada segundo. Para cada cluster de banco de dados em que você pretende usar o Aurora Serverless v2, é necessário definir um intervalo de capacidade: os valores mínimo e máximo de capacidade que cada escritor ou leitor do Aurora Serverless v2 pode escalar entre eles. O intervalo de capacidade é a mesma para cada gravador ou leitor do Aurora Serverless v2 em um cluster de banco de dados. Cada gravador ou leitor do Aurora Serverless v2 tem sua própria capacidade, inserindo-se em algum ponto desse intervalo.

A tabela a seguir mostra os intervalos de capacidade do Aurora Serverless v2 e a versão de mecanismo compatível do Aurora MySQL e do Aurora PostgreSQL.

Intervalo de capacidade (ACUs)	Versões compatíveis do Aurora MySQL	Versões compatíveis do Aurora PostgreSQL
De 0,5 a 128	3.02.0 e posterior	13.6 e posteriores, 14.3 e posteriores, 15.2 e posteriores, 16.1 e posteriores
De 0,5 a 256	3.06.0 e posterior	13.13 e posteriores, 14.10 e posteriores, 15.5 e posteriores, 16.1 e posteriores
0-256	3.08.0 e posterior	13.15 e posteriores, 14.12 e posteriores, 15.7 e posteriores, 16.3 e posteriores

A tabela a seguir mostra as versões da plataforma do Aurora Serverless v2 com os respectivos intervalos de ACU e características de desempenho.

Aurora Serverless v2Versão da plataforma	Intervalo de ACU	desempenho
1	0-128	Desempenho de linha de base
2	0-256	Desempenho de linha de base
3	0-256	Desempenho aprimorado em até 30% em comparação com a versão 2 da plataforma.

> **Note**
>
> O intervalo de ajuste de escala disponível para um cluster específico é determinado tanto pela versão do mecanismo quanto pela versão da plataforma. É possível ter uma versão de mecanismo mais capaz em execução em uma versão da plataforma menos capaz e vice-versa. O intervalo de ajuste de escala é determinado pela versão do mecanismo ou da plataforma menos capaz.

Você pode determinar em qual versão da plataforma seu cluster está sendo executado na seção "Configuração da instância" do Console de gerenciamento da AWS ou por meio da API, visualizando a `ServerlessV2PlatformVersion` para um DBCluster.

A menor capacidade do Aurora Serverless v2 que você pode definir é 0 ACUs, para versões do Aurora Serverless v2 que permitem o uso do recurso de pausa automática. É possível especificar um número maior se for menor ou igual ao valor máximo da capacidade. Definir a capacidade mínima para um número pequeno permite que clusters de banco de dados levemente carregados consumam recursos de computação mínimos. Ao mesmo tempo, eles ficam prontos para aceitar conexões imediatamente e aumentam a escala na vertical quando ficam ocupados.

Recomendamos definir o mínimo como um valor que permita que cada gravador ou leitor de banco de dados mantenha o conjunto de trabalho da aplicação no grupo de buffer. Dessa forma, o conteúdo do grupo de buffer não é descartado durante períodos ociosos. Para saber todas as considerações ao escolher o valor da capacidade mínima, consulte Escolher a configuração de capacidade mínima do Aurora Serverless v2 para um cluster. Para saber todas as considerações ao escolher o valor da capacidade máxima, consulte Escolher a configuração de capacidade máxima do Aurora Serverless v2 para um cluster.

Dependendo de como você configura os gravadores em uma implantação multi-AZ, as respectivas capacidades podem ser vinculadas à capacidade do gravador ou de forma independente. Para obter detalhes sobre como fazer isso, consulte Aurora Serverless v2Escalabilidade do.

O monitoramento do Aurora Serverless v2 envolve medir os valores de capacidade para o gravador e os leitores em seu cluster de banco de dados ao longo do tempo. Se a escala do banco de dados não for reduzida na vertical para a capacidade mínima, você poderá realizar ações como ajustar o mínimo e otimizar sua aplicação de banco de dados. Se o banco de dados atingir consistentemente a capacidade máxima, será possível realizar ações como aumentar o máximo. Também é possível otimizar sua aplicação de banco de dados e distribuir a carga de consulta por mais leitores.

As cobranças pela capacidade do Aurora Serverless v2 são medidas em termos de ACU-horas. Para obter informações sobre como as cobranças do Aurora Serverless v2 são calculadas, consulte a página Definição de preços do Aurora.

Suponha que o número total de gravadores e leitores no cluster seja *n*. Nesse caso, o cluster consome aproximadamente *n* x *minimum ACUs* quando você não está executando nenhuma operação de banco de dados. O próprio Aurora pode realizar operações de monitoramento ou manutenção que causem uma pequena quantidade de carga. Esse cluster não consome mais do que *n* x *maximum ACUs* quando o banco de dados está sendo executado na capacidade total.

Para obter mais detalhes sobre como escolher valores mínimo e máximo apropriados de ACU, consulte escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora. Os valores mínimo e máximo de ACU especificados também afetam a forma como funcionam alguns parâmetros de configuração do Aurora para Aurora Serverless v2. Para obter detalhes sobre a interação entre o intervalo de capacidade e os parâmetros de configuração, consulte Trabalhar com grupos de parâmetros para o Aurora Serverless v2.

Aurora Serverless v2Escalabilidade do

Para cada gravador ou leitor do Aurora Serverless v2, o Aurora monitora continuamente a utilização de recursos, como CPU, memória e rede. Essas medições são coletivamente denominadas carga. A carga inclui as operações de banco de dados realizadas pela aplicação. Também inclui processamento em segundo plano para o servidor de banco de dados e tarefas administrativas do Aurora. Quando a capacidade é limitada por qualquer um desses fatores, a escala do Aurora Serverless v2 é aumentada na vertical. A escala do Aurora Serverless v2 também é aumentada na vertical quando ele detecta problemas de performance que podem ser resolvidos ao fazê-lo. Você pode monitorar a utilização de recursos e como isso afeta a escalabilidade do Aurora Serverless v2 seguindo os procedimentos em Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2 e Monitorar a performance do Aurora Serverless v2 com o Performance Insights.

A carga pode variar entre o gravador e os leitores em seu cluster de banco de dados. O gravador lida com todas as instruções DDL (Data Definition Language), como `CREATE TABLE`, `ALTER TABLE` e `DROP TABLE`. O gravador também lida com todas as instruções de linguagem de manipulação de dados (DML), como `INSERT` e `UPDATE`. Os leitores podem processar instruções somente leitura, como consultas do `SELECT`.

Escalabilidade é a operação que aumenta ou diminui a capacidade do Aurora Serverless v2 para seu banco de dados. Com o Aurora Serverless v2, cada gravador e leitor têm seu próprio valor de capacidade atual, medido em ACUs. O Aurora Serverless v2 escala um gravador ou um leitor para uma capacidade maior quando sua capacidade atual é muito baixa para lidar com a carga. Ele dimensiona o gravador ou o leitor para uma capacidade menor quando sua capacidade atual é maior do que a necessária.

Ao contrário do Aurora Serverless v1, que é escalado dobrando a capacidade sempre que o cluster de banco de dados atinge um limite, o Aurora Serverless v2 pode aumentar a capacidade incrementalmente. Quando a demanda de sua workload começa a atingir a capacidade atual do banco de dados de um gravador ou um leitor,o Aurora Serverless v2 aumenta o número de ACUs para esse gravador ou leitor. O Aurora Serverless v2 escala a capacidade nos incrementos

necessários para oferecer a melhor performance para os recursos consumidos. A escalabilidade ocorre em incrementos tão pequenos quanto 0,5 ACUs. Quanto maior a capacidade atual, maior o incremento de escalabilidade e, portanto, com maior rapidez a escalabilidade poderá acontecer.

Como a escalabilidade do Aurora Serverless v2 é tão frequente, granular e não disruptiva, ela não causa eventos pontuais no Console de gerenciamento da AWS como o Aurora Serverless v1. Em vez disso, você pode medir as métricas do Amazon CloudWatch, como `ServerlessDatabaseCapacity` e `ACUUtilization` e monitorar seus valores mínimo, máximo e médio ao longo do tempo. Para saber mais sobre as métricas do Aurora, consulte Monitorar métricas em um cluster do Amazon Aurora. Para obter dicas sobre como monitorar o Aurora Serverless v2, consulte Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2.

É possível optar por fazer uma escala do leitor ao mesmo tempo que o gravador associado ou independentemente do gravador. Faça isso especificando o nível de promoção para esse leitor.

- Leitores em níveis de promoção 0 e 1 são escalados ao mesmo tempo que o gravador. Esse comportamento de escalabilidade torna os leitores nos níveis prioritários 0 e 1 ideais para disponibilidade. O motivo disso é que eles são sempre dimensionados para a capacidade certa a fim de assumir a workload do gravador em caso de failover.
- Os leitores em níveis de promoção 2 a 15 são escalados independentemente do gravador. Cada leitor permanece dentro dos valores mínimo e máximo da ACU especificados para o cluster. Quando um leitor é escalado independentemente do banco de dados do gravador associado, ele pode ficar ocioso e diminuir a escala na vertical enquanto o gravador continua processando um alto volume de transações. Ele ainda estará disponível como destino de failover, se nenhum outro leitor estiver disponível em níveis de promoção mais baixos. No entanto, se for promovido para ser o gravador, talvez seja necessário aumentar a escala na vertical para lidar com toda a workload do gravador.

Para obter detalhes sobre os níveis de promoção, consulte Escolher o nível de promoção para um leitor do Aurora Serverless v2.

As noções de pontos de escalabilidade e tempos limite associados do Aurora Serverless v1 não se aplicam ao Aurora Serverless v2. A escalabilidade do Aurora Serverless v2 pode ocorrer enquanto as conexões de banco de dados estão abertas, as transações SQL estão em andamento, as tabelas estão bloqueadas e as tabelas temporárias estão em uso. O Aurora Serverless v2 não espera que um ponto silencioso comece a ser escalado. A escalabilidade não interrompe nenhuma operação de banco de dados em andamento.

Se sua workload exigir mais capacidade de leitura do que está disponível com um único gravador e um único leitor, você poderá adicionar vários leitores do Aurora Serverless v2 ao cluster. Cada leitor do Aurora Serverless v2 pode ser escalado dentro dos valores de capacidade mínimo e máximo especificados para o seu cluster de banco de dados. É possível usar o endpoint leitor do cluster para direcionar sessões somente leitura para os leitores e reduzir a carga no gravador.

Se o Aurora Serverless v2 realiza ou não a escalabilidade e com que rapidez ela ocorre assim que é iniciada também dependem das configurações mínima e máxima de ACU para o cluster. Além disso, depende do fato de um leitor estar ou não configurado para ser escalado junto com o gravador ou independentemente dele. Para obter detalhes sobre os fatores que afetam a escalabilidade do Aurora Serverless v2, consulte Performance e escalabilidade no Aurora Serverless v2.

Escala para zero

Nas versões recentes do Aurora MySQL e do Aurora PostgreSQL, gravadores e leitores do Aurora Serverless v2 podem ter a escala reduzida para zero ACUs. Nós nos referimos a esse recurso como pausa e retomada automáticas ou pausa automática. Você pode escolher se deseja permitir esse comportamento especificando um valor zero ou diferente de zero para a capacidade mínima. Você também pode escolher quanto tempo esperar antes que uma instância do Aurora Serverless v2 seja pausada. Para obter informações sobre quais versões têm esse recurso, consulte Aurora Serverless v2Capacidade do. Para obter informações sobre como usá-las de maneira eficaz, consulte Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2.

Nas versões mais antigas do Aurora MySQL e do Aurora PostgreSQL, gravadores e leitores do Aurora Serverless v2 ociosos podem ter a escala reduzida na vertical para o valor mínimo de ACU especificado para o cluster, mas não até zero ACUs. Nesse caso, zero ACUs não está disponível como opção quando você define a faixa de capacidade. Esse comportamento é diferente do Aurora Serverless v1, que pode pausar após um período de ociosidade, mas leva algum tempo para retomar quando você abre uma nova conexão.

Quando o cluster de banco de dados com a capacidade do Aurora Serverless v2 não é necessária por algum tempo, você também pode parar e iniciar o cluster inteiro, como acontece com clusters de banco de dados provisionados. Essa técnica é mais apropriada para sistemas de desenvolvimento e teste, onde eles talvez não sejam necessários por muitas horas seguidas, e a velocidade de retomada do cluster não é crucial. O recurso de parar/iniciar cluster está disponível para todas as versões do Aurora Serverless v2. Para obter mais informações sobre esse recurso, consulte Interromper e iniciar um cluster de banco de dados do Amazon Aurora.

Aurora Serverless v2 e alta disponibilidade

A maneira de estabelecer alta disponibilidade para um cluster de banco de dados do Aurora é torná-lo um cluster de banco de dados multi-AZ. Um cluster de banco de dados do Aurora multi-AZ tem capacidade computacional disponível em todos os momentos em mais de uma zona de disponibilidade (AZ). Essa configuração mantém seu banco de dados em funcionamento mesmo em caso de uma interrupção significativa. O Aurora realiza um failover automático em caso de um problema que afete o gravador ou até mesmo a AZ inteira. Com o Aurora Serverless v2, você pode escolher a capacidade computacional em espera para aumentar e reduzir a escala na vertical junto com a capacidade do gravador. Dessa forma, a capacidade computacional na segunda AZ está pronta para assumir a workload atual a qualquer momento. Ao mesmo tempo, a capacidade computacional em todas as AZs pode reduzir a escala na vertical quando o banco de dados fica ocioso. Para obter detalhes sobre como o Aurora funciona com o Regiões da AWS e zonas de disponibilidade, consulte Alta disponibilidade de instâncias de banco de dados do Aurora.

O recurso multi-AZ do Aurora Serverless v2 usa leitores além do gravador. O suporte para leitores é novo para o Aurora Serverless v2 em comparação com o Aurora Serverless v1. É possível adicionar até 15 leitores do Aurora Serverless v2 distribuídos por três AZs para um cluster de banco de dados do Aurora.

Para aplicações essenciais aos negócios que devem permanecer disponíveis mesmo em caso de um problema que afete todo o cluster ou o toda a região da AWS, você pode configurar um banco de dados global do Aurora. É possível usar a capacidade de Aurora Serverless v2 nos clusters secundários para que eles estejam prontos para assumir o controle durante a recuperação de desastres. Eles também podem diminuir a escala na vertical quando o banco de dados não está ocupado. Para obter detalhes sobre os bancos de dados globais do Aurora, consulte Usar o Amazon Aurora Global Database.

Aurora Serverless v2O funciona como provisionado para failover e outros recursos de alta disponibilidade. Para obter mais informações, consulte Alta disponibilidade do Amazon Aurora.

Vamos supor que você queira garantir a máxima disponibilidade do cluster do Aurora Serverless v2. Você pode criar um leitor além do gravador. Se você atribuir o leitor ao nível 0 ou 1 de promoção, qualquer escalabilidade que ocorra para o gravador também acontecerá para o leitor. Dessa forma, um leitor com capacidade idêntica está sempre pronto para assumir o controle do gravador em caso de failover.

Suponha que você queira executar relatórios trimestrais para sua empresa ao mesmo tempo que o cluster continua processando transações. Se você adicionar um leitor do Aurora Serverless v2

ao cluster e atribuí-lo a um nível de promoção de 2 a 15, poderá se conectar diretamente a esse leitor para executar os relatórios. Dependendo de quão intenso for o uso da memória e da CPU por parte das consultas de relatórios, esse leitor poderá aumentar a escala na vertical para acomodar a workload. Depois, ele pode diminuir a escala na vertical novamente quando os relatórios forem concluídos.

Aurora Serverless v2 e armazenamento

O armazenamento de cada cluster de banco de dados do Aurora consiste em seis cópias de todos os seus dados, distribuídos por três AZs. Essa replicação de dados integrada se aplica independentemente de seu cluster de banco de dados incluir leitores além do gravador. Dessa forma, seus dados estão seguros, mesmo quanto a problemas que afetem a capacidade computacional do cluster.

Aurora Serverless v2 O armazenamento do tem as mesmas características de confiabilidade e durabilidade descritas em Armazenamento do Amazon Aurora. Isso ocorre porque o armazenamento para clusters de banco de dados do Aurora funcionará da mesma forma se a capacidade de computação usar o Aurora Serverless v2 ou provisionado.

Parâmetros de configuração para clusters do Aurora

Você pode ajustar todos os mesmos parâmetros de configuração de cluster e banco de dados para clusters com capacidade de Aurora Serverless v2quanto aos clusters de banco de dados provisionados. No entanto, alguns parâmetros relacionados à capacidade são tratados de forma diferente para o Aurora Serverless v2. Em um cluster de configuração mista, os valores de parâmetros especificados para esses parâmetros relacionados à capacidade ainda se aplicam a todos os gravadores e leitores provisionados.

Quase todos os parâmetros funcionam da mesma maneira para gravadores e leitores do Aurora Serverless v2 quanto aos provisionados. As exceções são alguns parâmetros que o Aurora ajusta automaticamente durante a escalabilidade e alguns parâmetros que o Aurora mantém em valores fixos que dependem da configuração de capacidade máxima.

Por exemplo, a quantidade de memória reservada para o cache de buffer aumenta à medida que um gravador ou um leitor aumenta a escala na vertical e diminui à medida que a escala é reduzida na vertical. Dessa forma, a memória poderá ser liberada quando seu banco de dados não estiver ocupado. Por outro lado, o Aurora define automaticamente o número máximo de conexões para um valor apropriado com base na configuração de capacidade máxima. Dessa forma, as conexões

ativas não serão descartadas se a carga cair e o Aurora Serverless v2 reduzir a escala na vertical. Para obter informações sobre como o Aurora Serverless v2 lida com parâmetros específicos, consulte Trabalhar com grupos de parâmetros para o Aurora Serverless v2.

Requisitos e limitações do Aurora Serverless v2

Ao criar um cluster onde você pretende usar instâncias de banco de dados do Aurora Serverless v2, preste atenção aos requisitos e às limitações a seguir.

Tópicos

- Disponibilidade de regiões e versões
- Clusters que usam o Aurora Serverless v2 devem ter um intervalo de capacidade especificado.
- Alguns recursos provisionados não são compatíveis com o Aurora Serverless v2
- Alguns aspectos do Aurora Serverless v2 são diferentes do Aurora Serverless v1.

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS. Para obter mais informações sobre a disponibilidade de versões e regiões com o Aurora e Aurora Serverless v2, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

O exemplo a seguir mostra comandos do AWS CLI para confirmar os valores exatos do mecanismo de banco de dados que você pode usar com Aurora Serverless v2 para um Região da AWS específico. O parâmetro `--db-instance-class` para Aurora Serverless v2 é sempre `db.serverless`. O parâmetro `--engine` pode ser `aurora-mysql` ou `aurora-postgresql`. Substitua os valores `--region` e `--engine` apropriados para confirmar os valores `--engine-version` que você pode usar. Se o comando não produzir nenhuma saída, o Aurora Serverless v2 não estará disponível para essa combinação de Região da AWS e mecanismo de banco de dados.

```
aws rds describe-orderable-db-instance-options --engine aurora-mysql --db-instance-
class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text

aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.serverless \
```

```
 --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
text
```

Clusters que usam o Aurora Serverless v2 devem ter um intervalo de capacidade especificado.

Um cluster do Aurora deve ter um atributo `ServerlessV2ScalingConfiguration` para que você possa adicionar qualquer instância de banco de dados que use a classe `db.serverless` de instância de banco de dados. Esse recurso especifica o intervalo de capacidade. A capacidade do Aurora Serverless v2 varia de, no mínimo, 0 unidade de capacidade do Aurora (ACU) a, no máximo, 256 ACUs, em incrementos de 0,5 ACU. O valor mínimo permitido depende da versão do Aurora. Cada ACU fornece o equivalente a aproximadamente 2 gibibytes (GiB) da RAM e da CPU e dos trabalhos em rede associados. Para obter detalhes sobre como o Aurora Serverless v2 usa as configurações de intervalo de capacidade, consulte Como funciona o Aurora Serverless v2.

Consulte os intervalos de capacidade permitidos de várias versões do mecanismo de banco de dados em Aurora Serverless v2Capacidade do.

Você pode especificar os valores de ACU mínimo e máximo no Console de gerenciamento da AWSao criar um cluster e a instância de banco de dados do Aurora Serverless v2 associada. Também é possível especificar a opção `--serverless-v2-scaling-configuration` no AWS CLI. Também é possível especificar o parâmetro `ServerlessV2ScalingConfiguration` com a API do Amazon RDS. Você pode especificar esse atributo ao criar um cluster ou modificar um existente. Para saber os procedimentos que definem o intervalo de capacidade, consulte Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster. Para ler uma discussão detalhada sobre como escolher valores de capacidade mínima e máxima e como essas configurações afetam alguns parâmetros do banco de dados, consulte escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora.

Alguns recursos provisionados não são compatíveis com o Aurora Serverless v2

Os seguintes recursos das instâncias de banco de dados provisionadas do Aurora atualmente não estão disponíveis para o Amazon Aurora Serverless v2:

- Fluxos de atividades do banco de dados (DAS).
- Gerenciamento de cache do cluster do Aurora PostgreSQL. O parâmetro de configuração `apg_ccm_enabled` não se aplica a instâncias de banco de dados do Aurora Serverless v2.

Alguns recursos do Aurora funcionam com Aurora Serverless v2, mas poderão causar problemas se o intervalo de capacidade for menor do que o necessário para os requisitos de memória desses recursos com sua workload específica. Nesse caso, seu banco de dados pode não funcionar tão bem quanto de costume ou encontrar erros de falta de memória. Para obter recomendações sobre como definir o intervalo de capacidade apropriado, consulte escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora. Para obter informações sobre solução de problemas se o banco de dados encontrar erros de falta de memória devido a um intervalo de capacidade configurado incorretamente, consulte Evitar erros de falta de memória.

O Auto Scaling do Aurora não é compatível. Esse é o tipo de escalabilidade que adiciona novos leitores para lidar com a workload de uso intenso de leitura, com base no uso da CPU. No entanto, a escalabilidade com base no uso da CPU não é significativa para o Aurora Serverless v2. Como alternativa, você pode criar instâncias de banco de dados do leitor do Aurora Serverless v2 com antecedência e deixá-las com a escala reduzida na vertical para baixa capacidade. Essa é uma maneira mais rápida e menos disruptiva de escalar a capacidade de leitura de um cluster em comparação com a inclusão de novas instâncias de banco de dados de modo dinâmico.

Alguns aspectos do Aurora Serverless v2 são diferentes do Aurora Serverless v1.

Se você for usuário do Aurora Serverless v1 e for a primeira vez que você usa o Aurora Serverless v2, consulte Diferenças entre os requisitos do Aurora Serverless v2 e do Aurora Serverless v1 para compreender como os requisitos são diferentes entre o Aurora Serverless v1 e o Aurora Serverless v2.

Criar um cluster de banco de dados que usa o Aurora Serverless v2

Para criar um cluster do Aurora ao qual você possa adicionar instâncias de banco de dados do Aurora Serverless v2, siga o mesmo procedimento usado em Criar um cluster de bancos de dados do Amazon Aurora. Com o Aurora Serverless v2, seus clusters são intercambiáveis com clusters provisionados. É possível ter clusters em que algumas instâncias de banco de dados usam o Aurora Serverless v2 e algumas são provisionadas.

Tópicos

- Configurações de clusters de banco de dados do Aurora Serverless v2
- Criar um cluster de banco de dados do Aurora Serverless v2

- Criar uma instância de banco de dados de gravador do Aurora Serverless v2

Configurações de clusters de banco de dados do Aurora Serverless v2

As configurações iniciais do cluster devem atender aos requisitos listados em Requisitos e limitações do Aurora Serverless v2. Especifique as seguintes configurações para garantir que você possa adicionar instâncias de banco de dados do Aurora Serverless v2 ao cluster:

Região da AWS

Criar um cluster em uma Região da AWS onde instâncias de banco de dados do Aurora Serverless v2 estejam disponíveis. Para obter detalhes sobre as regiões disponíveis, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

DB engine version (Versão do mecanismo de banco de dados)

Escolha uma versão do mecanismo compatível com o Aurora Serverless v2. Para obter informações sobre os requisitos da versão Aurora Serverless v2, consulte Requisitos e limitações do Aurora Serverless v2.

Classe de instância de banco de dados

Se você criar um cluster usando o Console de gerenciamento da AWS, escolha a classe para a instância de banco de dados de gravador simultaneamente. Escolha a classe de instância de banco de dados Serverless (Sem servidor). Ao escolher essa classe de instância de banco de dados, especifique também o intervalo de capacidade para a instância de banco de dados do gravador. Esse mesmo intervalo de capacidade se aplica a todas as outras instâncias de banco de dados do Aurora Serverless v2 adicionadas a esse cluster.

Se você não vir a opção Tecnologia sem servidor para a classe de instância de banco de dados, verifique se você escolheu uma versão do mecanismo de banco de dados compatível com Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

Quando você usa a AWS CLI ou a API do Amazon RDS, o parâmetro especificado para a classe de instância de banco de dados é `db.serverless`.

Intervalo de capacidade

Preencha os valores mínimo e máximo de unidade de capacidade do Aurora (ACU) que se aplicam a todas as instâncias de banco de dados no cluster. Essa opção está disponível nas páginas Create cluster (Criar cluster) e Add reader (Adicionar leitor) do console quando você escolhe Serverless (Sem servidor) para a classe da instância de banco de dados.

Consulte os intervalos de capacidade permitidos de várias versões do mecanismo de banco de dados em Aurora Serverless v2Capacidade do.

Caso não veja as caixas de ACU mínima e máxima, verifique se você escolheu a classe Tecnologia sem servidor da instância de banco de dados do gravador.

Se você inicialmente criar o cluster com uma instância de banco de dados provisionada, não especificará as ACUs mínima e máxima. Nesse caso, você pode modificar o cluster posteriormente para adicionar essa configuração. Também é possível adicionar uma instância de banco de dados do leitor do Aurora Serverless v2 ao cluster. É necessário especificar o intervalo de capacidade como parte desse processo.

Você só poderá adicionar instâncias de banco de dados do Aurora Serverless v2 ao cluster usando o AWS CLI ou a API do RDS depois de especificar o intervalo de capacidade para o cluster. Se tentar adicionar uma instância de banco de dados do Aurora Serverless v2, será gerado um erro. No AWS CLI ou nos procedimentos da API do RDS, o intervalo de capacidade é representado pelo atributo `ServerlessV2ScalingConfiguration`.

No caso de clusters que contenham mais de uma instância de banco de dados do leitor, a prioridade de failover de cada instância de banco de dados do leitor do Aurora Serverless v2 desempenha um papel importante na forma como a escala dessa instância é aumentada ou reduzida na vertical. Não é possível especificar a prioridade ao criar inicialmente o cluster. Tenha em mente essa propriedade ao adicionar uma segunda instância de banco de dados do leitor ao cluster ou fazê-lo posteriormente. Para ter mais informações, consulte Escolher o nível de promoção para um leitor do Aurora Serverless v2.

Criar um cluster de banco de dados do Aurora Serverless v2

É possível usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para criar um cluster de banco de dados do Aurora Serverless v2.

Console

Como criar um cluster com um gravador Aurora Serverless v2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.

3. Selecione Criar banco de dados. Na página exibida, escolha as seguintes opções:

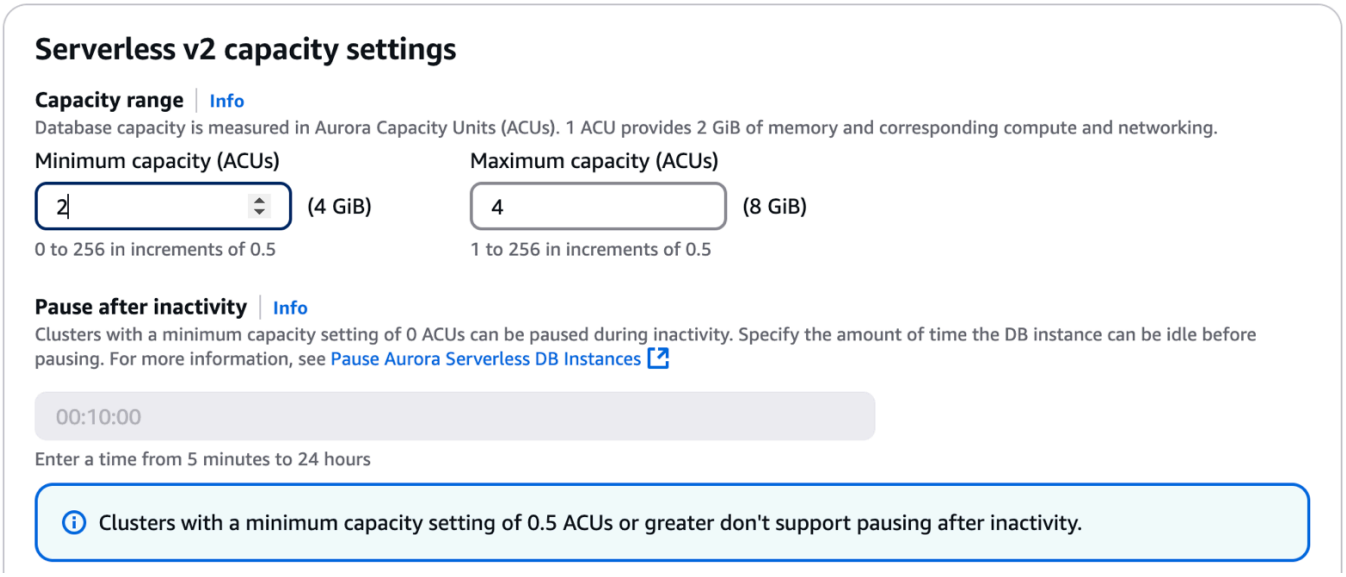
   - Em Tipo de mecanismo, escolha Aurora (compatível com MySQL) ou Aurora (compatível com PostgreSQL).
   - Em Versão, escolha uma das versões compatíveis para Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

4. Em Classe da instância de banco de dados, escolha Serverless v2.

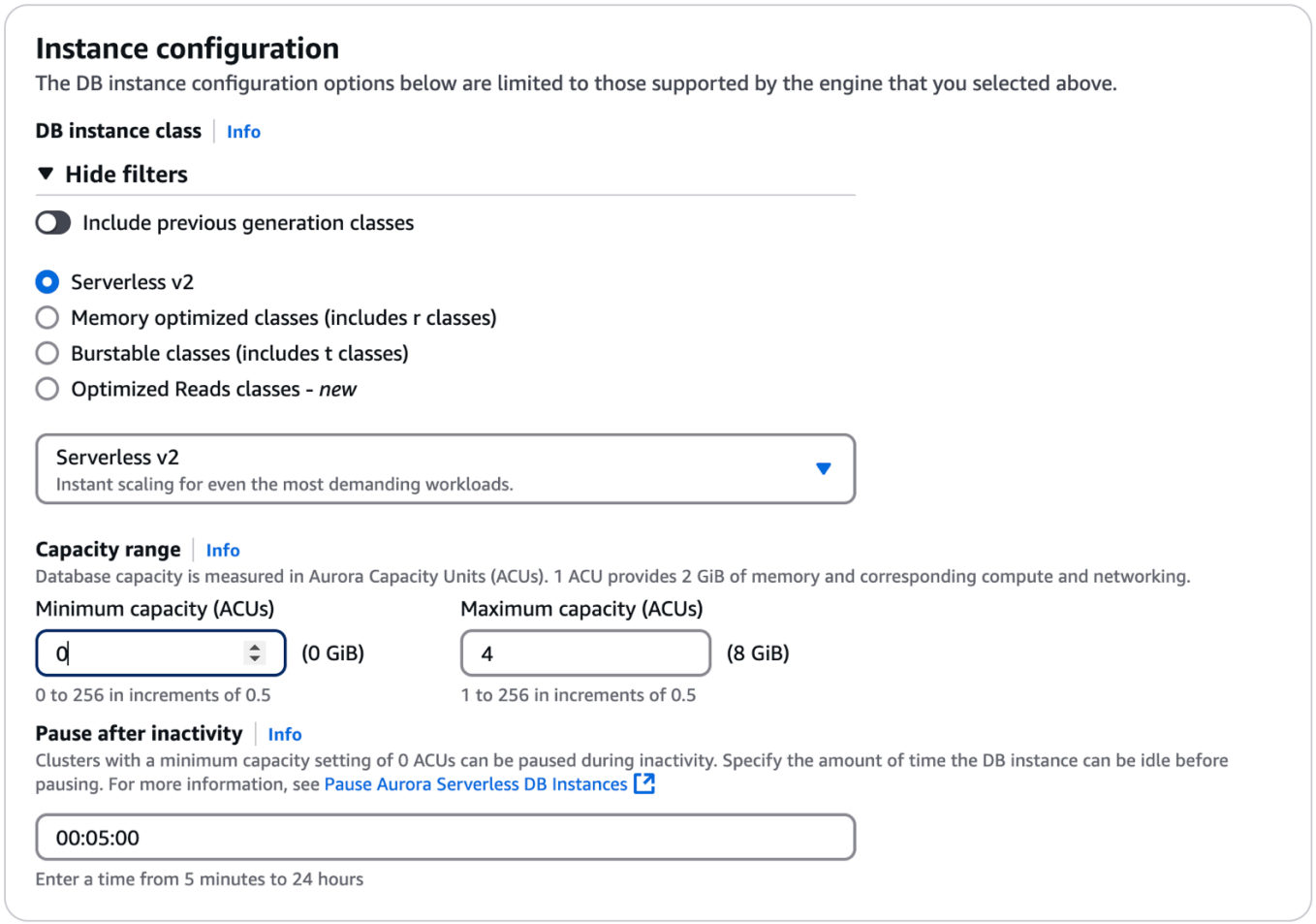
5. Em Intervalo de capacidade, você pode aceitar o intervalo padrão. Ou você pode escolher outros valores para unidades de capacidade mínima e máxima. É possível escolher entre 0 ACU, no mínimo, e 256 ACUs, no máximo, em incrementos de 0,5 ACU.

   Para obter mais informações sobre unidades de capacidade do Aurora Serverless v2, consulte Aurora Serverless v2Capacidade do e Performance e escalabilidade no Aurora Serverless v2.

   Dependendo do mecanismo e da versão que você escolher, o limite máximo poderá ser 128 ACUs, o limite mínimo poderá ser 0,5 ACUs ou ambos. Para obter detalhes sobre o limite para cada combinação de mecanismo e versão do Aurora, consulte Aurora Serverless v2Capacidade do.



   Escolher uma capacidade mínima de 0 ACUs habilita o recurso de pausa e retomada automáticas do Aurora Serverless v2. Nesse caso, você pode fazer uma escolha adicional de quanto tempo as instâncias de banco de dados do Aurora Serverless v2 esperam sem conexões de banco de dados antes de pausar automaticamente. Para obter informações sobre o recurso de pausa e retomada automáticas, consulte Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2.



6. Escolha qualquer outra configuração de cluster de banco de dados, conforme descrito em Configurações de clusters de bancos de dados do Aurora.
7. Escolha Criar banco de dados para criar o cluster do Aurora com um instância de banco de dados do Aurora Serverless v2 como a instância do gravador, também conhecida como instância do banco de dados primário.

CLI

Para criar um cluster de banco de dados compatível com instâncias de banco de dados do Aurora Serverless v2 usando o AWS CLI, siga o procedimento da CLI em Criar um cluster de bancos de dados do Amazon Aurora. Inclua estes parâmetros em seu comando `create-db-cluster`:

- --region *AWS_Region_where_Aurora Serverless v2_instances_are_available*
- --engine-version *serverless_v2_compatible_engine_version*

- --serverless-v2-scaling-configuration MinCapacity=*minimum_capacity*,MaxCapacity=*maximum_capacity*

O exemplo a seguir mostra a criação de um cluster de banco de dados do Aurora Serverless v2.

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.04.1 \
    --serverless-v2-scaling-configuration MinCapacity=1,MaxCapacity=4 \
    --master-username myuser \
    --manage-master-user-password
```

> **Note**
>
> Ao criar um cluster de banco de dados do Aurora Serverless v2 usando a AWS CLI, o modo do mecanismo aparece na saída como `provisioned` em vez de `serverless`. O modo de mecanismo `serverless` refere-se a Aurora Serverless v1.

Esse exemplo especifica a opção `--manage-master-user-password` para gerar a senha administrativa e gerenciá-la no Secrets Manager. Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager. Como alternativa, você pode usar a opção `--master-password` para especificar e gerenciar a senha por conta própria.

Para obter informações sobre os requisitos da versão Aurora Serverless v2, consulte Requisitos e limitações do Aurora Serverless v2. Para obter informações sobre os números permitidos para o intervalo de capacidade e o que esses números representam, consulte Aurora Serverless v2Capacidade do e Performance e escalabilidade no Aurora Serverless v2.

Para verificar se um cluster existente tem as configurações de capacidade especificadas, confira a saída do comando `describe-db-clusters` para atributo `ServerlessV2ScalingConfiguration`. Esse atributo é semelhante ao seguinte.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se você não especificar as ACUs mínima e máxima ao criar o cluster, poderá usar o comando `modify-db-cluster` depois para adicionar essa configuração. Até esse ponto, você não poderá adicionar nenhuma instância de banco de dados do Aurora Serverless v2 ao cluster. Se tentar adicionar uma instância de banco de dados do `db.serverless`, será gerado um erro.

API

Para criar um cluster de banco de dados compatível com instâncias de banco de dados do Aurora Serverless v2 usando a API do RDS, siga o procedimento da CLI em Criar um cluster de bancos de dados do Amazon Aurora. Escolha as configurações a seguir. Verifique se a sua operação `CreateDBCluster` inclui os seguintes parâmetros:

```
EngineVersion serverless_v2_compatible_engine_version
ServerlessV2ScalingConfiguration with MinCapacity=minimum_capacity and
 MaxCapacity=maximum_capacity
```

Para obter informações sobre os requisitos da versão Aurora Serverless v2, consulte Requisitos e limitações do Aurora Serverless v2. Para obter informações sobre os números permitidos para o intervalo de capacidade e o que esses números representam, consulte Aurora Serverless v2Capacidade do e Performance e escalabilidade no Aurora Serverless v2.

Para conferir se um cluster existente tem as configurações de capacidade especificadas, confira a saída da operação `DescribeDBClusters` para o atributo `ServerlessV2ScalingConfiguration`. Esse atributo é semelhante ao seguinte.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se você não especificar as ACUs mínima e máxima ao criar o cluster, poderá usar a operação `ModifyDBCluster` depois para adicionar essa configuração. Até esse ponto,

você não poderá adicionar nenhuma instância de banco de dados do Aurora Serverless v2 ao cluster. Se tentar adicionar uma instância de banco de dados do `db.serverless`, será gerado um erro.

Criar uma instância de banco de dados de gravador do Aurora Serverless v2

Embora você especifique o intervalo de capacidade do Aurora Serverless v2 ao criar um cluster do Aurora, é possível escolher se deseja usar o Aurora Serverless v2 ou provisionado para cada instância de banco de dados no cluster. Para começar a usar o Aurora Serverless v2 imediatamente em seu cluster de banco de dados, adicione uma instância de banco de dados do gravador que use a classe de instância `db.serverless`. No console, você normalmente faz essa escolha como parte do fluxo de trabalho para criar o cluster de banco de dados. Portanto, esse procedimento se aplica principalmente quando você faz a configuração por meio da CLI.

Console

Ao criar um cluster de banco de dados usando o Console de gerenciamento da AWS, você deve especificar as propriedades da instância de banco de dados do gravador simultaneamente. Para que a instância de banco de dados do gravador use o Aurora Serverless v2, escolha a classe de instância de banco de dados Serverless(Sem servidor).

Depois, defina o intervalo de capacidade para o cluster especificando os valores mínimo e máximo de unidade de capacidade do Aurora (ACU). Os mesmos valores mínimo e máximo se aplicam a todas as instâncias de banco de dados do Aurora Serverless v2 no cluster. Para obter esse procedimento e a importância das configurações de capacidade mínima e máxima, consulte Criar um cluster de banco de dados do Aurora Serverless v2.

Se você não criar uma instância de banco de dados do Aurora Serverless v2 ao criar o cluster, poderá adicionar uma ou mais instâncias de banco de dados do Aurora Serverless v2 posteriormente Para isso, siga os procedimentos em Adicionar um leitor do Aurora Serverless v2 e Converter um gravador ou leitor provisionado em Aurora Serverless v2. Especifique o intervalo de capacidade ao adicionar a primeira instância de banco de dados do Aurora Serverless v2 ao cluster. Você pode alterar o intervalo de capacidade mais tarde seguindo o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

CLI

Ao criar um cluster de banco de dados do Aurora Serverless v2 usando a AWS CLI, adicione explicitamente a instância de banco de dados do gravador usando o comando create-db-instance. Inclua o seguinte parâmetro:

- `--db-instance-class db.serverless`

O exemplo a seguir mostra a criação de uma instância de banco de dados de gravador do Aurora Serverless v2.

```
aws rds create-db-instance \
    --db-cluster-identifier my-serverless-v2-cluster \
    --db-instance-identifier my-serverless-v2-instance \
    --db-instance-class db.serverless \
    --engine aurora-mysql
```

Gerenciar clusters de banco de dados do Aurora Serverless v2

Com o Aurora Serverless v2, seus clusters são intercambiáveis com clusters provisionados. As propriedades do Aurora Serverless v2 aplicam-se a uma ou mais instâncias de banco de dados em um cluster de banco de dados. Dessa forma, os procedimentos para criar e modificar clusters, criar e restaurar snapshots etc. são basicamente os mesmos que para outros tipos de cluster do Aurora. Com relação a procedimentos gerais para gerenciar clusters e instâncias de banco de dados do Aurora, consulte Como gerenciar um cluster de banco de dados do Amazon Aurora.

Nos tópicos a seguir, saiba mais sobre considerações de gerenciamento para clusters que contêm instâncias de banco de dados do Aurora Serverless v2.

Tópicos

- Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster
- Conferir o intervalo de capacidade do Aurora Serverless v2
- Adicionar um leitor do Aurora Serverless v2
- Converter um gravador ou leitor provisionado em Aurora Serverless v2
- Converter um gravador ou leitor do Aurora Serverless v2 em provisionado
- Pausar instâncias de banco de dados do Aurora Serverless v2

- Escolher o nível de promoção para um leitor do Aurora Serverless v2
- Usar TLS/SSL com o Aurora Serverless v2
- Visualizar gravadores e leitores do Aurora Serverless v2
- Registro em log para o Aurora Serverless v2

Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster

Para modificar parâmetros de configuração ou outras configurações para clusters que contenham instâncias de banco de dados do Aurora Serverless v2 ou as próprias instâncias de banco de dados, siga os mesmos procedimentos gerais para os clusters provisionados. Para obter detalhes, consulte Modificar um cluster de bancos de dados Amazon Aurora.

A configuração mais importante exclusiva do Aurora Serverless v2 é o intervalo de capacidade. Depois de definir os valores mínimo e máximo da unidade de capacidade do Aurora (ACU) para um cluster do Aurora, você não precisa ajustar ativamente a capacidade das instâncias de banco de dados do Aurora Serverless v2 no cluster. O Aurora faz isso por você. Essa configuração é gerenciada no nível do cluster. Os mesmos valores mínimo e máximo da ACU se aplicam a cada instância de banco de dados do Aurora Serverless v2 no cluster.

Você pode definir os seguintes valores específicos:

- Minimum ACUs (Mínimo de ACUs): a instância de banco de dados do Aurora Serverless v2 pode reduzir a capacidade até esse número de ACUs.
- Maximum ACUs (Máximo de ACUs): a instância de banco de dados do Aurora Serverless v2 pode aumentar a capacidade até esse número de ACUs.

O intervalo de capacidade disponível para o Aurora Serverless v2 depende tanto da versão do mecanismo de banco de dados quanto da versão da plataforma. As versões mais recentes do mecanismo de banco de dados permitem uma capacidade máxima de 256 ACUs, uma capacidade mínima de 0 ACUs ou ambas. Consulte os intervalos de capacidade de várias versões do mecanismo de banco de dados em Aurora Serverless v2Capacidade do.

Consulte Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2 para saber mais sobre o recurso de pausa e retomada automáticas que é habilitado ao definir a capacidade mínima para 0 ACUs.

Consulte mais informações sobre como garantir que os clusters de banco de dados do Aurora Serverless v2 possam ser escalados até 256 ACUs em Atualização do cluster de banco de dados do Aurora Serverless v2 para permitir escalar até 256 ACUs.

> **Note**
>
> Quando você modifica o intervalo de capacidade de um cluster de banco de dados do Aurora Serverless v2, a alteração ocorre imediatamente, independentemente de você optar por aplicá-la imediatamente ou durante a próxima janela de manutenção programada.

Em Aurora Serverless v2, não é possível definir diretamente a capacidade atual sem modificar o intervalo de capacidade, pois a capacidade se ajusta dinamicamente com base na workload dentro do intervalo especificado. No entanto, é possível influenciar a capacidade indiretamente das seguintes maneiras:

- Ajustar a capacidade mínima: reduza temporariamente a capacidade mínima para reduzir a capacidade básica quando as workloads forem leves.
- Pausar a escalabilidade temporariamente: para fixar a capacidade em um valor específico, defina a capacidade mínima e máxima no mesmo nível.
- Escale proativamente para uso em picos: se você prevê workloads com picos, aumente a capacidade mínima com antecedência para manter uma linha de base mais alta durante períodos de alta atividade.

Para obter detalhes sobre os efeitos do intervalo de capacidade e como monitorá-lo e ajustá-lo, consulte Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2 e Performance e escalabilidade no Aurora Serverless v2. Seu objetivo é garantir que a capacidade máxima do cluster seja alta o suficiente para lidar com picos da workload, e o mínimo seja baixo o suficiente para minimizar os custos quando o cluster não estiver ocupado.

Suponha que você determine com base no monitoramento que o intervalo de ACU para o cluster deva ser maior, menor, mais amplo ou mais estreito. É possível definir a capacidade de um cluster do Aurora como um intervalo específico de ACUs com o Console de gerenciamento da AWS, o AWS CLI ou a API do Amazon RDS. Esse intervalo de capacidade se aplica a todas as instâncias de banco de dados do Aurora Serverless v2 no cluster.

Por exemplo, suponha que seu cluster tenha um intervalo de capacidade de 1 a 16 ACUs e contenha duas instâncias de banco de dados do Aurora Serverless v2. Depois, o cluster como um todo

consome entre duas ACUs (quando ocioso) e 32 ACUs (quando totalmente utilizado). Se você alterar o intervalo de capacidade de 8 para 40,5 ACUs, o cluster inteiro consumirá 16 ACUs quando ocioso e até 81 ACUs quando totalmente utilizado.

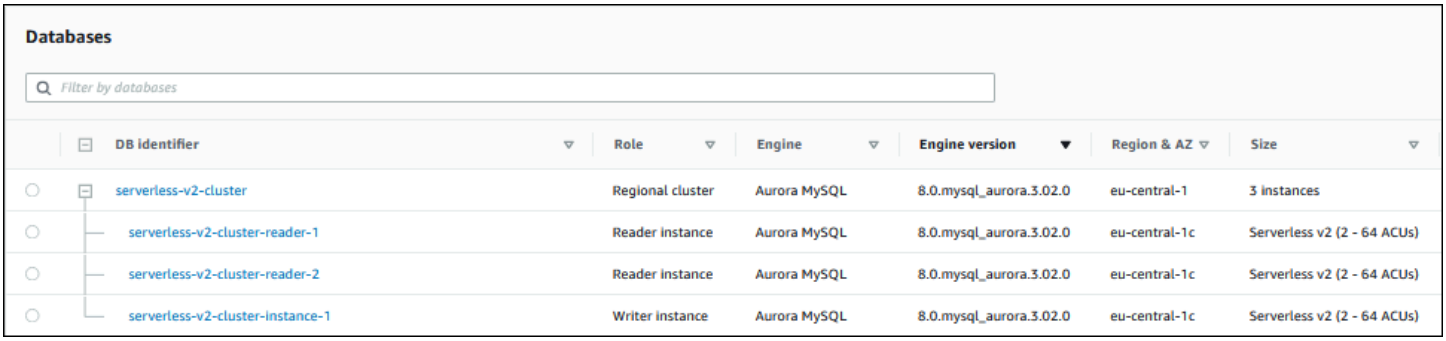
O Aurora define automaticamente determinados parâmetros para instâncias de banco de dados do Aurora Serverless v2 para valores que dependem do valor máximo de ACU no intervalo de capacidade. Para obter uma lista desses parâmetros, consulte Conexões máximas do Aurora Serverless v2. No caso de parâmetros estáticos que dependem desse tipo de cálculo, o valor é avaliado novamente quando você reinicializa a instância de banco de dados. Assim, é possível atualizar o valor desses parâmetros reinicializando a instância de banco de dados depois de alterar o intervalo de capacidade. Para conferir se você precisa reinicializar sua instância de banco de dados para obter essas alterações de parâmetros, confira o atributo `ParameterApplyStatus` da instância de banco de dados. Um valor de `pending-reboot`indica que a reinicialização aplicará alterações a alguns valores de parâmetros.

Console

Você pode definir o intervalo de capacidade de um cluster que contém instâncias de banco de dados do Aurora Serverless v2 com o Console de gerenciamento da AWS.

Ao usar o console, defina o intervalo de capacidade para o cluster ao adicionar a primeira instância de banco de dados do Aurora Serverless v2 a esse cluster. Você pode fazer isso ao escolher a classe de instância de banco de dados Serverless v2 (Sem servidor v2) para a instância de banco de dados do gravador ao criar o cluster. Ou você pode fazer isso ao escolher a classe de instância de banco de dados Serverless (Sem servidor) ao adicionar uma instância de banco de dados do leitor do Aurora Serverless v2 ao cluster. Ou ao converter uma instância de banco de dados provisionada existente no cluster à classe Serverless (Sem servidor). Para ver as versões completas desses procedimentos, consulte Criar uma instância de banco de dados de gravador do Aurora Serverless v2, Adicionar um leitor do Aurora Serverless v2 e Converter um gravador ou leitor provisionado em Aurora Serverless v2.

Qualquer intervalo de capacidade que você definir no nível do cluster se aplicará a todas as instâncias de banco de dados do Aurora Serverless v2 em seu cluster. A imagem a seguir mostra um cluster com várias instâncias de banco de dados do leitor do Aurora Serverless v2. Cada uma tem um intervalo de capacidade idêntico de duas a 64 ACUs.

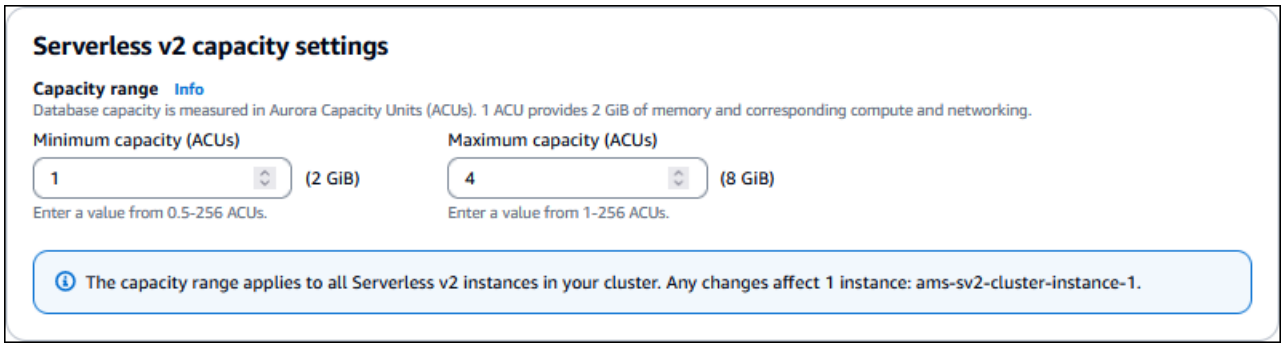


Como modificar o intervalo de capacidade de um cluster do Aurora Serverless v2

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster que contém suas instâncias de banco de dados do Aurora Serverless v2 na lista. O cluster já deve conter pelo menos uma instância de banco de dados do Aurora Serverless v2. Caso contrário, o Aurora não mostrará a seção Capacity range (Intervalo de capacidade).
4. Para Actions (Ações), escolha Modify (Modificar).
5. Na seção Capacity range (Intervalo de capacidade), escolha o seguinte:

   a. Insira um valor para Minimum ACUs (Mínimo de ACUs). O console mostra o intervalo de valores permitido. É possível escolher uma capacidade mínima de 0 a 256 ACUs. É possível escolher uma capacidade máxima de 1 a 256 ACUs. É possível ajustar os valores de capacidade em incrementos de 0,5 ACU. O intervalo de capacidade disponível depende tanto da versão do mecanismo de banco de dados quanto da versão da plataforma.

   b. Insira um valor para Maximum ACUs (Máximo de ACUs). Esse valor deve ser maior que ou igual a Minimum ACUs (Mínimo de ACUs). O console mostra o intervalo de valores permitido. A figura a seguir mostra essa opção.



6. Escolha Continue. A página Resumo das modificações é exibida.

7. Escolha Apply immediately (Aplicar imediatamente).

   A modificação de capacidade ocorre imediatamente, independentemente de você optar por aplicá-la imediatamente ou durante a próxima janela de manutenção programada.

8. Escolha Modify cluster (Modificar cluster) para aceitar o resumo das modificações. Você também pode escolher BAck (Voltar) para modificar suas alterações ou Cancel (Cancelar) para descartar suas alterações.

AWS CLI

Para definir a capacidade de um cluster no qual você pretenda usar instâncias de banco de dados do Aurora Serverless v2 usando o AWS CLI, execute o comando modify-db-cluster do AWS CLI. Especifique a opção `--serverless-v2-scaling-configuration`. Talvez o cluster já contenha uma ou mais instâncias de banco de dados do Aurora Serverless v2 ou você pode adicioná-las posteriormente. Valores válidos para os campos `MinCapacity` e `MaxCapacity` incluem os seguintes:

- `0`, `0.5`, `1`, `1.5`, 2 etc., em etapas de 0,5 até no máximo 256. O valor mínimo de ACU como 0 só está disponível nas versões do Aurora compatíveis com o recurso de pausa automática.

A capacidade máxima disponível depende tanto da versão do mecanismo de banco de dados quanto da versão da plataforma.

Neste exemplo, você define o intervalo de ACU de um cluster de banco de dados do Aurora chamado `sample-cluster` como no mínimo `48.5` e no máximo 64.

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
  --serverless-v2-scaling-configuration MinCapacity=48.5,MaxCapacity=64
```

A modificação de capacidade ocorre imediatamente, independentemente de você optar por aplicá-la imediatamente ou durante a próxima janela de manutenção programada.

Depois disso, você pode adicionar instâncias de banco de dados do Aurora Serverless v2 ao cluster, e cada nova instância de banco de dados pode ser escalada entre 48,5 e 64 ACUs. O novo intervalo de capacidade também se aplica a todas as instâncias de banco de dados do Aurora Serverless v2 já presentes no cluster. A escala das instâncias de banco de dados é aumentada ou reduzida na vertical para se inserir no novo intervalo de capacidade.

Para obter exemplos adicionais de configuração do intervalo de capacidade usando a CLI, consulte escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora.

Para modificar a configuração de escalabilidade do cluster de banco de dados do Aurora Serverless usando a AWS CLI, execute o comando modify-db-cluster da AWS CLI. Especifique a opção `--serverless-v2-scaling-configuration` para configurar a capacidade mínima e máxima. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: `0`, `0.5`, `1`, `1.5`, 2 etc., em incrementos de 0,5 ACUs até no máximo 256.
- Aurora PostgreSQL: `0`, `0.5`, `1`, `1.5`, 2 etc., em incrementos de 0,5 ACUs até no máximo 256.
- O valor mínimo de ACU como 0 só está disponível nas versões do Aurora compatíveis com o recurso de pausa automática.

No exemplo a seguir, você modificará a configuração de escalabilidade de uma instância de banco de dados do Aurora Serverless v2 denominada `sample-instance` que faz parte de um cluster chamado `sample-cluster`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

Para Windows:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster ^
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

API do RDS

É possível definir a capacidade de uma instância de bancos de dados do Aurora com a operação da API ModifyDBCluster. Especifique o parâmetro `ServerlessV2ScalingConfiguration`. Valores válidos para os campos `MinCapacity` e `MaxCapacity` incluem os seguintes:

- `0`, `0.5`, `1`, `1.5`, 2 etc., em etapas de 0,5 até no máximo 256. O valor mínimo de ACU como 0 só está disponível nas versões do Aurora compatíveis com o recurso de pausa automática.

É possível modificar a configuração de escalabilidade de um cluster que contém instâncias de banco de dados do Aurora Serverless v2 com a operação da API ModifyDBCluster. Especifique o parâmetro

`ServerlessV2ScalingConfiguration` para configurar a capacidade mínima e máxima. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: `0`, `0.5`, `1`, `1.5`, 2 etc., em incrementos de 0,5 ACUs até no máximo 256.
- Aurora PostgreSQL: `0`, `0.5`, `1`, `1.5`, 2 etc., em incrementos de 0,5 ACUs até no máximo 256.
- O valor mínimo de ACU como 0 só está disponível nas versões do Aurora compatíveis com o recurso de pausa automática.

A capacidade máxima disponível depende tanto da versão do mecanismo de banco de dados quanto da versão da plataforma.

A modificação de capacidade ocorre imediatamente, independentemente de você optar por aplicá-la imediatamente ou durante a próxima janela de manutenção programada.

Atualização do cluster de banco de dados do Aurora Serverless v2 para permitir escalar até 256 ACUs

Em alguns casos, quando você tenta escalar o cluster de banco de dados do Aurora Serverless v2 para capacidades acima de 128 ACUs, você recebe uma mensagem de erro. A mensagem de erro informa quais instâncias são incompatíveis com o novo intervalo de escalabilidade. Isso evidencia a relação fundamental entre o intervalo de capacidade, a versão do mecanismo de banco de dados e a versão da plataforma.

A impossibilidade de escalar para mais de 128 ACUs pode ocorrer por um dos dois motivos:

- Versão mais antiga do mecanismo de banco de dados: atualize a versão do mecanismo de banco de dados para uma que comporte 256 ACUs. Para obter mais informações, consulte Aurora Serverless v2Capacidade do.
- Versão mais antiga da plataforma: atualize a plataforma subjacente para o cluster de banco de dados do Aurora Serverless v2. Você pode fazer isso por meio de uma das seguintes maneiras:
  - Interrompa e reinicie o cluster de banco de dados. Quando o cluster for reiniciado, ele estará na versão da plataforma mais recente e capaz, que possa ter um valor máximo de ACU mais alto.

    No entanto, interromper e iniciar um cluster de banco de dados gera tempo de inatividade, geralmente de vários minutos. Para obter mais informações, consulte Interromper e iniciar um cluster de banco de dados do Amazon Aurora.
  - Use uma implantação azul/verde. Para obter mais informações, consulte Visão geral das implantações azul/verde do Amazon Aurora.

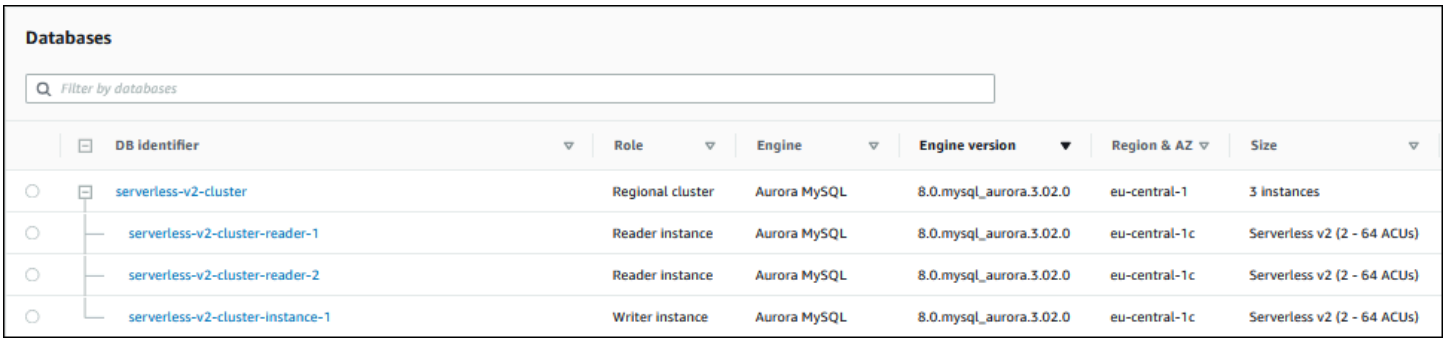
1. Crie uma implantação azul/verde. Para obter mais informações, consulte Criar uma implantação azul/verde no Amazon Aurora.
2. Confirme se você pode definir a capacidade máxima do ambiente de preparação (verde) para 256 ACUs.
3. Mude para o ambiente verde. Para obter mais informações, consulte Mudar uma implantação azul/verde no Amazon Aurora.
4. Exclua o cluster de banco de dados original.

> **Note**
>
> Se você mantiver as credenciais de banco de dados no AWS Secrets Manager, não poderá usar implantações azul/verde.
> O banco de dados global do Aurora não permite implantações azul/verde.

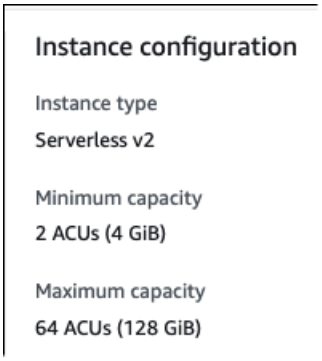
Conferir o intervalo de capacidade do Aurora Serverless v2

O procedimento para conferir o intervalo de capacidade do seu cluster do Aurora Serverless v2 exige primeiro definir um intervalo de capacidade. Caso ainda não tenha feito isso, siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

Qualquer intervalo de capacidade definido no nível do cluster se aplicará a todas as instâncias de banco de dados do Aurora Serverless v2 em seu cluster. A imagem a seguir mostra um cluster com várias instâncias de banco de dados do Aurora Serverless v2. Cada uma tem um intervalo de capacidade idêntico.



Você também pode visualizar a página de detalhes de qualquer instância de banco de dados do Aurora Serverless v2 no cluster. O intervalo de capacidade das instâncias de banco de dados é exibido na guia Configuration (Configuração).



Você também pode ver o intervalo de capacidade atual do cluster na página Modify (Modificar) do cluster. Nesse ponto, você pode alterar o intervalo de capacidade. Para saber todas as maneiras de definir ou alterar o intervalo de capacidade, consulte Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

Conferir o intervalo de capacidade atual de um cluster do Aurora

É possível conferir o intervalo de capacidade configurado para instâncias de banco de dados do Aurora Serverless v2 em um cluster examinando o atributo `ServerlessV2ScalingConfiguration` para o cluster. O exemplo de AWS CLI a seguir mostra um cluster com uma capacidade mínima de 0,5 unidades de capacidade do Aurora (ACUs) e uma capacidade máxima de 16 ACUs.

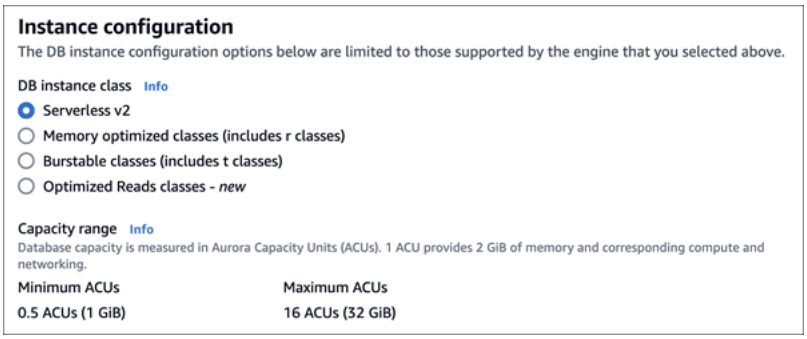
```
$ aws rds describe-db-clusters --db-cluster-identifier serverless-v2-64-acu-cluster \
  --query 'DBClusters[*].[ServerlessV2ScalingConfiguration]'
[
    [
        {
            "MinCapacity": 0.5,
            "MaxCapacity": 16.0
        }
    ]
]
```

Adicionar um leitor do Aurora Serverless v2

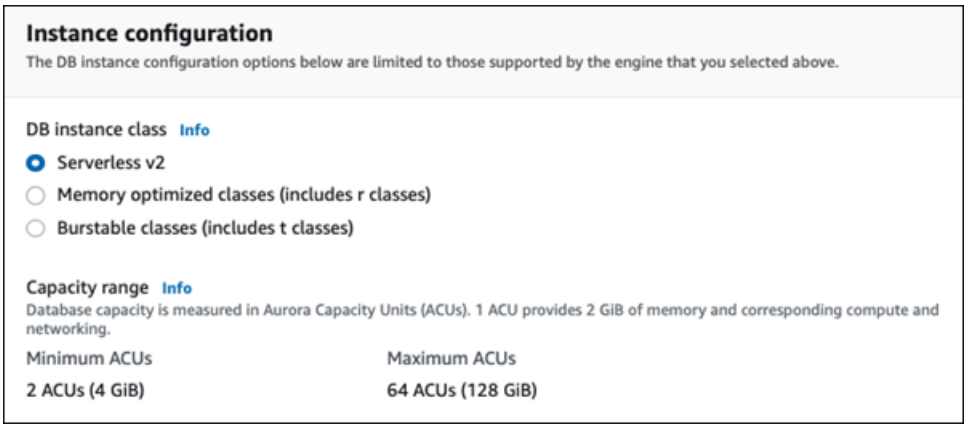
Para adicionar uma instância de banco de dados do leitor do Aurora Serverless v2 ao cluster, siga o mesmo procedimento geral que em Adicionar réplicas do Aurora a um cluster de banco de dados. Escolha a classe de instância Serverless v2 para a nova instância de banco de dados.

Se a instância de banco de dados do leitor for a primeira instância de banco de dados do Aurora Serverless v2 no cluster, escolha também o intervalo de capacidade. Essa configuração se aplica a essa instância de banco de dados do leitor e a todas as outras instâncias de banco de dados do Aurora Serverless v2 adicionadas ao cluster. Cada instância de banco de dados do Aurora Serverless v2 pode ser escalada entre os valores mínimo e máximo de ACUs.

Se já existirem outras instâncias do Aurora Serverless v2 no cluster, a caixa de diálogo Adicionar leitor mostrará o intervalo de capacidade atual do cluster. Nesse caso, não é possível alterar a capacidade. A imagem a seguir mostra o relatório da capacidade atual do cluster.



Se você já adicionou qualquer instância de banco de dados do Aurora Serverless v2 ao cluster, a inclusão de outra instância de banco de dados do leitor do Aurora Serverless v2 mostrará o intervalo de capacidade atual. A imagem a seguir mostra esses controles somente leitura.



Caso queira alterar o intervalo de capacidade do cluster, siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

No caso de clusters que contenham mais de uma instância de banco de dados do leitor, a prioridade de failover de cada instância de banco de dados do leitor do Aurora Serverless v2 desempenha um papel importante na forma como a escala dessa instância é aumentada ou reduzida na vertical. Não é possível especificar a prioridade ao criar inicialmente o cluster. Tenha em mente essa propriedade ao adicionar uma segunda instância de banco de dados do leitor ao cluster ou ao fazê-lo

posteriormente. Para obter mais informações, consulte Escolher o nível de promoção para um leitor do Aurora Serverless v2.

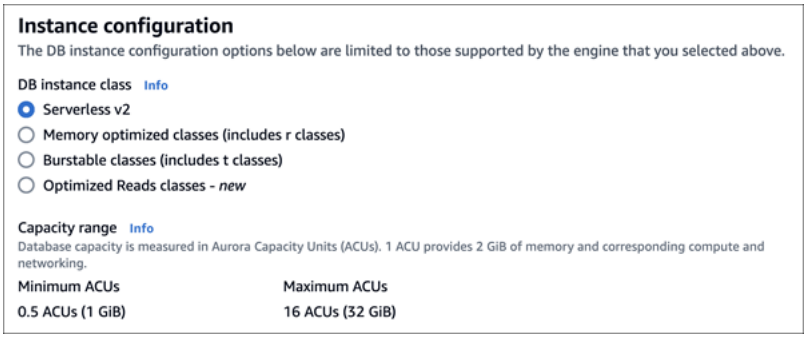
Para saber outras maneiras de ver o intervalo de capacidade atual de um cluster, consulte Conferir o intervalo de capacidade do Aurora Serverless v2.

Converter um gravador ou leitor provisionado em Aurora Serverless v2

É possível converter uma instância de banco de dados provisionada para usar o Aurora Serverless v2. Para fazer isso, siga o procedimento em Modificar uma instância de banco de dados em um cluster de banco de dados. O cluster deve atender aos requisitos em Requisitos e limitações do Aurora Serverless v2. Por exemplo, as instâncias de banco de dados do Aurora Serverless v2 exigem que o cluster esteja executando determinadas versões mínimas do mecanismo.

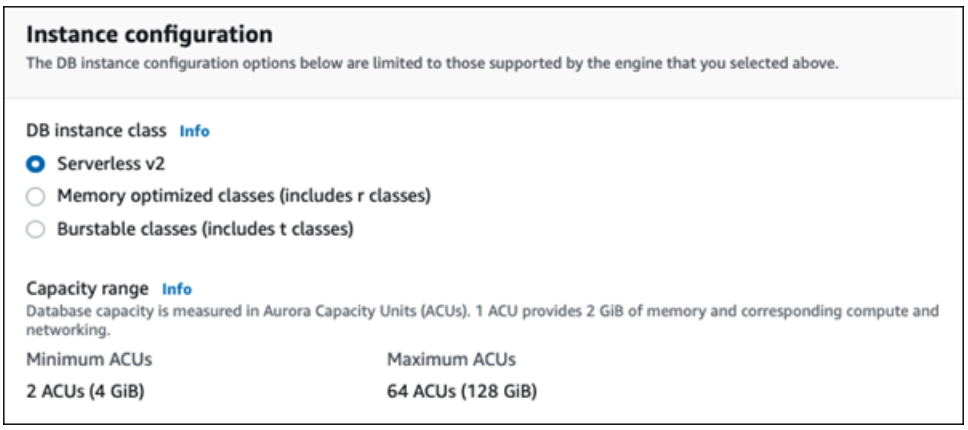
Suponha que você esteja convertendo um cluster provisionado em execução para aproveitar o Aurora Serverless v2. Nesse caso, é possível minimizar o tempo de inatividade convertendo uma instância de banco de dados em Aurora Serverless v2 como a primeira etapa no processo de alternância. Para saber o procedimento completo, consulte Alternar de um cluster provisionado para o Aurora Serverless v2.

Se a instância de banco de dados convertida for a primeira instância de banco de dados do Aurora Serverless v2 no cluster, escolha o intervalo de capacidade para o cluster como parte da operação Modify(Modificar). Esse intervalo de capacidade se aplica a todas as instâncias de banco de dados do Aurora Serverless v2 adicionadas ao cluster. A imagem a seguir mostra a página na qual especificar as unidades de capacidade mínima e máxima do Aurora (ACUs).



Para obter detalhes sobre a importância do intervalo de capacidade, consulte Aurora Serverless v2Capacidade do.

Se o cluster já contiver uma ou mais instâncias de banco de dados do Aurora Serverless v2, você verá o intervalo de capacidade existente durante a operação Modify (Modificar). A imagem a seguir mostra um exemplo desse painel de informações.



Caso queira alterar o intervalo de capacidade do cluster depois de adicionar mais instâncias de banco de dados do Aurora Serverless v2, siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

Converter um gravador ou leitor do Aurora Serverless v2 em provisionado

É possível converter uma instância de banco de dados do Aurora Serverless v2 em uma instância de banco de dados provisionada. Para fazer isso, siga o procedimento em Modificar uma instância de banco de dados em um cluster de banco de dados. Escolha uma classe de instância de banco de dados diferente de Serverless (Sem servidor).

Você pode converter um instância de banco de dados do Aurora Serverless v2 em provisionada se precisar de uma capacidade maior do que a disponível com as unidades de capacidade máxima do Aurora (ACUs) de uma instância de banco de dados do Aurora Serverless v2. Por exemplo, as maiores classes de instância de banco de dados db.r5 e db.r6g têm uma capacidade de memória maior do que a capacidade para a qual uma instância de banco de dados do Aurora Serverless v2 pode ser escalada.

> **Tip**
>
> Algumas classes de instância de banco de dados mais antigas, como db.r3 e db.t2, não estão disponíveis para as versões do Aurora que você usa com o Aurora Serverless v2. Para ver quais classes de instância de banco de dados você pode usar ao converter uma instância de banco de dados do Aurora Serverless v2 para uma provisionada, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.

Se você estiver convertendo a instância de banco de dados do gravador do seu cluster do Aurora Serverless v2 em provisionada, poderá seguir o procedimento em Alternar de um cluster provisionado para o Aurora Serverless v2, mas de forma inversa. Alterne uma das instâncias de banco de dados do leitor no cluster de Aurora Serverless v2 em provisionada. Depois, execute um failover para transformar essa instância de banco de dados provisionada no gravador.

O intervalo de capacidade especificado anteriormente para o cluster permanecerá em vigor, mesmo que todas as instâncias de banco de dados do Aurora Serverless v2 sejam removidas do cluster. Se quiser alterar o intervalo de capacidade, modifique o cluster, conforme explicado em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

Pausar instâncias de banco de dados do Aurora Serverless v2

Você pode configurar clusters do Aurora para pausar e retomar automaticamente as instâncias de banco de dados do Aurora Serverless v2. Para obter mais informações, consulte Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2.

Escolher o nível de promoção para um leitor do Aurora Serverless v2

No caso de clusters que contenham várias instâncias de banco de dados do Aurora Serverless v2 ou uma mistura de instâncias de banco de dados, do Aurora Serverless v2 e provisionadas, preste atenção à configuração do nível de promoção para cada instância de banco de dados do Aurora Serverless v2. Essa configuração controla mais aspectos do comportamento das instâncias de banco de dados do Aurora Serverless v2 do que das instâncias de banco de dados provisionadas.

No Console de gerenciamento da AWS, especifique essa configuração usando a opção Prioridade de failover em Configurações adicionaispara as páginas Criar banco de dados, Modificar instância e Adicionar leitor. Veja essa propriedade para instâncias de banco de dados existentes na coluna Priority tier (Nível prioritário) da página Databases (Bancos de dados). Você também pode ver essa propriedade na página de detalhes de um cluster de banco de dados ou instância de banco de dados.

No caso de instâncias de banco de dados provisionadas, a escolha do nível de 0 a 15 determina apenas a ordem na qual o Aurora escolhe qual instância de banco de dados do leitor promover ao gravador durante uma operação de failover. No caso de instâncias de banco de dados do leitor do Aurora Serverless v2, o número de níveis também determina se a instância de banco de dados tem a escala aumentada na vertical para corresponder à capacidade da instância de banco de dados do gravador ou se é escalada independentemente com base em sua própria workload. As instâncias de banco de dados do leitor do Aurora Serverless v2 nos níveis 0 ou 1 são mantidas

com uma capacidade mínima pelo menos tão alta quanto a da instância de banco de dados do gravador. Dessa forma, elas estão prontas para assumir o controle da instância de banco de dados do gravador em caso de failover. Se a instância de banco de dados do gravador for uma instância de banco de dados provisionada, o Aurora estimará a capacidade do Aurora Serverless v2 equivalente. Ele usa essa estimativa como a capacidade mínima para a instância de banco de dados do leitor do Aurora Serverless v2.

Aurora Serverless v2As instâncias de banco de dados de leitor do nos níveis 2 a 15 não têm a mesma restrição em sua capacidade mínima. Quando estiverem ociosas, elas poderão ter a escala reduzida na vertical para o valor mínimo da unidade de capacidade do Aurora (ACU) especificado no intervalo de capacidade do cluster.

O exemplo do AWS CLI do Linux a seguir mostra como examinar os níveis de promoção de todas as instâncias de banco de dados em seu cluster. O campo final inclui um valor de `True` para a instância de banco de dados do gravador e `False` para todas as instâncias de banco de dados do leitor.

```
$ aws rds describe-db-clusters --db-cluster-identifier promotion-tier-demo \
  --query 'DBClusters[*].DBClusterMembers[*].
[PromotionTier,DBInstanceIdentifier,IsClusterWriter]' \
  --output text

1   instance-192  True
1   tier-01-4840  False
10  tier-10-7425  False
15  tier-15-6694  False
```

O exemplo da AWS CLI do Linux a seguir mostra como alterar o nível de promoção de uma instância de banco de dados específica em seu cluster. Os comandos primeiro modificam a instância de banco de dados com um novo nível de promoção. Depois, eles esperam que a instância de banco de dados fique disponível novamente e confirmam o novo nível de promoção para a instância de banco de dados.

```
$ aws rds modify-db-instance --db-instance-identifier instance-192 --promotion-tier 0
$ aws rds wait db-instance-available --db-instance-identifier instance-192
$ aws rds describe-db-instances --db-instance-identifier instance-192 \
  --query '*[].[PromotionTier]' --output text
0
```

Para obter mais orientações sobre como especificar níveis de promoção para diferentes casos de uso, consulte Aurora Serverless v2Escalabilidade do.

Usar TLS/SSL com o Aurora Serverless v2

O Aurora Serverless v2 pode usar o protocolo Transport Layer Security/Secure Sockets Layer (TLS/SSL) para criptografar as comunicações entre clientes e suas instâncias de banco de dados do Aurora Serverless v2. Ele é compatível com TLS/SSL versões 1.0, 1.1 e 1.2. Para obter informações gerais sobre como usar o TLS/SSL com o Aurora, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

Para saber mais sobre como se conectar ao banco de dados do Aurora MySQL com o cliente MySQL, consulte Conectar-se a uma instância de banco de dados que esteja executando o mecanismo de banco de dados MySQL.

O Aurora Serverless v2 é compatível com todos os modos TLS/SSL disponíveis para o cliente do MySQL (`mysql`) e o cliente do PostgreSQL (`psql`), incluindo aqueles listados na tabela a seguir.

Descrição do modo TLS/SSL	mysql	psql
Conectar sem usar TLS/SSL.	DISABLED	desabilitar
Tente a conexão usando TLS/SSL primeiro, mas volte para não SSL, se necessário.	PREFERRED	preferir (padrão)
Imponha o uso de TLS/SSL.	REQUIRED	require
Imponha o TLS/SSL e verifique a autoridade de certificação (CA).	VERIFY_CA	verify-ca
Imponha o TLS/SSL, verifique a CA e verifique o hostname da CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v2O usa certificados curinga. Se você especificar a opção “verificar CA” ou “verificar CA e nome do host da CA” ao usar TLS/SSL, primeiro baixe o armazenamento de confiança CA 1 raiz da Amazon no Amazon Trust Services. Depois de fazer isso, você pode identificar esse arquivo formatado em PEM em seu comando client. Para fazer isso usando o cliente PostgreSQL, faça o seguinte.

Para Linux, macOS ou Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Para saber mais sobre como trabalhar com o banco de dados do Aurora PostgreSQL usando o cliente do Postgres, consulte Conectar-se a uma instância de banco de dados que esteja executando o mecanismo de banco de dados do PostgreSQL.

Para obter mais informações sobre como se conectar a clusters de bancos de dados Aurora em geral, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Pacotes de cifras compatíveis com clusters de banco de dados do Aurora Serverless v2

Usando conjuntos de cifras configuráveis, você pode ter mais controle sobre a segurança de suas conexões de banco de dados. É possível especificar uma lista de conjuntos de cifras que você deseja permitir para proteger conexões TLS/SSL do cliente com o banco de dados. Com conjuntos de cifras configuráveis, você pode controlar a criptografia de conexão aceita pelo servidor de banco de dados. Isso impede o uso de cifras que não são seguras ou que não são mais usadas.

Aurora Serverless v2Os clusters de banco de dados do que são baseados no Aurora MySQL são compatíveis com os mesmos conjuntos de cifras que os clusters de banco de dados provisionados pelo Aurora MySQL. Para obter informações sobre esses conjuntos de cifras, consulte Configurar conjuntos de criptografia para conexões com clusters de banco de dados do Aurora MySQL.

Aurora Serverless v2Os clusters de banco de dados do que são baseados no Aurora PostgreSQL são compatíveis com os mesmos conjuntos de cifras que os clusters de banco de dados provisionados pelo Aurora PostgreSQL. Para obter informações sobre esses conjuntos de cifras, consulte Configurar conjuntos de cifras para conexões a clusters de banco de dados PostgreSQL do Aurora.

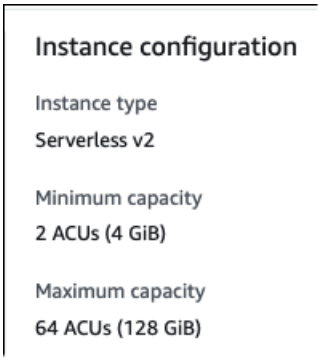
Visualizar gravadores e leitores do Aurora Serverless v2

É possível visualizar os detalhes das instâncias de banco de dados do Aurora Serverless v2 da mesma forma que você faz para instâncias de banco de dados provisionadas. Para isso, siga o procedimento geral de Visualizar um cluster de bancos de dados Amazon Aurora. Um cluster pode conter todas as instâncias de banco de dados do Aurora Serverless v2, todas as instâncias de banco de dados provisionadas ou algumas de cada.

Depois de criar uma ou mais instâncias de banco de dados do Aurora Serverless v2, será possível visualizar quais delas são do tipo Serverless (Sem servidor) e quais são do tipo Instance (Instância). Também é possível visualizar as unidades de capacidade mínima e máxima do Aurora (ACUs) que a instância de banco de dados do Aurora Serverless v2 pode usar. Cada ACU é uma combinação da capacidade de processamento (CPU) e de memória (RAM). Esse intervalo de capacidade se aplica a todas as instâncias de banco de dados do Aurora Serverless v2 no cluster. Para que o procedimento confira o intervalo de capacidade de um cluster ou de qualquer instância de banco de dados do Aurora Serverless v2 no cluster, consulte Conferir o intervalo de capacidade do Aurora Serverless v2.

No Console de gerenciamento da AWS, as instâncias de banco de dados do Aurora Serverless v2 são marcadas sob a coluna Size (Tamanho) na página Databases (Bancos de dados). As instâncias de banco de dados provisionadas mostram o nome de uma classe de instância de banco de dados, como r6g.xlarge. As instâncias de banco de dados do Aurora Serverless mostram Serverless (Sem servidor) para a classe de instância de banco de dados, juntamente com a capacidade mínima e máxima da instância de banco de dados. Por exemplo, você pode ver Serverless v2 (4—64 ACUs) ou Serverless v2 (1-40 ACUs).

Você pode encontrar as mesmas informações na guia Configuration (Configuração) para cada instância de banco de dados do Aurora Serverless v2 no console. Por exemplo, talvez você veja uma seção Instance type (Tipo de instância) como a seguinte. Aqui, o valor Instance type (Tipo de instância) é Serverless v2, o valor Minimum capacity (Capacidade mínima) é 2 ACUs (4 GiB) e o valor Maximum capacity (Capacidade máxima) é 64 ACUs (128 GiB).



Você pode monitorar a capacidade de cada instância de banco de dados do Aurora Serverless v2 com o tempo. Dessa forma, é possível conferir as ACUs mínima, máxima e média consumidas por cada instância de banco de dados. Você também pode conferir a proximidade da instância de banco de dados de sua capacidade mínima ou máxima. Para ver esses detalhes no Console de gerenciamento da AWS, examine os grafos das métricas do Amazon CloudWatch na guia Monitoring (Monitoramento) para a instância de banco de dados. Para obter mais informações sobre as métricas

a serem observadas e interpretá-las, consulte Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2.

Registro em log para o Aurora Serverless v2

Para ativar o registro em log do banco de dados, especifique os logs a serem habilitados usando parâmetros de configuração em seu grupo de parâmetros personalizado.

No caso do Aurora MySQL, você pode habilitar os logs a seguir.

Aurora MySQL	Descrição
`general_log`	Cria o log geral. Defina como 1 para ativar. O padrão é desativado (0).
`log_queries_not_using_indexes`	Registra todas as consultas no log de consultas lentas que não usam um índice. O padrão é desativado (0). Defina como 1 para ativar esse log.
`long_query_time`	Impede que consultas em execução rápida sejam registradas no log de consultas lentas. Pode ser definido como um float entre 0 e 31536000. O padrão é 0 (não ativo).
`server_audit_events`	A lista de eventos a serem capturados nos logs. Os valores compatíveis são `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Defina como 1 para ativar o log de auditoria do servidor. Se você ativar isso, você poderá especificar os eventos de auditoria a serem enviados, CloudWatch listando-os no `server_audit_events` parâmetro.
`slow_query_log`	Cria um log de consulta lento. Defina como 1 para ativar o log de consulta lenta. O padrão é desativado (0).

Para obter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.

Para o Aurora PostgreSQL, você pode habilitar os logs a seguir em suas instâncias de banco de dados do Aurora Serverless v2.

Aurora PostgreSQL	Descrição
`log_connections`	Registra cada conexão bem-sucedida.
`log_disconnections`	Registra o fim de uma sessão, incluindo a duração.
`log_lock_waits`	O padrão é 0 (desativado). Defina como 1 para registrar esperas por bloqueio.
`log_min_duration_statement`	A duração mínima (em milissegundos) para que uma instrução seja executada antes de ser registrada.
`log_min_messages`	Define os níveis de mensagem registrados. Os valores compatíveis são `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Para registrar dados de performance no log do `postgres`, defina o valor como `debug1`.
`log_temp_files`	Registra o uso de arquivos temporários que estão acima dos kilobytes (kB) especificados.
`log_statement`	Controla as instruções SQL específicas que são registradas. Os valores compatíveis são `none`, `ddl`, `mod` e `all`. O padrão é `none`.

Tópicos

- Registro em log com o Amazon CloudWatch
- Visualizar logs do Aurora Serverless v2 no Amazon CloudWatch
- Monitoramento da capacidade com Amazon CloudWatch

- Monitorar a atividade de pausa e retomada do Aurora Serverless v2

Registro em log com o Amazon CloudWatch

Depois de seguir o procedimento em Registro em log para o Aurora Serverless v2 para escolher quais logs de banco de dados ativar, você pode escolher quais logs carregar (“publicar”) para o Amazon CloudWatch.

É possível usar o Amazon CloudWatch para analisar os dados de log, criar alarmes e visualizar métricas. Por padrão, logs de erros para o Aurora Serverless v2 são habilitados e carregados automaticamente no CloudWatch. Você também pode carregar outros logs de instâncias de banco de dados do Aurora Serverless v2 no CloudWatch.

Depois, selecione de qual desses logs fazer upload para o CloudWatch usando as configurações Log exports (Exportações de log) no Console de gerenciamento da AWS ou a opção `--enable-cloudwatch-logs-exports` na AWS CLI.

É possível escolher qual dos seus logs do Aurora Serverless v2 carregar para o CloudWatch. Para obter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.

Como em qualquer tipo de cluster de Aurora banco de dados, você não pode modificar o grupo de parâmetros de cluster de banco de dados padrão. Em vez disso, crie seu próprio grupo de parâmetros de cluster de banco de dados com base em um parâmetro padrão para o cluster de banco de dados e o tipo de mecanismo. Recomendamos que você crie seu grupo de parâmetros de cluster de banco de dados personalizado antes de criar seu cluster de Aurora Serverless v2 banco de dados, para que ele esteja disponível para escolher quando você cria um banco de dados no console.

> **Note**
>
> Para o Aurora Serverless v2, você pode criar um cluster de banco de dados e grupos de parâmetros de banco de dados. Isso é diferente com o Aurora Serverless v1, no qual você só pode criar grupos de parâmetros de cluster de banco de dados.

Visualizar logs do Aurora Serverless v2 no Amazon CloudWatch

Depois de usar o procedimento em Registro em log com o Amazon CloudWatch para escolher quais logs de banco de dados serão ativados, você pode visualizar o conteúdo dos logs.

Para obter mais informações sobre como usar o CloudWatch com os logs do CloudWatch ou do Aurora MySQL, consulte Monitorar eventos de log no Amazon CloudWatch e Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs.

Para visualizar logs do cluster de banco de dados do Aurora Serverless v2

1. Abra o console do CloudWatch, em https://console.aws.amazon.com/cloudwatch/.
2. Escolha a Região da AWS.
3. Escolha Grupos de logs.
4. Escolha o log do cluster de banco de dados do Aurora Serverless v2 na lista. O padrão de nomenclatura de log é o seguinte.

```
/aws/rds/cluster/cluster-name/log_type
```

> **Note**
>
> Para clusters de banco de dados Aurora Serverless v2 compatíveis com o Aurora MySQL, o log de erros inclui eventos de escalabilidade do grupo de buffer mesmo quando não há erros.

Monitoramento da capacidade com Amazon CloudWatch

Com Aurora Serverless v2, é possível usar o CloudWatch para monitorar a capacidade e a utilização de todas as instâncias de banco de dados do Aurora Serverless v2 em seu cluster. Você pode visualizar métricas específicas da instância para conferir a capacidade de cada instância de banco de dados do Aurora Serverless v2 à medida que a escala dela é reduzida ou aumentada na vertical. Você também pode comparar as métricas relacionadas à capacidade com outras métricas para ver como as alterações nas workloads afetam o consumo de recursos. Por exemplo, você pode comparar `ServerlessDatabaseCapacity` com `DatabaseUsedMemory`, `DatabaseConnections` e `DMLThroughput` para avaliar como seu cluster de banco de dados está respondendo durante as operações. Para obter detalhes sobre as métricas relacionadas à

capacidade que se aplicam ao Aurora Serverless v2, consulte Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2.

Para monitorar a capacidade do Aurora Serverless v2 cluster de banco de dados

1. Abra o console do CloudWatch, em https://console.aws.amazon.com/cloudwatch/.
2. Escolha Metrics (Métricas). Todas as métricas disponíveis aparecem como cartões no console, agrupadas por nome do serviço.
3. Escolha RDS.
4. (Opcional) Use a caixa Search (Pesquisar) para encontrar as métricas que são especialmente importantes para Aurora Serverless v2: `ServerlessDatabaseCapacity`, `ACUUtilization`, `CPUUtilization` e `FreeableMemory`.

Recomendamos que você configure um painel do CloudWatch para monitorar a capacidade do cluster de banco de dados do Aurora Serverless v2 usando as métricas relacionadas à capacidade. Para saber como, consulte Criação de painéis com o CloudWatch.

Para saber mais sobre o uso do Amazon CloudWatch com o Amazon Aurora, consulte Publicar logs do Amazon Aurora MySQL no Amazon CloudWatch Logs.

Monitorar a atividade de pausa e retomada do Aurora Serverless v2

O Aurora grava um arquivo de log separado para instâncias de banco de dados do Aurora Serverless v2 com a pausa automática habilitada. O Aurora grava no log em cada intervalo de 10 minutos no qual a instância não está pausada. O Aurora retém até sete desses logs, alternados diariamente. O arquivo de log atual é nomeado `instance.log`, e os logs mais antigos são nomeados usando o padrão `instance.`*`YYYY-MM-DD`*`.`*`N`*`.log`.

Esse log é habilitado por padrão para instâncias de banco de dados do Aurora Serverless v2 com a pausa automática habilitada. Você pode visualizar o conteúdo desse log no Console de gerenciamento da AWS ou usando a AWS CLI ou a API do RDSP. Atualmente, não é possível fazer o upload desse log para o CloudWatch.

Os eventos listados nos Eventos de instância de banco de dados fornecem uma visão geral de alto nível das atividades de pausa e retomada, como as seguintes:

- Quando a instância começa a pausar e quando termina de pausar.
- Quando a instância começa a ser retomada e quando termina de ser retomada.

- Casos em que a instância tentou pausar, mas alguma condição a impediu.

O `instance.log` fornece detalhes mais granulares sobre os motivos pelos quais uma instância do Aurora Serverless v2 conseguiu ou não pausar.

O log pode indicar que uma instância foi retomada por diferentes motivos:

- atividade do usuário: uma solicitação de conexão com o banco de dados. Pode ser de uma sessão interativa do cliente, de uma chamada de API de dados do RDS ou de uma solicitação de download dos logs da instância.
- atividade em segundo plano: uma categoria ampla que inclui todos os motivos pelos quais o Aurora retoma uma instância. Por exemplo, quando uma solicitação de conexão a uma instância de leitura faz com que a instância de gravação seja retomada, o log do leitor indica atividade do usuário, enquanto o log do gravador classifica essa solicitação de retomada como atividade em segundo plano. Consulte Retomar uma instância do Aurora Serverless v2 pausada automaticamente para saber os motivos pelos quais o Aurora pode retomar uma instância além de uma solicitação de conexão do usuário.

Quando o Aurora não está ciente de nenhuma condição que impeça a instância de pausar quando o intervalo de pausa automática expirar, ele grava periodicamente uma mensagem informativa no log. O log apresenta a seguinte mensagem para clusters que contém uma única instância de banco de dados:

```
[INFO] No auto-pause blockers registered since time
```

Para clusters com várias instâncias de banco de dados, a mensagem é um pouco diferente. Isso porque um gravador pode não conseguir fazer uma pausa devido à atividade em qualquer uma das instâncias de leitura. Se a atividade no leitor terminar antes que o intervalo de pausa automática expire no gravador, o gravador poderá pausar no horário esperado.

```
[INFO] No auto-pause blockers registered since time.
Database might be required to maintain compute capacity for high availability.
```

Se uma operação de pausa for iniciada, mas uma nova solicitação de conexão de banco de dados chegar antes que a instância termine de pausar, o log apresentará esta mensagem:

```
[INFO] Unable to pause database due to a new database activity
```

Se o Aurora tomar conhecimento de quaisquer condições que definitivamente impeçam a pausa da instância, o log apresentará a seguinte mensagem que lista todas essas condições:

```
[INFO] Auto-pause blockers registered since time: list_of_conditions
```

Dessa forma, o Aurora não impede que você ative a replicação, a integração ETL zero, o Aurora Global Database e outros em combinação com o recurso de pausa automática. O log informa quando o uso desses recursos pode impedir que a pausa automática entre em vigor.

A seguir estão os motivos pelos quais uma instância do Aurora Serverless v2 pode exceder o intervalo de tempo limite da pausa automática, mas ser impedida de pausar:

- atividade do banco de dados antes do tempo limite da pausa automática: a instância de banco de dados recebeu uma solicitação de conexão antes que o intervalo de tempo limite expirasse.
- membro do banco de dados global: se o cluster de banco de dados fizer parte de um banco de dados global do Aurora, as instâncias do Aurora Serverless v2 no cluster não pausarão. Um cluster pode mudar de um cluster autônomo para um cluster de banco de dados global. Assim, instâncias que antes eram pausadas automaticamente podem parar de pausar e relatar esse motivo no log. Quando um cluster se torna membro de um banco de dados global, ele não volta a ser um cluster autônomo até ser desanexado explicitamente. O cluster primário ainda é considerado parte do banco de dados global, mesmo se você desanexar todos os clusters secundários.
- capacidade de replicação configurada: a instância de banco de dados do gravador tem a replicação específica do mecanismo habilitada, seja replicação de binlog para MySQL ou replicação lógica para PostgreSQL. Essa condição também pode ser causada pelo uso de outro recurso do Aurora que exige a ativação da replicação, como integrações ETL zero ou fluxos de atividade de banco de dados (DAS).
- atraso contínuo no backup: se o sistema de armazenamento Aurora não terminar de aplicar as alterações de armazenamento até o momento atual, a instância de gravação não pausa até que se atualize. Essa condição afeta apenas a instância de gravação e espera-se que seja relativamente breve.
- ação de manutenção do serviço ou do cliente: se uma operação de manutenção for iniciada, a instância de banco de dados não pausará novamente até que a operação termine. Essa condição inclui uma ampla variedade de operações que podem ser iniciadas por você ou pelo Aurora, como atualizações, clonagem, alteração de configurações, download de arquivos de log e outros. Esse evento também acontece quando você solicita a exclusão de uma instância e o Aurora retoma brevemente a instância como parte do mecanismo de exclusão.

- problema de comunicação transitório: se o Aurora não conseguir determinar se a configuração de escalabilidade atualmente tem uma configuração de capacidade mínima de zero ACUs, ele não pausa a instância. Espera-se que essa seja uma ocorrência muito rara.

Performance e escalabilidade no Aurora Serverless v2

Os procedimentos e exemplos a seguir mostram como definir o intervalo de capacidade para clusters do Aurora Serverless v2 e suas instâncias de banco de dados associadas. Você também pode usar os procedimentos a seguir para monitorar o nível de ocupação de suas instâncias de banco de dados. Depois, é possível usar suas constatações para determinar se é necessário ajustar o intervalo de capacidade para cima ou para baixo.

Antes de usar estes procedimentos, verifique se sabe com funciona a escalabilidade do Aurora Serverless v2. O mecanismo de escalabilidade difere do Aurora Serverless v1. Para obter detalhes, consulte Aurora Serverless v2Escalabilidade do.

Sumário

- escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora
  - Escolher a configuração de capacidade mínima do Aurora Serverless v2 para um cluster
  - Escolher a configuração de capacidade máxima do Aurora Serverless v2 para um cluster
  - Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora MySQL
  - Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora PostgreSQL
- Trabalhar com grupos de parâmetros para o Aurora Serverless v2
  - Valores de parâmetro padrão
  - Conexões máximas do Aurora Serverless v2
  - Parâmetros ajustados pelo Aurora à medida que a escala do Aurora Serverless v2 é aumentada ou reduzida na vertical
  - Parâmetros calculados pelo Aurora com base na capacidade máxima do Aurora Serverless v2
- Evitar erros de falta de memória
- Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2
  - Como as métricas Aurora Serverless v2 se aplicam à sua fatura da AWS
  - Exemplos de comandos do CloudWatch para métricas do Aurora Serverless v2

- Monitorar a performance do Aurora Serverless v2 com o Performance Insights
- Solução de problemas de capacidade do Aurora Serverless v2

escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora

Com instâncias de banco de dados do Aurora Serverless v2, você define o intervalo de capacidade que se aplica a todas as instâncias de banco de dados em seu cluster ao mesmo tempo em que adiciona a primeira instância de banco de dados do Aurora Serverless v2. Para saber o procedimento, consulte Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.

Você também pode alterar o intervalo de capacidade de um cluster existente. As seções a seguir abordam com mais detalhes como escolher valores mínimos e máximos apropriados e o que acontece quando você faz uma alteração no intervalo de capacidade. Por exemplo, alterar o intervalo de capacidade pode modificar os valores padrão de alguns parâmetros de configuração. Aplicar todas as alterações de parâmetros pode exigir a reinicialização de cada instância de banco de dados do Aurora Serverless v2.

Tópicos

- Escolher a configuração de capacidade mínima do Aurora Serverless v2 para um cluster
- Escolher a configuração de capacidade máxima do Aurora Serverless v2 para um cluster
- Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora MySQL
- Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora PostgreSQL

Escolher a configuração de capacidade mínima do Aurora Serverless v2 para um cluster

É tentador escolher sempre 0,5 para a configuração de capacidade mínima do Aurora Serverless v2. Esse valor permite que a instância de banco de dados reduza sua escala na vertical para a menor capacidade quando estiver completamente ociosa, permanecendo ativa. Você também pode ativar o comportamento de pausa automática especificando uma capacidade mínima de 0 ACUs, conforme explicado em Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2. No entanto, dependendo de como você usa esse cluster e das outras configurações definidas, uma

capacidade mínima diferente pode ser o mais eficaz. Considere os seguintes fatores ao escolher a configuração de capacidade mínima:

- A taxa de escalabilidade para uma instância de banco de dados do Aurora Serverless v2 depende de sua capacidade atual. Quanto maior a capacidade atual, com maior rapidez será possível aumentar a escala dela na vertical. Se você precisar aumentar a escala da instância de banco de dados na vertical com rapidez até uma capacidade muito alta, considere definir a capacidade mínima como um valor em que o intervalo de escalabilidade atenda às suas necessidades.
- Se você normalmente modifica a classe de suas instâncias de banco de dados prevendo uma workload especialmente alta ou baixa, é possível usar essa experiência para realizar uma estimativa aproximada do intervalo de capacidade do Aurora Serverless v2 equivalente. Para determinar o tamanho da memória a ser usado em momentos de baixo tráfego, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

  Por exemplo, suponha que você use a classe de instância de banco de dados db.r6g.xlarge quando o cluster tem uma workload baixa. Essa classe de instância de banco de dados tem 32 GiB de memória. Assim, é possível especificar uma configuração mínima de unidade de capacidade do Aurora (ACU) de 16 para configurar uma instância de banco de dados do Aurora Serverless v2 que possa ser reduzida para aproximadamente a mesma capacidade. O motivo é que cada ACU corresponde a aproximadamente 2 GiB de memória. Você pode especificar um valor um pouco menor para permitir que a instância de banco de dados seja reduzida ainda mais no caso de sua instância db.r6g.xlarge às vezes ser subutilizada.
- Se a aplicação funcionar de forma mais eficiente quando as instâncias de banco de dados tiverem uma certa quantidade de dados no cache de buffer, considere especificar uma configuração mínima de ACU em que a memória seja grande o suficiente para manter os dados acessados com frequência. Caso contrário, alguns dados serão removidos do cache de buffer quando a escala das instâncias de banco de dados do Aurora Serverless v2 é reduzida na vertical para um tamanho de memória menor. Então, quando a escala das instâncias de banco de dados for novamente aumentada na vertical, as informações serão lidas de volta no cache do buffer ao longo do tempo. Se a quantidade de E/S para trazer os dados de volta para o cache de buffer for substancial, talvez seja mais eficaz escolher um valor mínimo de ACU mais alto.
- Se suas instâncias de banco de dados do Aurora Serverless v2 são executadas na maior parte do tempo em uma determinada capacidade, considere especificar uma configuração de capacidade mínima inferior à linha de base, mas não muito mais baixa. Aurora Serverless v2 As instâncias de banco de dados podem estimar com mais eficiência quanto e com que rapidez a escala será aumentada na vertical quando a capacidade atual não é drasticamente menor do que a capacidade necessária.

- Se a workload provisionada tiver requisitos de memória muito altos para classes de instâncias de banco de dados pequenas, como T3 ou T4g, escolha uma configuração mínima de ACU que forneça memória comparável a uma instância de banco de dados R5 ou R6g.

  Especificamente, recomendamos a seguinte capacidade mínima para uso com os recursos especificados (essas recomendações estão sujeitas a alterações):

  - Performance Insights: duas ACUs
  - Bancos de dados globais do Aurora: oito ACUs (aplica-se somente à Região da AWS primária)
- No Aurora, a replicação ocorre na camada de armazenamento, portanto, a capacidade do leitor não afeta diretamente a replicação. No entanto, para instâncias de banco de dados de leitor do Aurora Serverless v2 que escalam de forma independente, certifique-se de que a capacidade mínima seja suficiente para lidar com workloads durante períodos intensivos de gravação, a fim de evitar a latência da consulta. Se instâncias de banco de dados de leitor nos níveis de promoção de 2 a 15 apresentarem problemas de desempenho, considere aumentar a capacidade mínima do cluster. Para obter detalhes sobre como determinar se as instâncias de banco de dados do leitor são escaladas junto com o gravador ou de forma independente, consulte Escolher o nível de promoção para um leitor do Aurora Serverless v2.
- Se você tem um cluster de banco de dados com instâncias de banco de dados de leitor do Aurora Serverless v2, os leitores não escalam junto com a instância de banco de dados de gravador quando o nível de promoção dos leitores não é 0 ou 1. Nesse caso, definir uma capacidade mínima baixa pode ocasionar atraso excessivo de replicação. Isso ocorre porque os leitores podem não ter capacidade suficiente para aplicar alterações do gravador quando o banco de dados está ocupado. Recomendamos definir a capacidade mínima como um valor que represente uma quantidade comparável de memória e CPU à instância de banco de dados de gravador.
- O valor do parâmetro `max_connections` para instâncias de banco de dados do Aurora Serverless v2 é baseado no tamanho da memória derivado do máximo de ACUs. No entanto, quando você especifica uma capacidade mínima de 0 ou 0,5 ACU em instâncias de banco de dados compatíveis com o PostgreSQL, o valor máximo de `max_connections` é limitado a 2.000.

  Se você pretende usar o cluster do Aurora PostgreSQL para uma workload de alta conexão, considere usar uma configuração mínima de ACU de 1 ou superior. Para obter detalhes sobre como o Aurora Serverless v2 lida com o parâmetro de configuração `max_connections`, consulte Conexões máximas do Aurora Serverless v2.
- O tempo necessário para uma instância de banco de dados do Aurora Serverless v2 escalar de sua capacidade mínima para sua capacidade máxima depende da diferença entre seus valores mínimo e máximo de ACU. Quando a capacidade atual da instância de banco de dados é grande,

a escala do Aurora Serverless v2 é aumentada na vertical em incrementos maiores do que quando a instância de banco de dados parte de uma pequena capacidade. Assim, se você especificar uma capacidade máxima relativamente grande e a instância de banco de dados passar a maior parte do tempo próxima dessa capacidade, considere aumentar a configuração mínima de ACU. Dessa forma, a escala de uma instância de banco de dados ociosa pode ser aumentada na vertical novamente até a capacidade máxima com maior rapidez.

Escolher a configuração de capacidade máxima do Aurora Serverless v2 para um cluster

É tentador sempre escolher algum valor alto para configuração de capacidade do Aurora Serverless v2. Uma capacidade máxima grande permite que a escala da instância de banco de dados seja aumentada na vertical ao máximo quando está executando uma workload intensa. Um valor baixo evita a possibilidade de cobranças inesperadas. Dependendo de como você usa esse cluster e das outras configurações definidas, o valor mais efetivo pode ser maior ou menor do que o imaginado originalmente. Considere os seguintes fatores ao escolher a configuração de capacidade máxima:

- A capacidade máxima deve ser pelo menos tão alta quanto a capacidade mínima. É possível definir a capacidade mínima e máxima como valores idênticos. No entanto, nesse caso, a escala da capacidade nunca é aumentada nem reduzida na vertical. Assim, usar valores idênticos para a capacidade mínima e máxima não é apropriado além das situações de teste.
- A capacidade máxima deve ser superior a 0,5 ACU. É possível definir a capacidade mínima e máxima como o mesmo valor na maioria dos casos. No entanto, não é possível especificar 0,5 para o mínimo e o máximo. Use um valor de 1 ou superior para a capacidade máxima.
- Se você normalmente modifica a classe de suas instâncias de banco de dados prevendo uma workload especialmente alta ou baixa, é possível usar essa experiência para calcular o intervalo de capacidade do Aurora Serverless v2 equivalente. Para determinar o tamanho da memória a ser usado em momentos de alto tráfego, consulte Especificações de hardware para classes de instância de banco de dadospara o Aurora.

  Por exemplo, suponha que você use a classe de instância de banco de dados db.r6g.4xlarge quando o cluster tem uma workload alta. Essa classe de instância de banco de dados tem 128 GiB de memória. Assim, é possível especificar uma configuração máxima de ACU de 64 para configurar uma instância de banco de dados do Aurora Serverless v2 cuja escala possa ser aumentada na vertical para aproximadamente a mesma capacidade. O motivo é que cada ACU corresponde a aproximadamente 2 GiB de memória. Você pode especificar um valor um pouco maior para aumentar ainda mais a escala da instância de banco de dados na vertical, caso sua

instância de banco de dados db.r6g.4xlarge às vezes não tenha capacidade suficiente para lidar com a workload de forma eficaz.

- Se você tiver um limite orçamentário para o uso do banco de dados, escolha um valor que permaneça dentro desse limite, mesmo que todas as suas instâncias de banco de dados do Aurora Serverless v2 sejam executadas na capacidade máxima o tempo todo. Lembre-se de que, quando há n instâncias de banco de dados do Aurora Serverless v2 em seu cluster, em tese, a capacidade do Aurora Serverless v2 que o cluster pode consumir a qualquer momento é n vezes a configuração máxima de ACU para o cluster. (A quantidade real consumida pode ser menor, por exemplo, se alguns leitores forem escalados independentemente do gravador.)
- Se você fizer uso de instâncias de banco de dados do leitor do Aurora Serverless v2 para descarregar parte da workload somente leitura da instância de banco de dados do gravador, talvez você possa escolher uma configuração de capacidade máxima mais baixa. Esse procedimento é realizado para indicar que nem todas as instâncias de banco de dados do leitor precisam ser escaladas para um valor tão alto como seria necessário se o cluster contivesse apenas uma única instância de banco de dados.
- Suponha que você queira evitar uso excessivo devido a parâmetros de banco de dados mal configurados ou consultas ineficientes em sua aplicação. Nesse caso, você pode evitar o uso excessivo acidental escolhendo uma configuração de capacidade máxima menor que a mais alta absoluta que seja possível definir.
- Se os picos decorrentes da atividade real do usuário forem raros, mas acontecerem, você pode levar essas ocasiões em consideração ao escolher a configuração de capacidade máxima. Se a prioridade for a aplicação continuar a funcionar com performance e escalabilidade totais, especifique uma configuração de capacidade máxima maior do que a observada no uso normal. Se for aceitável a aplicação ser executada com throughput reduzido durante picos extremos de atividade, escolha uma configuração de capacidade máxima um pouco menor. Escolha uma configuração que ainda tenha memória e recursos de CPU suficientes para manter a aplicação em execução.
- Se você ativar configurações em seu cluster que aumentem o uso de memória para cada instância de banco de dados, leve essa memória em consideração ao decidir sobre o valor máximo de ACU. Estão entre essas configurações aquelas para Performance Insights, consultas paralelas do Aurora MySQL, esquema de performance do Aurora MySQL e replicação de log binário do Aurora MySQL. Verifique se o valor máximo de ACU permite que a escala das instâncias de banco de dados do Aurora Serverless v2 seja aumentada na vertical o suficiente para lidar com a workload quando esses recursos estiverem sendo usados. Para obter informações sobre a solução de

problemas causados pela combinação de uma configuração de ACU máxima baixa e recursos do Aurora que ocasionam sobrecarga de memória, consulte Evitar erros de falta de memória.

Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora MySQL

Os exemplos da AWS CLI a seguir mostram como atualizar o intervalo da ACU para instâncias de banco de dados do Aurora Serverless v2 em um cluster existente do Aurora MySQL. Inicialmente, o intervalo de capacidade do cluster é de 8 a 32 ACUs.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

A instância de banco de dados está ociosa e sua escala é reduzida verticalmente para 8 ACUs. As configurações relacionadas à capacidade a seguir se aplicam à instância de banco de dados neste momento. Para representar o tamanho do grupo de buffer em unidades facilmente legíveis, o dividimos por dois para a potência de 30, produzindo uma medida em gibibytes (GiB). O motivo é que as medições relacionadas à memória para o Aurora usam unidades baseadas em potências de dois, não em potências de dez.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|                9294577664 |
+---------------------------+
1 row in set (0.00 sec)
```

```
mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|   8.65625 |
+-----------+
1 row in set (0.00 sec)
```

Depois, alteramos o intervalo de capacidade do cluster. Após o término do comando `modify-db-cluster`, o intervalo de ACU do cluster é de 12,5 a 80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

A alteração do intervalo de capacidade causou alterações nos valores padrão de alguns parâmetros de configuração. O Aurora pode aplicar alguns desses novos padrões imediatamente. No entanto, algumas das alterações de parâmetros são aplicadas somente após a reinicialização. O status do `pending-reboot` indica que é necessária uma reinicialização para aplicar algumas alterações de parâmetro.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

Neste ponto, o cluster está ocioso e a instância de banco de dados `serverless-v2-instance-1` está consumindo 12,5 ACUs. O parâmetro `innodb_buffer_pool_size` já está ajustado com base na capacidade atual da instância de banco de dados. O parâmetro `max_connections` ainda reflete o valor da capacidade máxima anterior. A redefinição desse valor exige a reinicialização da instância de banco de dados.

> **Note**
>
> Se você definir o parâmetro `max_connections` diretamente em um grupo de parâmetros de banco de dados personalizado, nenhuma reinicialização será necessária.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               15572402176 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+---------------+
| gibibytes     |
+---------------+
| 14.5029296875 |
+---------------+
1 row in set (0.00 sec)
```

Agora reinicializamos a instância de banco de dados e aguardamos até que ela fique disponível novamente.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
```

```
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

O status `pending-reboot` está limpo. O valor `in-sync` confirma que o Aurora aplicou todas as alterações de parâmetros pendentes.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

O parâmetro `innodb_buffer_pool_size` aumentou para seu tamanho final para uma instância de banco de dados ociosa. O parâmetro `max_connections` aumentou para refletir um valor derivado do valor máximo de ACU. A fórmula que o Aurora usa para `max_connections` causa um aumento de 1.000 quando o tamanho da memória dobra.

```
mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               16139681792 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|  15.03125 |
+-----------+
1 row in set (0.00 sec)
```

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              4000 |
+-------------------+
1 row in set (0.00 sec)
```

Definimos o intervalo de capacidade de 0,5 a 128 ACUs e reinicializamos a instância de banco de dados. Agora, a instância de banco de dados ociosa tem um tamanho de cache de buffer menor que 1 GiB, então nós a medimos em mebibytes (MiB). O valor `max_connections` de 5.000 é derivado do tamanho da memória da configuração de capacidade máxima.

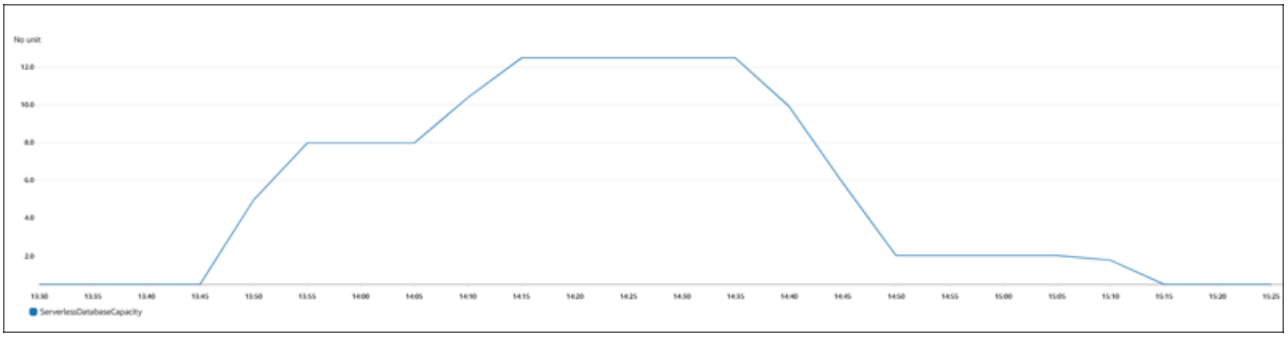
```
mysql> select @@innodb_buffer_pool_size / pow(2,20) as mebibytes, @@max_connections;
+-----------+-------------------+
| mebibytes | @@max_connections |
+-----------+-------------------+
|       672 |              5000 |
+-----------+-------------------+
1 row in set (0.00 sec)
```

Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora PostgreSQL

Os exemplos de CLI a seguir mostram como atualizar o intervalo de ACU para instâncias de banco de dados do Aurora Serverless v2 em um cluster existente do Aurora PostgreSQL.

1. O intervalo de capacidade do cluster começa em 0,5 a 1 ACU.
2. Altere o intervalo de capacidade para 8 a 32 ACUs.
3. Altere o intervalo de capacidade para 12,5 a 80 ACUs.
4. Altere o intervalo de capacidade para 0,5 a 128 ACUs.
5. Retorne a capacidade para seu intervalo inicial de 0,5 a 1 ACU.

A figura a seguir mostra as alterações de capacidade no Amazon CloudWatch.



A instância de banco de dados está ociosa e sua escala é reduzida verticalmente para 0,5 ACUs. As configurações relacionadas à capacidade a seguir se aplicam à instância de banco de dados neste momento.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Depois, alteramos o intervalo de capacidade do cluster. Após o término do comando `modify-db-cluster`, o intervalo de ACU do cluster é de 8,0 a 32.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

A alteração do intervalo de capacidade causa alterações nos valores padrão de alguns parâmetros de configuração. O Aurora pode aplicar alguns desses novos padrões imediatamente. No entanto, algumas das alterações de parâmetros são aplicadas somente após a reinicialização. O status do `pending-reboot` indica que é necessária uma reinicialização para aplicar algumas alterações de parâmetro.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

Neste ponto, o cluster está ocioso e a instância de banco de dados `serverless-v2-instance-1` está consumindo 8,0 ACUs. O parâmetro `shared_buffers` já está ajustado com base na capacidade atual da instância de banco de dados. O parâmetro `max_connections` ainda reflete o valor da capacidade máxima anterior. A redefinição desse valor exige a reinicialização da instância de banco de dados.

> **Note**
>
> Se você definir o parâmetro `max_connections` diretamente em um grupo de parâmetros de banco de dados personalizado, nenhuma reinicialização será necessária.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Reinicializamos a instância de banco de dados e aguardamos até que ela fique disponível novamente.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Agora que a instância de banco de dados foi reinicializada, o status `pending-reboot` é apagado. O valor `in-sync` confirma que o Aurora aplicou todas as alterações de parâmetros pendentes.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Após a reinicialização, `max_connections` mostra o valor da nova capacidade máxima.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Depois, alteramos o intervalo de capacidade do cluster para 12,5 a 80 ACUs.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
```

```
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

Neste ponto, o cluster está ocioso e a instância de banco de dados `serverless-v2-instance-1` está consumindo 12,5 ACUs. O parâmetro `shared_buffers` já está ajustado com base na capacidade atual da instância de banco de dados. O valor `max_connections` ainda é 5.000.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Nós reinicializamos novamente, mas os valores dos parâmetros permanecem os mesmos. Isso ocorre porque `max_connections` tem um valor máximo de 5.000 para um cluster de banco de dados Aurora Serverless v2 executando o Aurora PostgreSQL.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Agora definimos o intervalo de capacidade de 0,5 a 128 ACUs. A escala do cluster de banco de dados é reduzida verticalmente para 10 ACUs e depois para 2. Reinicializamos a instância de banco de dados.

```
postgres=> show max_connections;
 max_connections
-----------------
 2000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

O valor de `max_connections` para instâncias de banco de dados do Aurora Serverless v2 é baseado no tamanho da memória derivado do máximo de ACUs. No entanto, quando você especifica uma capacidade mínima de 0 ou 0,5 ACU em instâncias de banco de dados compatíveis com o PostgreSQL, o valor máximo de `max_connections` é limitado a 2.000.

Agora, retornamos a capacidade ao intervalo inicial de 0,5 a 1 ACU e reinicializamos a instância de banco de dados. O parâmetro `max_connections` retornou ao seu valor original.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Trabalhar com grupos de parâmetros para o Aurora Serverless v2

Quando você cria o seu cluster de banco de dados do Aurora Serverless v2, você escolhe um mecanismo de banco de dados do Aurora específico e um grupo de parâmetros de cluster de banco de dados associado. Se você não estiver familiarizado com a forma como o Aurora usa grupos

de parâmetros para aplicar configurações de forma consistente em clusters, consulte Grupos de parâmetros para Amazon Aurora. Todos esses procedimentos para criar, modificar, aplicar e realizar outras ações para grupos de parâmetros se aplicam ao Aurora Serverless v2.

O recurso de grupo de parâmetros geralmente funciona da mesma forma entre clusters provisionados e clusters que contenham instâncias de banco de dados do Aurora Serverless v2:

- Os valores de parâmetros de cluster padrão para todas as instâncias de banco de dados no cluster são definidos pelo grupo de parâmetros.
- É possível substituir alguns parâmetros para instâncias de banco de dados específicas determinando um grupo de parâmetros de banco de dados personalizado para elas. Você pode fazer isso durante a depuração ou o ajuste de performance para instâncias de banco de dados específicas. Por exemplo, digamos que você tenha um cluster que contenha algumas instâncias de banco de dados do Aurora Serverless v2 e algumas instâncias de banco de dados provisionadas. Nesse caso, você pode especificar alguns parâmetros diferentes para as instâncias de banco de dados provisionadas usando um grupo de parâmetros de banco de dados personalizado.
- No caso do Aurora Serverless v2, é possível usar todos os parâmetros que tenham o valor `provisioned` no atributo `SupportedEngineModes` no grupo de parâmetros. No Aurora Serverless v1, você só pode usar o subconjunto de parâmetros que tem `serverless` no atributo `SupportedEngineModes`.

Tópicos

- Valores de parâmetro padrão
- Conexões máximas do Aurora Serverless v2
- Parâmetros ajustados pelo Aurora à medida que a escala do Aurora Serverless v2 é aumentada ou reduzida na vertical
- Parâmetros calculados pelo Aurora com base na capacidade máxima do Aurora Serverless v2

Valores de parâmetro padrão

A diferença crucial entre instâncias de banco de dados provisionadas e as instâncias de banco de dados do Aurora Serverless v2 é que o Aurora substitui todos os valores de parâmetro personalizados por determinados parâmetros relacionados à capacidade da instância de banco de dados. Os valores de parâmetros personalizados ainda se aplicam a qualquer instância de banco de dados provisionada em seu cluster. Para obter mais detalhes sobre como as instâncias de banco de dados do Aurora Serverless v2 interpretam os parâmetros dos grupos de parâmetros do

Aurora, consulte Parâmetros de configuração para clusters do Aurora. Para saber quais parâmetros específicos são substituídos pelo Aurora Serverless v2, consulte Parâmetros ajustados pelo Aurora à medida que a escala do Aurora Serverless v2 é aumentada ou reduzida na vertical e Parâmetros calculados pelo Aurora com base na capacidade máxima do Aurora Serverless v2.

Você pode obter uma lista de valores padrão para os grupos de parâmetros padrão dos vários mecanismos de banco de dados do Aurora usando o comando da CLI describe-db-cluster-parameters e consultando da Região da AWS. Veja a seguir os valores que você pode usar para as opções `--db-parameter-group-family` e `-db-parameter-group-name` para versões de mecanismo compatíveis com o Aurora Serverless v2.

Mecanismo de banco de dados e versão	Família de grupos de parâmetros	Nome do grupo de parâmetros padrão
Aurora MySQL versão 3	`aurora-mysql8.0`	`default.aurora-mysql8.0`
Aurora PostgreSQL versão 13.x	`aurora-postgresql13`	`default.aurora-postgresql13`
Aurora PostgreSQL versão 14.x	`aurora-postgresql14`	`default.aurora-postgresql14`
Aurora PostgreSQL versão 15.x	`aurora-postgresql15`	`default.aurora-postgresql15`
Aurora PostgreSQL versão 16.x	`aurora-postgresql16`	`default.aurora-postgresql16`
Aurora PostgreSQL versão 17.x	`aurora-postgresql17`	`default.aurora-postgresql17`

O exemplo a seguir obtém uma lista de parâmetros do grupo de clusters de banco de dados padrão para o Aurora MySQL versão 3 o Aurora PostgreSQL 13. Estas são as versões do Aurora MySQL e do Aurora PostgreSQL utilizadas com o Aurora Serverless v2.

Para Linux, macOS ou Unix:

```
aws rds describe-db-cluster-parameters \
```

```
  --db-cluster-parameter-group-name default.aurora-mysql8.0 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-postgresql13 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text
```

Para Windows:

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-mysql8.0 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text

aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-postgresql13 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

Conexões máximas do Aurora Serverless v2

No caso do Aurora MySQL e do Aurora PostgreSQL, as instâncias de banco de dados do Aurora Serverless v2 têm o parâmetro `max_connections` constante para que as conexões não sejam desativadas quando a escala da instância de banco de dados é reduzida na vertical. O valor padrão para esse parâmetro é derivado de uma fórmula com base no tamanho da memória da instância de banco de dados. Para obter detalhes sobre a fórmula e os valores padrão para classes de instâncias de banco de dados provisionadas, consulte Número máximo de conexões com uma instância de bancos de dados Aurora MySQL e Número máximo de conexões com uma instância de bancos de dados Aurora PostgreSQL.

Quando o Aurora Serverless v2 avalia a fórmula, ele usa o tamanho da memória com base nas unidades de capacidade máxima do Aurora (ACUs) para a instância de banco de dados, não no

valor atual de ACU. Se você alterar o valor padrão, recomendamos usar uma variação da fórmula em vez de especificar um valor constante. Dessa forma, o Aurora Serverless v2 pode usar uma configuração apropriada com base na capacidade máxima.

Ao alterar a capacidade máxima de um cluster de banco de dados Aurora Serverless v2, você precisa reinicializar as instâncias de banco de dados Aurora Serverless v2 para atualizar o valor `max_connections`. Isso ocorre porque `max_connections` é um parâmetro estático do Aurora Serverless v2.

A tabela a seguir mostra os valores padrão de `max_connections` para o Aurora Serverless v2 com base no valor máximo da ACU.

ACUs máximas	Conexões máximas padrão no Aurora MySQL	Conexões máximas padrão no Aurora PostgreSQL
1	90	189
4	135	823
8	1.000	1.669
16	2.000	3.360
32	3.000	5.000
64	4.000	5.000
128	5.000	5.000
192	6.000	5.000
256	6.000	5.000

> **Note**
>
> O valor de `max_connections` para instâncias de banco de dados do Aurora Serverless v2 é baseado no tamanho da memória derivado do máximo de ACUs. No entanto, quando você especifica uma capacidade mínima de 0 ou 0,5 ACU em instâncias de banco de dados compatíveis com o PostgreSQL, o valor máximo de `max_connections` é limitado a 2.000.

Para exemplos específicos que mostram como o `max_connections` muda com o valor máximo da ACU, consulte Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora MySQL e Exemplo: Alterar o intervalo de capacidade do Aurora Serverless v2 de um cluster do Aurora PostgreSQL.

Parâmetros ajustados pelo Aurora à medida que a escala do Aurora Serverless v2 é aumentada ou reduzida na vertical

Durante a escalabilidade automática, o Aurora Serverless v2 precisa ser capaz de alterar os parâmetros para que cada instância de banco de dados funcione melhor com a capacidade aumentada ou reduzida. Dessa forma, não é possível substituir alguns parâmetros relacionados à capacidade. Para alguns parâmetros que possam ser substituídos, evite a codificação de forma rígida. As considerações a seguir se aplicam a essas configurações relacionadas à capacidade.

No caso do Aurora MySQL, o Aurora Serverless v2 redimensiona alguns parâmetros dinamicamente durante a escalabilidade. No caso dos parâmetros a seguir, o Aurora Serverless v2 não usa nenhum valor de parâmetro personalizado que seja especificado:

- `innodb_buffer_pool_size`
- `innodb_purge_threads`
- `table_definition_cache`
- `table_open_cache`

No caso do Aurora PostgreSQL, o Aurora Serverless v2 redimensiona os parâmetros a seguir dinamicamente durante a escalabilidade. No caso dos parâmetros a seguir, o Aurora Serverless v2 não usa nenhum valor de parâmetro personalizado que seja especificado:

- `shared_buffers`

No caso de todos os parâmetros diferentes dos listados aqui, as instâncias de banco de dados do Aurora Serverless v2 funcionam da mesma forma que as instâncias de banco de dados provisionadas. O valor padrão para o parâmetro é herdado do grupo de parâmetros de cluster. É possível modificar o valor padrão para todo o cluster usando um grupo de parâmetros de cluster personalizado. Você também pode modificar o valor padrão para determinadas instâncias de banco de dados usando um grupo de parâmetros de banco de dados personalizado. Os parâmetros dinâmicos são atualizados imediatamente. As alterações nos parâmetros estáticos só são aplicadas após a reinicialização da instância do banco de dados.

Parâmetros calculados pelo Aurora com base na capacidade máxima do Aurora Serverless v2

No caso dos parâmetros a seguir, o Aurora PostgreSQL usa valores padrão que são derivados do tamanho da memória com base na configuração máxima de ACU, da mesma forma que ocorre com o `max_connections`:

- `autovacuum_max_workers`
- `autovacuum_vacuum_cost_limit`
- `autovacuum_work_mem`
- `effective_cache_size`
- `maintenance_work_mem`

Evitar erros de falta de memória

Se uma de suas instâncias de banco de dados do Aurora Serverless v2 atinge de forma consistente o limite de sua capacidade máxima, o Aurora indica essa condição definindo o status da instância de banco de dados como `incompatible-parameters`. Enquanto a instância de banco de dados apresentar o status `incompatible-parameters`, algumas operações ficarão bloqueadas. Por exemplo, não é possível atualizar a versão do mecanismo.

Normalmente, sua instância de banco de dados entra nesse status quando ela é reiniciada com frequência devido a erros de falta de memória. O Aurora registra um evento quando esse tipo de reinicialização acontece. Você pode visualizar o evento seguindo o procedimento em Visualizar eventos do Amazon RDS. O uso de memória excepcionalmente alto pode ocorrer devido à sobrecarga causada pela ativação de configurações como Performance Insights e autenticação do IAM. A causa também pode ser uma workload pesada em sua instância de banco de dados ou o gerenciamento dos metadados associados a um grande número de objetos de esquema.

Se a pressão da memória se tornar menor para que a instância de banco de dados não atinja sua capacidade máxima com muita frequência, o Aurora alterará automaticamente o status da instância de banco de dados de volta para `available`.

Para se recuperar dessa condição, você pode realizar algumas ou todas as seguintes ações:

- Aumente o limite inferior de capacidade para as instâncias de banco de dados do Aurora Serverless v2 alterando o valor mínimo da unidade de capacidade (ACU) do Aurora para o cluster. Isso evita problemas em que um banco de dados ocioso tem a escala reduzida na vertical para

uma capacidade com menos memória do que o necessário para os recursos que estão ativados no cluster. Depois de alterar as configurações de ACU do cluster, reinicie a instância de banco de dados do Aurora Serverless v2. Com esse procedimento, você avalia se o Aurora consegue redefinir o status de volta como `available`.

- Aumente o limite superior de capacidade das instâncias de banco de dados do Aurora Serverless v2 alterando o valor máximo da unidade de capacidade (ACU) do Aurora para o cluster. Isso evita problemas em que não é possível aumentar a escala de um banco de dados ocupado na vertical até uma capacidade com memória suficiente para os recursos ativados no cluster e na workload do banco de dados. Depois de alterar as configurações de ACU do cluster, reinicie a instância de banco de dados do Aurora Serverless v2. Com esse procedimento, você avalia se o Aurora consegue redefinir o status de volta como `available`.
- Desative as configurações que exigem sobrecarga de memória. Por exemplo, digamos que você tenha recursos, como o AWS Identity and Access Management (IAM), o Performance Insights ou a replicação de logs binários do Aurora MySQL ativados, mas não os use. Se esse for o caso, você pode desativá-los. Também é possível ajustar os valores de capacidade mínima e máxima do cluster para valores mais altos com o objetivo de levar em conta a memória utilizada por esses recursos. Para obter as diretrizes sobre como escolher configurações de capacidade mínima e máxima, consulte escolher o intervalo de capacidade do Aurora Serverless v2 para um cluster do Aurora.
- Reduza a workload na instância de banco de dados. Por exemplo, você pode adicionar instâncias de banco de dados de leitor ao cluster para distribuir a carga de consultas somente leitura por mais instâncias de banco de dados.
- Ajuste o código SQL usado pela aplicação para usar menos recursos. Por exemplo, você pode examinar seus planos de consulta, conferir o log de consultas lento ou ajustar os índices em suas tabelas. Também é possível realizar outros tipos tradicionais de ajuste SQL.

Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2

Para começar a usar o Amazon CloudWatch para sua instância de banco de dados do Aurora Serverless v2, consulte Visualizar logs do Aurora Serverless v2 no Amazon CloudWatch. Para saber mais sobre como monitorar clusters de Aurora banco de dados por meio de CloudWatch, consulte Monitorar eventos de log no Amazon CloudWatch.

É possível visualizar suas instâncias de banco de dados do Aurora Serverless v2 no CloudWatch para monitorar a capacidade consumida por cada instância de banco de dados com a métrica `ServerlessDatabaseCapacity`. Você também pode monitorar todas as métricas padrão do

CloudWatch para o Aurora, como `DatabaseConnections` e `Queries`. Para ver a lista completa de métricas do CloudWatch que é possível monitorar para o Aurora, consulte Métricas do Amazon CloudWatch para o Amazon Aurora. As métricas são divididas em métricas em nível de cluster e em nível de instância, em Métricas no nível do cluster do Amazon Aurora e Métricas no nível da instância do Amazon Aurora.

É importante monitorar as métricas de nível de instância do CloudWatch a seguir para que você compreenda como suas instâncias de banco de dados do Aurora Serverless v2 estão sendo expandidas ou reduzidas. Todas essas métricas são calculadas a cada segundo. Dessa forma, você pode monitorar o status atual das instâncias de banco de dados do Aurora Serverless v2. É possível definir alarmes para notificá-lo se alguma instância de banco de dados do Aurora Serverless v2 se aproximar de um limite das métricas relacionadas à capacidade. Você pode determinar se as configurações de capacidade mínima e máxima são apropriadas ou se você precisa ajustá-las. É possível determinar onde concentrar seus esforços para otimizar a eficiência de seu banco de dados.

- `ServerlessDatabaseCapacity`. como métrica específica da instância, ela relata o número de ACUs representadas pela capacidade atual da instância de banco de dados. Como métrica específica do cluster, ela representa a média dos valores de `ServerlessDatabaseCapacity` de todas as instâncias de banco de dados do Aurora Serverless v2 no cluster. Essa métrica é apenas uma métrica específica do cluster no Aurora Serverless v1. No Aurora Serverless v2, está disponível no nível da instância de banco de dados e no nível do cluster.
- `ACUUtilization`. Essa métrica é nova no Aurora Serverless v2. Esse valor é representado como uma porcentagem. É calculado como o valor da métrica `ServerlessDatabaseCapacity` dividida pelo valor máximo de ACU do cluster de banco de dados. Considere as seguintes diretrizes para interpretar essa métrica e agir:
  - Se essa métrica se aproximar de um valor de `100.0`, a escala da instância de banco de dados foi aumentada na vertical ao nível mais elevado possível. Considere aumentar a configuração máxima de ACU para o cluster. Dessa forma, as instâncias de banco de dados do gravador e do leitor podem ser escaladas para uma capacidade maior.
  - Suponha que uma workload somente leitura faça com que uma instância de banco de dados do leitor se aproxime de um `ACUUtilization` de `100.0`, enquanto a instância de banco de dados do gravador não está próxima de sua capacidade máxima. Nesse caso, considere adicionar mais instâncias de banco de dados de leitor ao cluster. Dessa forma, você pode distribuir a parte somente leitura da workload por mais instâncias de banco de dados, reduzindo a carga em cada instância de banco de dados do leitor.

  - Suponha que você esteja executando uma aplicação de produção, em que a performance e a escalabilidade sejam as principais considerações. Nesse caso, é possível definir o valor máximo de ACU para o cluster como um número elevado. Seu objetivo é que a métrica `ACUUtilization` permaneça sempre abaixo de `100.0`. Com um alto valor máximo de ACU, você pode ter certeza de que há espaço suficiente caso haja picos inesperados na atividade do banco de dados. Você só será cobrado pela capacidade do banco de dados que for realmente consumida.

- `CPUUtilization`. Essa métrica no Aurora Serverless v2 é interpretada de modo diferente do que é em instâncias de banco de dados provisionadas. No caso do Aurora Serverless v2, esse valor é uma porcentagem calculada como a quantidade de CPU que está sendo usada atualmente dividida pela capacidade da CPU disponível abaixo do valor máximo de ACU do cluster de banco de dados. O Aurora monitora esse valor automaticamente e aumenta a escala de sua instância de banco de dados do Aurora Serverless v2 na vertical quando ela usa consistentemente uma alta proporção de sua capacidade de CPU.

  Se essa métrica se aproximar de um valor de `100.0`, a instância de banco de dados atingiu sua capacidade máxima de CPU. Considere aumentar a configuração máxima de ACU para o cluster. Se essa métrica se aproximar de um valor de `100.0` em uma instância de banco de dados do leitor, considere adicionar mais instâncias de banco de dados do leitor ao cluster. Dessa forma, você pode distribuir a parte somente leitura da workload por mais instâncias de banco de dados, reduzindo a carga em cada instância de banco de dados do leitor.

- `FreeableMemory`. Esse valor representa a quantidade de memória não utilizada que está disponível quando a instância de banco de dados do Aurora Serverless v2 é escalada para sua capacidade máxima. Para cada ACU em que a capacidade atual está abaixo da capacidade máxima, esse valor aumenta em aproximadamente 2 GiB. Assim, essa métrica não se aproxima de zero até que a escala da instância de banco de dados seja aumentada na vertical ao nível mais alto possível.

  Se essa métrica se aproximar de um valor de `0`, a escala da instância de banco de dados foi aumentada na vertical ao nível mais alto possível e está se aproximando do limite de sua memória disponível. Considere aumentar a configuração máxima de ACU para o cluster. Se essa métrica se aproximar de um valor de `0` em uma instância de banco de dados do leitor, considere adicionar mais instâncias de banco de dados do leitor ao cluster. Dessa forma, é possível distribuir a parte somente leitura da workload por mais instâncias de banco de dados, reduzindo o uso de memória em cada instância de banco de dados do leitor.

- `TempStorageIOPS`. O número de IOPS realizadas no armazenamento local anexado à instância de banco de dados. Ele inclui as IOPS para leituras e gravações. Essa métrica representa uma contagem e é medida uma vez por segundo. É uma nova métrica do Aurora Serverless v2. Para obter detalhes, consulte Métricas no nível da instância do Amazon Aurora.
- `TempStorageThroughput`. A quantidade de dados transferidos entre o armazenamento local associado e a instância de banco de dados. Essa métrica é apresentada em bytes e é medida uma vez por segundo. É uma nova métrica do Aurora Serverless v2. Para obter detalhes, consulte Métricas no nível da instância do Amazon Aurora.

Normalmente, a maioria das expansões de instâncias de banco de dados do Aurora Serverless v2 é ocasionada pelo uso de memória e atividade da CPU. As métricas `TempStorageIOPS` e `TempStorageThroughput` podem ajudar você a diagnosticar os casos raros em que a atividade de rede para transferências entre sua instância de banco de dados e dispositivos de armazenamento local é responsável por aumentos inesperados de capacidade. Para monitorar outras atividades de rede, você pode usar estas métricas existentes:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

É possível fazer com que o Aurora publique alguns ou todos os logs de banco de dados no Amazon CloudWatch Logs. Consulte as seguintes instruções, dependendo do mecanismo de banco de dados:

- the section called “Publicar logs do Aurora PostgreSQL no CloudWatch Logs”
- the section called “Publicar logs do Aurora MySQL no CloudWatch Logs”

Como as métricas Aurora Serverless v2 se aplicam à sua fatura da AWS

As cobranças do Aurora Serverless v2 em sua fatura da AWS são calculadas com base na mesma métrica do `ServerlessDatabaseCapacity` que você pode monitorar. O mecanismo de faturamento pode diferir da média computada do CloudWatch para essa métrica nos casos em

que você usa a capacidade do Aurora Serverless v2 por somente parte de uma hora. Também pode ser diferente se os problemas do sistema deixarem a métrica do CloudWatch indisponível por breves períodos. Dessa forma, talvez você veja em sua fatura um valor um pouco diferente de horas de ACU do que o exibido se você mesmo calcular o número a partir do valor médio de `ServerlessDatabaseCapacity`.

Exemplos de comandos do CloudWatch para métricas do Aurora Serverless v2

Os exemplos da AWS CLI a seguir demonstram como monitorar as métricas mais importantes do CloudWatch relacionadas ao Aurora Serverless v2. Em cada caso, substitua a string `Value=` do parâmetro `--dimensions` pelo identificador de sua própria instância de banco de dados do Aurora Serverless v2.

O exemplo do Linux a seguir exibe os valores de capacidade mínima, máxima e média de uma instância de banco de dados, medidos a cada dez minutos em uma hora. O comando `date` do Linux especifica as horas de início e término em relação à data e hora atuais. A função `sort_by` no parâmetro `--query` classifica os resultados cronologicamente com base no campo `Timestamp`.

```
aws cloudwatch get-metric-statistics --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Os exemplos do Linux a seguir demonstram o monitoramento da capacidade de cada instância de banco de dados em um cluster. Eles medem a utilização mínima, máxima e média da capacidade de cada instância de banco de dados. As medições são realizadas uma vez por hora durante um período de três horas. Esses exemplos usam a métrica `ACUUtilization` que representa uma porcentagem do limite superior em ACUs, em vez de `ServerlessDatabaseCapacity` representando um número fixo de ACUs. Dessa forma, não é necessário conhecer os números reais para os valores mínimo e máximo de ACU no intervalo de capacidade. É possível ver porcentagens que variam de 0 a 100.

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_writer_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_reader_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

O exemplo do Linux a seguir realiza medições semelhantes às anteriores. Nesse caso, as medidas são para a métrica `CPUUtilization`. As medições são realizadas a cada dez minutos durante um período de uma hora. Os números representam a porcentagem de CPU disponível usada, com base nos recursos da CPU disponíveis para a configuração de capacidade máxima para a instância de banco de dados.

```
aws cloudwatch get-metric-statistics --metric-name "CPUUtilization" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

O exemplo do Linux a seguir realiza medições semelhantes às anteriores. Nesse caso, as medidas são para a métrica `FreeableMemory`. As medições são realizadas a cada dez minutos durante um período de uma hora.

```
aws cloudwatch get-metric-statistics --metric-name "FreeableMemory" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Monitorar a performance do Aurora Serverless v2 com o Performance Insights

É possível usar o Performance Insights para monitorar a performance das instâncias de banco de dados do Aurora Serverless v2. Para saber os procedimentos do Performance Insights, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

Os novos contadores do Performance Insights a seguir se aplicam a instâncias de banco de dados do Aurora Serverless v2:

- `os.general.serverlessDatabaseCapacity`: a capacidade atual da instância de banco de dados em ACUs. O valor corresponde à métrica `ServerlessDatabaseCapacity` do CloudWatch para a instância de banco de dados.
- `os.general.acuUtilization`: a porcentagem de capacidade atual da capacidade máxima configurada. O valor corresponde à métrica `ACUUtilization` do CloudWatch para a instância de banco de dados.
- `os.general.maxConfiguredAcu`: a capacidade máxima que você configurou para essa instância de banco de dados do Aurora Serverless v2. É medida em ACUs.
- `os.general.minConfiguredAcu`: a capacidade mínima que você configurou para essa instância de banco de dados do Aurora Serverless v2. É medida em ACUs.

Para ver a lista completa de contadores do Performance Insights, consulte Métricas de contadores do Performance Insights.

Quando valores `vCPU` são mostrados para uma instância de banco de dados do Aurora Serverless v2 no Performance Insights, esses valores representam estimativas com base no valor de ACU para a instância de banco de dados. No intervalo padrão de um minuto, todos os valores de vCPU fracionários são arredondados para o número inteiro mais próximo. Para intervalos mais longos, o valor de vCPU mostrado é a média dos valores inteiros de vCPU para cada minuto.

Solução de problemas de capacidade do Aurora Serverless v2

Em alguns casos, a escala do Aurora Serverless v2 não é reduzida verticalmente para a capacidade mínima, mesmo sem carga no banco de dados. Isso pode acontecer por um dos seguintes motivos.

- Determinados recursos podem aumentar o uso de recursos e impedir que a escala do banco de dados seja reduzida verticalmente para a capacidade mínima. Esses recursos incluem o seguinte:
  - Bancos de dados globais do Aurora
  - Exportar logs do CloudWatch
  - Habilitar `pg_audit` em clusters de banco de dados compatíveis com o Aurora PostgreSQL
  - Monitoramento avançado
  - Insights de Performance

Para obter mais informações, consulte Escolher a configuração de capacidade mínima do Aurora Serverless v2 para um cluster.

- Se a escala de uma instância de leitor não estiver sendo reduzida verticalmente ao mínimo e permanecer com a mesma capacidade ou maior que a instância do gravador, confira o nível de prioridade da instância do leitor. As instâncias de banco de dados do leitor do Aurora Serverless v2 no nível 0 ou 1 são mantidas em uma capacidade mínima pelo menos tão alta quanto a instância de banco de dados do gravador. Altere o nível de prioridade do leitor para 2 ou superior para que ele aumente e reduza a escala verticalmente independentemente do gravador. Para ter mais informações, consulte Escolher o nível de promoção para um leitor do Aurora Serverless v2.
- Defina os parâmetros do banco de dados que afetam o tamanho da memória compartilhada como seus valores padrão. Definir um valor maior do que o padrão aumenta a necessidade de memória compartilhada e evita que a escala do banco de dados seja reduzida verticalmente para a capacidade mínima. Os exemplos são `max_connections` e `max_locks_per_transaction`.

> **Note**
>
> Atualizar parâmetros de memória compartilhada exige que o banco de dados seja reiniciado para que as alterações entrem em vigor.

- Workloads de banco de dados pesadas podem aumentar o uso de recursos.
- Grandes volumes de banco de dados podem aumentar o uso de recursos.

  O Amazon Aurora usa recursos de memória e CPU para o gerenciamento de clusters de banco de dados. O Aurora exige mais CPU e memória para gerenciar clusters de banco de dados com volumes maiores de banco de dados. Se a capacidade mínima do cluster for menor do que a mínima exigida para o gerenciamento do cluster, a escala do cluster não será reduzida verticalmente para a capacidade mínima.
- Processos em segundo plano, como a limpeza, também podem aumentar o uso de recursos.
- As limitações da versão da plataforma podem afetar os recursos de ajuste de escala. O intervalo de ajuste de escala disponível para determinado cluster é influenciada tanto pela versão do mecanismo quanto pelo hardware (versão da plataforma). É possível ter uma versão de mecanismo mais capaz em execução em uma versão da plataforma menos capaz e vice-versa.

Se a escala do banco de dados ainda não for reduzida verticalmente até a capacidade mínima configurada, pare e reinicie o banco de dados para recuperar quaisquer fragmentos de memória que

possam ter se acumulado ao longo do tempo. Interromper e iniciar um banco de dados ocasiona tempo de inatividade, por isso recomendamos fazer isso com moderação.

Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2

Se as instâncias de banco de dados do Aurora Serverless v2 não tiverem conexões iniciadas por atividade do usuário dentro de um período especificado, é possível especificar que elas reduzam a escala verticalmente para zero ACU e pausem automaticamente. É possível fazer isso especificando um valor mínimo de ACU de zero para seu cluster de banco de dados. Não há cobranças pela capacidade da instância enquanto ela estiver no estado de pausa. Habilitar o recurso de pausa e retomada automáticas (pausa automática) em clusters do Aurora que são pouco usados ou têm longos períodos de inatividade pode ajudar no gerenciamento de custos da frota de bancos de dados.

> **Note**
>
> O recurso de pausa automática está disponível no Aurora Serverless v2 com Aurora PostgreSQL e Aurora MySQL. Talvez seja necessário atualizar a versão do mecanismo de banco de dados do Aurora para aproveitar esse recurso. Consulte Aurora Serverless v2Capacidade do para ver as versões do mecanismo com disponibilidade de capacidade mínima de 0 ACUs.

Tópicos

- Visão geral do recurso de pausa automática do Aurora Serverless v2
- Pré-requisitos e limitações do recurso de pausa automática do Aurora Serverless v2
- Ativar e desativar o recurso de pausa automática
- Como funciona o recurso de pausa automática do Aurora Serverless v2
- Como a pausa automática do Aurora Serverless v2 funciona para diferentes tipos de clusters do Aurora
- Monitorar clusters do Aurora que usam a pausa automática
- Solucionar problemas da pausa automática do Aurora Serverless v2
- Considerações sobre o design da aplicação para o recurso de pausa automática do Aurora Serverless v2

Visão geral do recurso de pausa automática do Aurora Serverless v2

As instâncias de banco de dados do Aurora Serverless v2 podem pausar automaticamente após um período sem conexões de usuário e retomar automaticamente quando chega uma solicitação de conexão. O recurso de pausa/retomada automática do Aurora Serverless v2 ajuda no gerenciamento dos custos de sistemas que não têm um objetivo de nível de serviço (SLO) rigoroso. Por exemplo, você pode habilitar esse recurso em clusters usados para desenvolvimento e testes ou para aplicações internas em que uma breve pausa é aceitável enquanto o banco de dados é retomado. Se a workload tiver períodos de inatividade e puder tolerar pequenos atrasos na conexão enquanto a instância é retomada, considere usar a pausa automática nas instâncias do Aurora Serverless v2 para reduzir custos.

É possível controlar esse comportamento especificando se as instâncias de banco de dados do Aurora Serverless v2 em um cluster podem ou não pausar automaticamente e por quanto tempo cada instância deve ficar ociosa antes de pausar. Para habilitar o comportamento de pausa automática para todas as instâncias de banco de dados do Aurora Serverless v2 em um cluster do Aurora, defina o valor mínimo da capacidade do cluster para zero ACUs.

Se você anteriormente aproveitou o recurso do Aurora Serverless v1 que escalava para zero ACUs após um período de inatividade, você pode atualizar para o Aurora Serverless v2 e usar seu recurso de pausa automática correspondente.

Os benefícios da economia de custos do recurso de pausa automática são semelhantes ao uso do recurso interromper/iniciar cluster. A pausa automática no Aurora Serverless v2 tem os benefícios adicionais de uma retomada mais rápida do que iniciar um cluster interrompido e da automatização do processo de determinação de quando pausar e retomar cada instância de banco de dados.

O recurso de pausa automática também fornece granularidade adicional no controle dos custos dos recursos computacionais no cluster. É possível habilitar algumas instâncias de leitura para pausarem mesmo enquanto a instância de gravação e outros leitores no cluster permanecem ativos o tempo todo. Para fazer isso, atribua às instâncias de leitura que podem pausar independentemente de outras instâncias uma prioridade de failover no intervalo de 2 a 15.

As instâncias de gravação e todas as instâncias de leitura com prioridade de failover 0 e 1 sempre pausam e retomam ao mesmo tempo. Assim, as instâncias desse grupo pausam depois que nenhuma delas tem conexões no intervalo de tempo especificado.

Os clusters de banco de dados do Aurora podem conter uma combinação de instâncias de banco de dados de gravação e leitura e instâncias de banco de dados do Aurora Serverless v2

e provisionadas. Portanto, para usar esse recurso de forma eficaz, é útil entender os seguintes aspectos do mecanismo de pausa automática:

- As circunstâncias nas quais uma instância de banco de dados pode ser pausada automaticamente.
- Quando uma instância de banco de dados pode ser impedida de pausar. Por exemplo, habilitar alguns recursos do Aurora ou realizar determinados tipos de operações no cluster pode impedir que as instâncias pausem, mesmo sem nenhuma conexão com essas instâncias.
- As consequências para o monitoramento e o faturamento enquanto uma instância está pausada.
- Quais ações fazem com que uma instância de banco de dados retome o processamento.
- Como a capacidade de uma instância de banco de dados muda em torno dos eventos de pausa e retomada.
- Como controlar o intervalo de inatividade antes da pausa de uma instância de banco de dados.
- Como codificar a lógica da aplicação para lidar com o período em que uma instância de banco de dados está retomando o processamento.

Pré-requisitos e limitações do recurso de pausa automática do Aurora Serverless v2

Antes de usar o recurso de pausa automática, verifique quais versões do mecanismo usar. Além disso, verifique se a pausa automática funciona se combinada com os outros recursos do Aurora que você pretende usar. Não é possível ativar a pausa automática se você estiver usando uma versão do mecanismo que não oferece suporte ao recurso. Você não receberá nenhum erro se usar recursos incompatíveis em combinação com a pausa automática. Se o cluster estiver usando recursos ou configurações incompatíveis, as instâncias do Aurora Serverless v2 não serão pausadas automaticamente.

- Se você estiver usando o Aurora PostgreSQL, o mecanismo de banco de dados deve estar executando pelo menos a versão 16.3, 15.7, 14.12 ou 13.15.
- Se você estiver usando o Aurora MySQL, o mecanismo de banco de dados deve estar executando a versão 3.08.0 ou superior.
- Para obter uma lista completa de versões de mecanismo e as regiões da AWS nas quais o recurso está disponível, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

- Quando uma instância do Aurora Serverless v2 é retomada, sua capacidade pode ser menor do que quando a instância foi pausada. Para obter detalhes, consulte Diferenças no comportamento de pausa automática entre o Aurora Serverless v2 e o Aurora Serverless v1.

Certas condições ou configurações impedem que as instâncias do Aurora Serverless v2 sejam pausadas automaticamente. Para ter mais informações, consulte Situações em que o Aurora Serverless v2 não faz pausas automáticas.

Ativar e desativar o recurso de pausa automática

É possível ativar e desativar o recurso de pausa automática no nível do cluster. Para fazer isso, use os mesmos procedimentos usados ao ajustar as capacidades mínima e máxima do cluster. O recurso de pausa automática é representado por uma capacidade mínima de 0 ACUs.

Tópicos

- Ativar a pausa automática para instâncias do Aurora Serverless v2 em um cluster
- Intervalo de tempo limite de pausa automática configurável do Aurora Serverless v2
- Retomar uma instância do Aurora Serverless v2 pausada automaticamente
- Desativar a pausa automática para instâncias do Aurora Serverless v2 em um cluster

Ativar a pausa automática para instâncias do Aurora Serverless v2 em um cluster

Siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster. Para a capacidade mínima, escolha 0 ACUs. Ao escolher uma capacidade mínima de 0 ACUs, você também pode especificar por quanto tempo a instância ficará ociosa antes de ser pausada automaticamente.

O exemplo de CLI a seguir mostra como criar um cluster do Aurora com o recurso de pausa automática habilitado e o intervalo de pausa automática definido para 10 minutos (600 segundos).

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0 \
    --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=4,SecondsUntilAutoPause=600 \
```

```
    --master-username myuser \
    --manage-master-user-password
```

O exemplo de CLI a seguir mostra como você pode ativar o recurso de pausa automática em um cluster existente do Aurora. Este exemplo define o intervalo de pausa automática como 1 hora (3.600 segundos).

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=80,SecondsUntilAutoPause=3600

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 0,
    "MaxCapacity": 80.0,
    "SecondsUntilAutoPause": 3600
}
```

Intervalo de tempo limite de pausa automática configurável do Aurora Serverless v2

O intervalo de tempo limite é representado no atributo `ServerlessV2ScalingConfiguration` do seu cluster do Aurora. É possível especificar esse intervalo no Console de gerenciamento da AWS ao criar ou modificar um cluster do Aurora, se a capacidade mínima estiver definida como zero ACUs. Você pode especificá-lo na AWS CLI usando o parâmetro `--serverless-v2-scaling-configuration` ao criar ou modificar um cluster do Aurora. Você pode especificá-lo na API do RDS usando o parâmetro `ServerlessV2ScalingConfiguration` ao criar ou modificar um cluster do Aurora.

O intervalo mínimo que pode ser definido é de 300 segundos (5 minutos). Esse é o padrão se nenhum intervalo for especificado. O intervalo máximo que pode ser definido é de 86.400 segundos (um dia).

Suponha que você desative o recurso de pausa automática de um cluster alterando a capacidade mínima do cluster para um valor diferente de zero. Nesse caso, a propriedade de intervalo é removida do atributo `ServerlessV2ScalingConfiguration`. A ausência dessa propriedade fornece uma confirmação extra de que o recurso de pausa automática está desativado para esse cluster. Se, posteriormente, você reativar a pausa automática, poderá especificar qualquer intervalo personalizado novamente naquele momento.

Retomar uma instância do Aurora Serverless v2 pausada automaticamente

- Ao se conectar a uma instância do Aurora Serverless v2 pausada, ela automaticamente é retomada e aceita a conexão.
- Uma tentativa de conexão que não inclui credenciais válidas ainda faz com que a instância de banco de dados seja retomada.
- Se você se conectar pelo endpoint do gravador, o Aurora retomará a instância de banco de dados do gravador se ela estiver pausada automaticamente. Ao mesmo tempo, o Aurora retoma todas as instâncias de leitura pausadas automaticamente que tenham prioridade de failover 0 ou 1, o que significa que sua capacidade está vinculada à capacidade da instância de gravação.
- Se você se conectar pelo endpoint do leitor, o Aurora escolherá uma instância do leitor aleatoriamente. Se essa instância do leitor estiver pausada, o Aurora a retomará. O Aurora também retoma primeiro a instância do gravador porque a instância do gravador deve estar sempre ativa se alguma instância do leitor estiver ativa. Quando o Aurora retoma essa instância do gravador, isso também faz com que todas as instâncias do leitor nos níveis zero e um de promoção de failover sejam retomadas.
- Se você enviar uma solicitação ao seu cluster pela da API de dados do RDS, o Aurora retomará a instância do gravador se ela estiver pausada. Em seguida, o Aurora processa a solicitação da API de dados.
- Ao alterar o valor de um parâmetro de configuração em um grupo de parâmetros de cluster de banco de dados, o Aurora retoma automaticamente todas as instâncias do Aurora Serverless v2 pausadas em todos os clusters que usam esse grupo de parâmetros de cluster. De forma semelhante, ao alterar um valor de parâmetro em um grupo de parâmetros de banco de dados, o Aurora retoma automaticamente quaisquer instâncias do Aurora Serverless v2 pausadas que usam esse grupo de parâmetros de banco de dados. O mesmo comportamento de retomada automática se aplica ao modificar um cluster para atribuir um grupo de parâmetros de cluster diferente ou ao modificar uma instância para atribuir um grupo de parâmetros de banco de dados diferente.
- Executar uma solicitação de retrocesso automaticamente retoma a instância do gravador do Aurora Serverless v2 se ela estiver pausada. O Aurora processa a solicitação de retrocesso após a retomada da instância do gravador. Você pode retroceder para um momento em que uma instância do Aurora Serverless v2 estava pausada.
- Tirar um snapshot do cluster ou excluir um snapshot não retoma nenhuma instância do Aurora Serverless v2.
- Criar um clone do Aurora faz com que o Aurora retome a instância do gravador do cluster que está sendo clonado.

- Se uma instância pausada receber muitas solicitações de conexão antes de terminar de ser retomada, algumas sessões talvez não consigam se conectar. Recomendamos implementar a lógica de repetição para conexões com clusters do Aurora que têm algumas instâncias do Aurora Serverless v2 com pausa automática habilitada. Por exemplo, você pode tentar novamente qualquer conexão com falha três vezes.
- O Aurora pode realizar alguns tipos de pequenas manutenções internas sem ativar uma instância. No entanto, alguns tipos de manutenção que ocorrem durante a janela de manutenção do cluster exigem que o Aurora retome a instância. Quando a manutenção é concluída, a instância é automaticamente pausada novamente se não houver mais atividade após o intervalo especificado.

  > **Note**
  >
  > O Aurora não retoma automaticamente uma instância pausada para trabalhos agendados específicos do mecanismo, como os realizados na extensão `pg_cron` do PostgreSQL ou no programador de eventos do MySQL. Para garantir que esses trabalhos sejam executados, inicie uma conexão com a instância manualmente antes do horário agendado. O Aurora não coloca na fila nenhum trabalho cujo horário agendado ocorra enquanto a instância de banco de dados está pausada. Esses trabalhos são ignorados quando a instância é retomada posteriormente.

- Se o cluster do Aurora passar por um failover enquanto uma instância do Aurora Serverless v2 estiver pausada automaticamente, o Aurora pode retomar uma instância e, em seguida, promover essa instância para ser o gravador. O mesmo pode acontecer se uma ou mais instâncias de banco de dados forem removidas do cluster enquanto uma instância estiver pausada. Nesse caso, a instância se torna o gravador imediatamente quando é retomada.
- As operações que alteram as propriedades do cluster também fazem com que todas as instâncias do Aurora Serverless v2 pausadas automaticamente sejam retomadas. Por exemplo, uma instância pausada automaticamente é retomada para operações como:
  - Alterar o intervalo de escalabilidade do cluster.
  - Atualizar a versão do mecanismo do cluster.
  - Descrever ou baixar arquivos de log de uma instância pausada. Você pode examinar dados históricos de log de instâncias pausadas habilitando uploads de log para o CloudWatch e analisando os logs pelo CloudWatch.

Desativar a pausa automática para instâncias do Aurora Serverless v2 em um cluster

Siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster. Para a capacidade mínima, escolha um valor de 0,5 ou maior. Ao desativar o recurso de pausa automática, o intervalo de ociosidade da instância é redefinido. Se você reativar a pausa automática, precisará especificar um novo intervalo de tempo limite.

O exemplo de CLI a seguir mostra como você pode desativar o recurso de pausa automática em um cluster existente do Aurora. A saída `describe-db-clusters` mostra que o atributo `SecondsUntilAutoPause` é removido quando a capacidade mínima é definida como um valor diferente de zero.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 2,
    "MaxCapacity": 80.0
}
```

Como funciona o recurso de pausa automática do Aurora Serverless v2

Você pode usar as informações a seguir para planejar o uso do recurso de pausa automática. Compreender as circunstâncias em que as instâncias são pausadas, retomadas ou permanecem ativas pode ajudar você a equilibrar as compensações entre disponibilidade, capacidade de resposta e economia de custos.

Tópicos

- O que acontece quando as instâncias do Aurora Serverless v2 são pausadas
- O que acontece quando as instâncias do Aurora Serverless v2 pausadas automaticamente são retomadas
- Como o faturamento de instâncias funciona para clusters do Aurora Serverless v2 pausados automaticamente
- Situações em que o Aurora Serverless v2 não faz pausas automáticas
- Como a pausa automática funciona com o recurso interromper/iniciar do cluster

- Como a manutenção e as atualizações funcionam para clusters do Aurora Serverless v2 pausados automaticamente
- Diferenças no comportamento de pausa automática entre o Aurora Serverless v2 e o Aurora Serverless v1

O que acontece quando as instâncias do Aurora Serverless v2 são pausadas

Quando uma instância de banco de dados do Aurora Serverless v2 faz uma pausa após um período sem conexões:

- O Aurora começa a pausar a instância após o término do intervalo especificado sem nenhuma conexão com a instância, independentemente de quantas ACUs a instância tenha no momento.
- O mecanismo de pausa não é instantâneo. Uma instância do Aurora Serverless v2 que está prestes a ser pausada automaticamente pode esperar um pouco para acompanhar todas as alterações no armazenamento do Aurora.
- As cobranças para essa instância são suspensas. A métrica `ServerlessV2Usage` tem um valor de 0 enquanto a instância está pausada.
- O valor do status para a instância não é alterado. O status ainda é exibido como "disponível".
- A instância para de gravar nos arquivos de log do banco de dados. Ela para de enviar métricas para o CloudWatch, além de registrar 0% para `CPUUtilization` e `ACUUtilization` e 0 para `ServerlessDatabaseCapacity`.
- O Aurora emite eventos quando uma instância de banco de dados do Aurora Serverless v2 começa e termina a pausa e se o mecanismo de pausa é interrompido ou não é bem-sucedido. Consulte Eventos de instância de banco de dados para obter detalhes sobre esses eventos.

O que acontece quando as instâncias do Aurora Serverless v2 pausadas automaticamente são retomadas

Quando uma instância de banco de dados do Aurora Serverless v2 é retomada após ser pausada automaticamente, as seguintes condições se aplicam:

- Quaisquer alterações de parâmetros que estão em `pending-reboot` são aplicadas quando a instância é retomada.
- O Aurora emite eventos no nível da instância quando cada instância de banco de dados do Aurora Serverless v2 começa a ser retomada, termina de ser retomada e se a instância não puder ser

retomada por algum motivo. Consulte Eventos de instância de banco de dados para obter detalhes sobre esses eventos.

- Todas as conexões solicitadas são estabelecidas após a conclusão da retomada da instância de banco de dados. Como o tempo típico de retomada pode ser de aproximadamente 15 segundos, recomendamos que você ajuste as configurações de tempo limite do cliente para mais de 15 segundos. Por exemplo, nas configurações do driver JDBC, você pode ajustar os valores das configurações `connectTimeout` e `sslResponseTimeout` para que sejam acima de 15 segundos.

> **Note**
>
> Se uma instância do Aurora Serverless v2 permanecer pausada por mais de 24 horas, o Aurora pode colocar a instância em uma suspensão mais profunda que demora mais para ser retomada. Nesse caso, o tempo de retomada pode ser de 30 segundos ou mais, aproximadamente equivalente a uma reinicialização da instância. Se o banco de dados tiver períodos de inatividade que durem mais de um dia, recomendamos definir os tempos limite de conexão para 30 segundos ou mais.

Como o faturamento de instâncias funciona para clusters do Aurora Serverless v2 pausados automaticamente

Enquanto uma instância do Aurora Serverless v2 está pausada automaticamente, sua cobrança de instância é zero. A métrica `ServerlessV2Usage` é zero durante esse período. A AWS ainda cobra pelo armazenamento do Aurora e por outros aspectos do cluster que não estão vinculados a essa instância específica de banco de dados.

Situações em que o Aurora Serverless v2 não faz pausas automáticas

- Se o valor mínimo da capacidade do cluster de banco de dados for maior que zero ACUs, as instâncias do Aurora Serverless v2 no cluster não serão pausadas automaticamente. Se houver clusters existentes com instâncias do Aurora Serverless v2 de antes da disponibilidade do recurso de pausa automática, a configuração de capacidade mínima mais baixa era de 0,5 ACUs. Para usar o recurso de pausa automática com esses clusters, modifique a configuração de capacidade mínima para zero ACUs.
- Se alguma conexão iniciada pelo usuário estiver aberta em uma instância do Aurora Serverless v2, a instância não será pausada. A instância também não será pausada enquanto atividades como

aplicação de patches e atualizações estiverem em andamento. As conexões administrativas que o Aurora usa para verificações de integridade não são contadas como atividade e não impedem a pausa da instância.

- Em um cluster do Aurora PostgreSQL com a replicação lógica habilitada, ou em um cluster do Aurora MySQL com replicação de binlog habilitada, a instância do gravador e quaisquer instâncias do leitor nos níveis zero e um de promoção de failover não pausam automaticamente. O Aurora executa uma quantidade mínima constante de atividade para verificar a integridade da conexão de replicação.

  Para clusters com replicação habilitada, é possível ter instâncias do leitor do Aurora no cluster de origem que pausam automaticamente. Para fazer isso, defina a prioridade de failover delas como um valor diferente de zero ou um. Dessa forma, elas podem ser pausadas independentemente da instância do gravador.
- Caso seu cluster Aurora tenha um proxy RDS associado, o proxy manterá uma conexão aberta com cada instância de banco de dados no cluster. Portanto, nenhuma instância do Aurora Serverless v2 desse cluster será pausada automaticamente.
- Se o cluster for o principal em um banco de dados global do Aurora, o Aurora não pausará automaticamente a instância do gravador do Aurora Serverless v2. Isso porque é necessário um nível constante de atividade na instância do gravador para gerenciar os outros clusters no banco de dados global. Como a instância do gravador permanece ativa, quaisquer instâncias do leitor do Aurora Serverless v2 com prioridade de failover zero ou um também não são pausadas automaticamente.
- As instâncias do Aurora Serverless v2 nos clusters secundários de um banco de dados global do Aurora não pausam automaticamente. Se um cluster de banco de dados for promovido a um cluster autônomo, o recurso de pausa automática se tornará efetivo se todas as outras condições forem atendidas.
- Em um cluster com uma integração ETL zero associada ao Redshift, a instância do gravador e quaisquer instâncias do leitor nos níveis zero e um de promoção de failover não são pausadas automaticamente.
- Além da atividade na porta principal do banco de dados da instância, se uma instância do Aurora PostgreSQL tiver o recurso Babelfish habilitado, quaisquer conexões e atividades na porta T-SQL evitarão que a instância seja pausada automaticamente.

Como a pausa automática funciona com o recurso interromper/iniciar do cluster

Você pode interromper e iniciar um cluster do Aurora quando o recurso de pausa automática estiver habilitado. Não importa se algumas instâncias estão pausadas. Ao iniciar o cluster novamente, todas as instâncias do Aurora Serverless v2 pausadas são retomadas automaticamente.

Enquanto um cluster do Aurora está interrompido, nenhuma instância do Aurora Serverless v2 pausada é retomada automaticamente com base nas tentativas de conexão. Assim que o cluster é iniciado novamente, os mecanismos normais de pausa e retomada das instâncias do Aurora Serverless v2 se aplicam.

Como a manutenção e as atualizações funcionam para clusters do Aurora Serverless v2 pausados automaticamente

- Enquanto uma instância do Aurora Serverless v2 está pausada automaticamente, se você tentar atualizar o cluster do Aurora, o Aurora retoma a instância e a atualiza.
- O Aurora retoma periodicamente todas as instâncias do Aurora Serverless v2 pausadas automaticamente para realizar a manutenção, como pequenas atualizações de versões e alterações em propriedades, como grupos de parâmetros.
- Depois que uma instância do Aurora Serverless v2 é ativada para uma operação administrativa, como uma atualização ou manutenção, o Aurora espera pelo menos 20 minutos antes de pausar a instância novamente. Isso serve para permitir que quaisquer operações em segundo plano sejam concluídas. O período de 20 minutos também evita pausar e retomar a instância várias vezes se a instância passar por múltiplas operações administrativas em sucessão.

Diferenças no comportamento de pausa automática entre o Aurora Serverless v2 e o Aurora Serverless v1

- O tempo de retomada é melhorado no Aurora Serverless v2 em comparação com o Aurora Serverless v1. O tempo de retomada é normalmente de cerca de 15 segundos se a instância estiver pausada por menos de 24 horas. Se a instância estiver pausada por mais de 24 horas, o tempo de retomada poderá ser maior.
- A forma como o Aurora Serverless v2 se aplica aos clusters multi-AZ significa que algumas instâncias de banco de dados no cluster podem ser pausadas enquanto outras estão ativas. A instância do gravador é retomada sempre que algum leitor estiver em execução, porque o gravador é necessário para coordenar determinadas atividades no cluster. Como o Aurora Serverless v1 não usa instâncias do leitor, todo o cluster sempre estaria pausado ou ativo.

- Quando o endpoint do leitor escolhe aleatoriamente uma instância do leitor à qual se conectar, essa instância do leitor pode já estar ativa ou pausada automaticamente. Assim, o tempo para acessar a instância do leitor pode variar e ser mais difícil de prever. Portanto, os clusters multi-AZ que usam o Aurora Serverless v2 e pausam automaticamente podem se beneficiar da configuração de endpoints personalizados para casos de uso específicos de somente leitura, em vez de direcionar todas as sessões somente leitura para o endpoint do leitor.
- As instâncias do Aurora Serverless v2 passam por operações de manutenção com a mesma frequência que as instâncias provisionadas. Como o Aurora retoma automaticamente as instâncias quando essa manutenção é necessária, é possível notar que algumas instâncias do Aurora Serverless v2 são retomadas com mais frequências do que os clusters do Aurora Serverless v1 eram retomados.

Como a pausa automática do Aurora Serverless v2 funciona para diferentes tipos de clusters do Aurora

As considerações sobre o recurso de pausa automática dependem de quantas instâncias estão em seu cluster do Aurora, dos níveis de promoção de failover das instâncias do leitor e se todas as instâncias são do Aurora Serverless v2 ou uma combinação de instâncias do Aurora Serverless v2 e provisionadas.

Tópicos

- Layouts de cluster do Aurora recomendados ao usar a pausa automática
- Como a pausa automática do Aurora Serverless v2 funciona para a instância do gravador em um cluster de banco de dados
- Como a pausa automática do Aurora Serverless v2 funciona em clusters multi-AZ
- Como a pausa automática do Aurora Serverless v2 funciona em clusters com instâncias provisionadas

Layouts de cluster do Aurora recomendados ao usar a pausa automática

Quando o recurso de pausa automática está habilitado, você pode organizar seu cluster do Aurora para obter o equilíbrio ideal entre alta disponibilidade, resposta rápida e escalabilidade de acordo com seu caso de uso. Você faz isso escolhendo a combinação de instâncias do Aurora Serverless v2, instâncias provisionadas e níveis de promoção de failover para as instâncias de banco de dados em seu cluster.

Os seguintes tipos de configurações demonstram diferentes compensações entre alta disponibilidade e otimização de custos para seu cluster:

- Para um sistema de desenvolvimento e teste, você pode configurar um cluster de banco de dados de zona de disponibilidade única com uma instância de banco de dados do Aurora Serverless v2. A única instância atende a todas as solicitações de leitura e gravação. Quando o cluster não é usado por intervalos de tempo significativos, a instância de banco de dados é pausada. Nesse ponto, os custos de computação do banco de dados para o cluster também são pausados.
- Em um sistema que executa uma aplicação em que a alta disponibilidade é uma prioridade, mas o cluster ainda tem períodos em que está totalmente ocioso, é possível configurar um cluster multi-AZ onde as instâncias de banco de dados do gravador e do leitor são do Aurora Serverless v2. Defina a instância do leitor como prioridade de failover zero ou um, desta forma, o gravador e a instância do leitor pausam e retomam ao mesmo tempo. Agora você tem o benefício de um failover rápido enquanto o cluster está ativo. Quando o cluster permanece ocioso por mais tempo do que o limite da pausa automática, as cobranças da instância de banco de dados das duas instâncias são pausadas. Quando o cluster retoma o processamento, a primeira sessão do banco de dados demora um pouco para se conectar.
- Suponha que seu cluster esteja constantemente ativo com uma quantidade mínima de atividade e exija uma resposta rápida para qualquer conexão. Nesse caso, você pode criar um cluster com mais de uma instância do leitor do Aurora Serverless v2 e dissociar as capacidades de algumas instâncias de leitura do gravador. Especifique a prioridade de failover zero ou uma para a instância do gravador e uma instância do leitor. Especifique pelo menos duas prioridades para as outras instâncias do leitor. Dessa forma, as instâncias do leitor nos níveis de prioridade mais alta podem pausar automaticamente, mesmo enquanto o gravador e um dos leitores permanecem ativos.

  Nesse caso, você pode empregar algumas outras técnicas para garantir que o cluster permaneça continuamente disponível e, ao mesmo tempo, reduzir a escala verticalmente para uma baixa capacidade durante os períodos de inatividade:

  - Você pode usar instâncias provisionadas para o gravador e o leitor de prioridade zero ou um. Dessa forma, duas instâncias de banco de dados nunca pausam automaticamente e estão sempre disponíveis para atender ao tráfego do banco de dados e executar failovers.
  - Você pode configurar um endpoint personalizado que inclua as instâncias do Aurora Serverless v2 nos níveis de prioridade mais alta, mas não o gravador ou os leitores do nível de promoção zero ou um. Dessa forma, é possível direcionar sessões somente leitura que não são sensíveis à latência para os leitores que podem ser pausados automaticamente. Você pode evitar usar o endpoint do leitor para essas solicitações, porque o Aurora pode direcionar as conexões do

endpoint do leitor para a instância do leitor sempre ativa ou para uma das instâncias pausadas automaticamente. O uso do endpoint personalizado permite que você direcione conexões para diferentes grupos de instâncias com base em sua preferência por resposta rápida ou por capacidade de ajuste de escala extra.

Como a pausa automática do Aurora Serverless v2 funciona para a instância do gravador em um cluster de banco de dados

Quando um cluster de banco de dados do Aurora contém apenas uma única instância de banco de dados, o mecanismo de pausa e retomada automática da instância de banco de dados é simples. Ele depende apenas da atividade na instância do gravador. É possível definir essa configuração para clusters usados para desenvolvimento e testes ou para executar aplicações em que a alta disponibilidade não é crucial. Observe que, em um cluster de instância única, o Aurora direciona as conexões pelo endpoint do leitor para a instância de banco de dados do gravador. Assim, para um cluster de banco de dados de instância única, tentar se conectar ao endpoint do leitor faz com que a instância de banco de dados do gravador pausada automaticamente seja retomada.

Os seguintes fatores adicionais se aplicam aos clusters do Aurora com várias instâncias de banco de dados:

- Em um cluster de banco de dados do Aurora, a instância de banco de dados do gravador normalmente é acessada com frequência. Portanto, é possível notar que a instância de banco de dados do gravador permanece ativa, mesmo quando uma ou mais das instâncias de banco de dados do leitor pausam automaticamente.
- Determinadas atividades nas instâncias de banco de dados do leitor exigem que a instância de banco de dados do gravador esteja disponível. Portanto, as instâncias de banco de dados do gravador não podem ser pausadas até que todas as instâncias do leitor também sejam pausadas. A retomada de qualquer instância do leitor automaticamente retoma a instância do gravador, mesmo que a aplicação não esteja acessando a instância do gravador diretamente.
- As instâncias do leitor do Aurora Serverless v2 nos níveis zero e um de promoção de failover escalam para manter sua capacidade sincronizada com a instância do gravador. Assim, quando uma instância do gravador do Aurora Serverless v2 é retomada, o mesmo acontece com quaisquer instâncias do leitor do Aurora Serverless v2 que estejam nos níveis de promoção zero ou um.

Como a pausa automática do Aurora Serverless v2 funciona em clusters multi-AZ

Em um cluster de banco de dados do Aurora que contém uma instância do gravador e uma ou mais instâncias de banco de dados do leitor, algumas instâncias de banco de dados do Aurora Serverless v2 podem ser pausadas enquanto outras estão ativas. A instância do gravador e quaisquer instâncias do leitor com prioridade de failover 0 e 1 sempre pausam e são retomadas ao mesmo tempo. As instâncias do leitor com prioridade diferente de 0 ou 1 podem pausar e retomar independentemente das outras instâncias.

Ao usar esse recurso para clusters com várias instâncias do leitor, é possível gerenciar custos sem sacrificar a alta disponibilidade. A instância do gravador e outra uma ou duas instâncias do leitor podem permanecer ativas o tempo todo, enquanto instâncias adicionais do leitor podem pausar quando não são necessárias para lidar com tráfego de leitura de alto volume.

A pausa automática de uma instância do leitor depende se sua capacidade pode escalar de forma independente ou se a capacidade estiver vinculada à capacidade da instância de banco de dados do gravador. Essa propriedade de ajuste de escala depende da prioridade de failover da instância de banco de dados do leitor. Quando a prioridade do leitor é zero ou um, a capacidade do leitor acompanha a capacidade da instância de banco de dados do gravador. Portanto, para permitir que as instâncias de banco de dados do leitor sejam pausadas automaticamente na maior variedade de situações, defina a prioridade para um valor maior que um.

O tempo para retomar uma instância do leitor pode ser um pouco maior do que para retomar uma instância do gravador. Para obter uma resposta mais rápida caso as instâncias possam ser pausadas, conecte-se ao endpoint do cluster.

Como a pausa automática do Aurora Serverless v2 funciona em clusters com instâncias provisionadas

Quaisquer instâncias de banco de dados provisionadas no cluster de banco de dados do Aurora não são pausadas automaticamente. Somente instâncias de banco de dados do Aurora Serverless v2, com a classe de instância `db.serverless`, podem usar o recurso de pausa automática.

Quando o cluster do Aurora contém qualquer instância de banco de dados provisionada, nenhuma instância do gravador do Aurora Serverless v2 faz uma pausa automática. Isso se deve ao requisito de que a instância do gravador permaneça disponível enquanto quaisquer instâncias do leitor estiverem ativas. O fato de o gravador do Aurora Serverless v2 permanecer ativo também significa que quaisquer instâncias do leitor do Aurora Serverless v2 com prioridade de failover 0 e 1 não pausarão automaticamente em um cluster híbrido que contém instâncias provisionadas.

Monitorar clusters do Aurora que usam a pausa automática

Para monitorar o Aurora, é preciso ter familiaridade com os procedimentos de monitoramento em Monitorar métricas do Amazon Aurora com o Amazon CloudWatch e as métricas do CloudWatch listadas em Referência de métricas do Amazon Aurora. Lembre-se de que há algumas considerações especiais ao monitorar clusters do Aurora que usam o recurso de pausa automática:

- Pode haver períodos em que as instâncias do Aurora Serverless v2 não estão registrando os dados de log e a maioria das métricas porque as instâncias estão pausadas. As únicas métricas enviadas ao CloudWatch enquanto uma instância está pausada são 0% para `CPUUtilization` e `ACUUtilization`, e 0 para `ServerlessDatabaseCapacity`.
- É possível verificar se as instâncias do Aurora Serverless v2 estão pausando com mais ou menos frequência do que o esperado. Para fazer isso, verifique com que frequência a métrica `ServerlessDatabaseCapacity` muda de um valor diferente de zero para zero e por quanto tempo ela permanece zero. Se as instâncias não permanecerem pausadas pelo tempo esperado, não haverá uma grande economia de custos. Se as instâncias pausarem e forem retomadas com mais frequência do que o pretendido, o cluster pode ter uma latência desnecessária ao responder às solicitações de conexão. Para obter informações sobre os fatores que afetam se as instâncias do Aurora Serverless v2 podem pausar e com que frequência, consulte Pré-requisitos e limitações do recurso de pausa automática do Aurora Serverless v2, Situações em que o Aurora Serverless v2 não faz pausas automáticas e Solucionar problemas da pausa automática do Aurora Serverless v2.
- Você também pode examinar um arquivo de log que registra as operações automáticas de pausa e retomada de uma instância do Aurora Serverless v2. Se uma instância não pausou após o término do intervalo de tempo limite, esse arquivo de log também incluirá o motivo pelo qual a pausa automática não ocorreu. Para ter mais informações, consulte Monitorar a atividade de pausa e retomada do Aurora Serverless v2.

Tópicos

- Verificar se uma instância do Aurora Serverless v2 está pausada
- Eventos para operações de pausa automática e retomada automática
- Como a pausa automática funciona com o Insights de Performance e o Monitoramento aprimorado
- Como o recurso de pausa automática do Aurora Serverless v2 interage com as métricas do Aurora

Verificar se uma instância do Aurora Serverless v2 está pausada

Para determinar se uma instância do Aurora Serverless v2 está no estado de pausa, observe a métrica `ACUUtilization` da instância. Essa métrica tem um valor zero enquanto a instância está pausada.

Enquanto uma instância do Aurora Serverless v2 estiver pausada, seu valor de status ainda é listado como Disponível. O mesmo se aplica quando uma instância pausada do Aurora Serverless v2 está em processo de retomada. Isso ocorre porque você pode se conectar com sucesso a essa instância, mesmo que a conexão sofra um pequeno atraso.

Qualquer métrica relacionada à disponibilidade das instâncias do Aurora considera o período em que a instância está pausada como o tempo em que a instância estava disponível.

Eventos para operações de pausa automática e retomada automática

O Aurora emite eventos para instâncias do Aurora Serverless v2 quando as operações de pausa automática e retomada automática começam, terminam ou são canceladas. Os eventos relacionados ao recurso de pausa automática vão de `RDS-EVENT-0370` auto `RDS-EVENT-0374`. Consulte Eventos de instância de banco de dados para obter detalhes sobre esses eventos.

Como a pausa automática funciona com o Insights de Performance e o Monitoramento aprimorado

Enquanto uma instância do Aurora Serverless v2 está pausada, o Aurora não coleta informações de monitoramento dessa instância por meio do Insights de Performance ou do Monitoramento aprimorado. Quando a instância é retomada, pode haver um breve atraso até o Aurora retomar a coleta dessas informações de monitoramento.

Como o recurso de pausa automática do Aurora Serverless v2 interage com as métricas do Aurora

Enquanto uma instância do Aurora Serverless v2 está pausada, ela não emite a maioria das métricas do CloudWatch nem grava nenhuma informação nos logs do banco de dados. As únicas métricas enviadas ao CloudWatch enquanto uma instância está pausada são 0% para `CPUUtilization` e `ACUUtilization`, e 0 para `ServerlessDatabaseCapacity`.

Quando o CloudWatch está computando estatísticas relacionadas à disponibilidade e ao tempo de atividade da instância ou do cluster, as instâncias do Aurora Serverless v2 são consideradas disponíveis durante o tempo em que estão pausadas.

Ao iniciar uma ação da AWS CLI ou da API do RDS para descrever ou baixar os logs de uma instância pausada do Aurora Serverless v2, a instância é retomada automaticamente para disponibilizar as informações do log.

Exemplo de métricas do CloudWatch

Os exemplos da AWS CLI a seguir mostram como é possível observar como a capacidade de uma instância muda ao longo do tempo. Durante o período, a instância faz uma pausa automática e depois é retomada. Enquanto está pausada, a métrica `ServerlessDatabaseCapacity` relata um valor zero. Para determinar se a instância foi pausada em algum momento durante o período, verificamos se a capacidade mínima durante esse período foi zero.

O exemplo de Linux a seguir representa uma instância do Aurora Serverless v2 que foi pausada automaticamente por algum tempo. A métrica `ServerlessDatabaseCapacity` foi amostrada a cada minuto durante um período de três minutos. O valor mínimo de ACU de 0,0 confirma que a instância foi pausada em algum momento durante cada minuto.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '3 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:11:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:12:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

Em seguida, tentamos fazer uma conexão com a instância pausada do Aurora Serverless v2. Neste exemplo, usamos uma senha incorreta intencionalmente para que a tentativa de conexão não seja bem-sucedida. Apesar da falha, a tentativa de conexão faz com que o Aurora retome a instância pausada.

```
$ mysql -h my_cluster_endpoint.rds.amazonaws.com -u admin -pwrong-password

ERROR 1045 (28000): Access denied for user 'admin'@'ip_address' (using password: YES)
```

O exemplo de Linux a seguir demonstra que a instância pausada foi retomada, permaneceu inativa por aproximadamente cinco minutos, e depois voltou ao estado de pausa. A instância foi retomada com uma capacidade de 2,0 ACUs. Então ela aumentou a escala verticalmente levemente, por exemplo, para realizar alguma limpeza interna. Como ela não recebeu nenhuma tentativa de conexão do usuário dentro do período de tempo limite de cinco minutos, ela passou de 4,0 ACUs diretamente para o estado de pausa.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '8 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
DATAPOINTS 2.0 2024-08-02T22:14:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:15:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:16:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:17:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:18:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:19:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:20:00+00:00 None
```

Se o relatório tivesse a intenção de mostrar a rapidez com que a instância aumentou a escala verticalmente para lidar com a workload, poderíamos especificar `Maximum` para a estatística em vez de `Minimum`. Para planejamento de capacidade e estimativa de custos, podemos especificar um período mais longo e usar a estatística `Average`. Dessa forma, poderíamos determinar os valores de capacidade típicos durante períodos de alta atividade, baixa atividade e estado de pausa. Para examinar o comportamento durante os momentos precisos de pausa e retomada, podemos especificar um período de um segundo e examinar um intervalo de tempo menor. Os valores do carimbo de data e hora na saída, como `2024-08-02T22:13:00+00:00`, demonstram o formato para especificar parâmetros precisos para as opções `--start-time` e `--end-time`.

Solucionar problemas da pausa automática do Aurora Serverless v2

Se você notar que as instâncias do Aurora Serverless v2 não estão pausando com a frequência esperada, verifique as seguintes possíveis causas:

- Confirme se a versão do Aurora que você está usando é compatível com uma capacidade mínima de zero ACUs. Consulte Aurora Serverless v2Capacidade do para obter os intervalos de capacidade das diferentes versões do Aurora.
- Confirme se o valor mínimo da capacidade do cluster está definido como zero ACUs.
- Confirme se a instância em questão está realmente usando a instância do Aurora Serverless v2 de classe `db.serverless`, e não uma das classes de instância provisionadas.
- Confirme se o cluster não está usando nenhum dos recursos ou configurações incompatíveis de Pré-requisitos e limitações do recurso de pausa automática do Aurora Serverless v2.
- Examine o arquivo de log que mostra quando as instâncias do Aurora Serverless v2 foram pausadas, retomadas ou quando o Aurora não conseguiu pausar ou retomar uma instância por algum motivo. Para ter mais informações, consulte Monitorar a atividade de pausa e retomada do Aurora Serverless v2.
- Verifique se algum cliente ou aplicação está mantendo as conexões abertas por longos períodos de tempo. Por outro lado, verifique se alguma aplicação que usa a API de dados do RDS ou as funções do Lambda está enviando solicitações frequentes para que a instância nunca fique ociosa por tempo suficiente para pausar. Você pode examinar as métricas do CloudWatch, como `ConnectionAttempts` e `DatabaseConnections`. Para ter mais informações, consulte Métricas no nível da instância do Amazon Aurora.
- Se uma instância do leitor raramente pausar, verifique sua prioridade de failover. Se o leitor for usado para ajuste de escala de leitura e não como em espera em caso de failover, defina-o com uma prioridade na faixa de 2 a 15.
- Se a instância do gravador raramente pausar, verifique o uso das instâncias do leitor. O gravador só pode pausar se todo o cluster estiver ocioso. Isso ocorre porque uma conexão com qualquer instância do leitor faz com que o gravador retome.
- Se sua aplicação atingir os tempos limite durante as solicitações de conexão enquanto as instâncias do Aurora Serverless v2estão retomando, considere aumentar o intervalo de tempo limite usado pelo código da aplicação ou pela estrutura de banco de dados subjacente. Tempos limite de conexão mais longos reduzem a possibilidade de falhas nas conexões enquanto as instâncias do Aurora Serverless v2 estão sendo retomadas. No entanto, os tempos limite mais longos também podem tornar a aplicação mais lenta para detectar problemas de disponibilidade no cluster.

  Por outro lado, considere aumentar o intervalo de pausa automática para que as aplicações não encontrem instâncias pausadas com tanta frequência.

Se não houver um equilíbrio lógico entre o comportamento de pausa automática e a responsividade do cluster para a aplicação, esse cluster pode não ser um bom candidato para usar o recurso de pausa automática.

- Se você estimar por quanto tempo as instâncias do Aurora Serverless v2 ficarão pausadas, é preciso ter em mente que há fatores que tornam impraticável fazer previsões precisas.
  - As instâncias podem retomar periodicamente para realizar manutenção, atualizações de versões menores ou aplicar alterações em grupos de parâmetros.
  - Para clusters multi-AZ, há situações em que a retomada de uma instância faz com que outras instâncias também retomem. Retomar qualquer leitor sempre faz com que o escritor retome. Retomar o escritor sempre faz com que quaisquer leitores com prioridade de failover 0 e 1 retomem.

  Recomendamos medir o tempo real de pausa em um período de vários dias, com uma workload realista. Em seguida, use essas medições para definir uma linha de base para a proporção de tempo em que você pode esperar que uma instância esteja pausada.
- Operações internas, como a limpeza do MySQL, o agendador de eventos do MySQL, o autovacuum do PostgreSQL ou trabalhos do PostgreSQL agendados por meio da extensão `pg_cron`, não estão em execução ou não estão sendo concluídos. A instância não é retomada automaticamente para realizar essas operações que não envolvem uma conexão do usuário com o banco de dados. Se essas operações internas estiverem em andamento quando o tempo limite da pausa automática expirar, tarefas de manutenção, como a limpeza do MySQL e o autovacuum do PostgreSQL, serão canceladas. Os trabalhos agendados pelo agendador de eventos do MySQL ou pela extensão `pg_cron` do PostgreSQL também são cancelados se estiverem em andamento quando o Aurora iniciar a operação de pausa.

  Se for necessário garantir que a instância seja ativada periodicamente para realizar operações agendadas, é possível iniciar uma conexão para retomar a instância antes do horário de início do trabalho. Também é possível aumentar o intervalo de tempo limite da pausa automática para que operações como o autovacuum possam ser executadas por mais tempo após o término da atividade do usuário. Você também pode usar mecanismos como as funções do Lambda para realizar operações de banco de dados em um cronograma, de forma que a instância seja retomada automaticamente, se necessário.

Considerações sobre o design da aplicação para o recurso de pausa automática do Aurora Serverless v2

Quando uma instância de banco de dados do Aurora Serverless v2 é retomada após ser pausada automaticamente, ela começa com uma capacidade relativamente pequena e aumenta a escala verticalmente. Essa capacidade inicial se aplica mesmo que a instância de banco de dados tenha tido uma capacidade maior imediatamente antes de ser pausada automaticamente.

Use esse recurso com aplicações que podem tolerar um intervalo de aproximadamente 15 segundos ao estabelecer uma conexão. Isso leva em consideração o caso típico em que uma instância do Aurora Serverless v2 é retomada devido a uma ou a um pequeno número de conexões de entrada. Se a instância estiver pausada por mais de 24 horas, o tempo para retomada poderá ser maior.

Se a aplicação já estava usando o Aurora Serverless v1 e o recurso de pausa automática, talvez já existam intervalos de tempo limite habilitados para tentativas de conexão. Se você já estiver usando o Aurora Serverless v2 em combinação com o recurso interromper/iniciar de cluster, o tempo de retomada para instâncias do Aurora Serverless v2 pausadas automaticamente é geralmente muito menor do que o tempo para iniciar um cluster que está parado.

Ao codificar a lógica de conexão na aplicação, tente a conexão novamente se a primeira tentativa retornar um erro que tenha uma causa transitória. (Se o erro for devido a uma falha de autenticação, corrija as credenciais antes de tentar novamente.) Um erro que ocorre imediatamente após a retomada pode ser um erro de tempo limite ou algum erro relacionado aos limites do banco de dados. Tentar novamente pode lidar com problemas nos casos mais raros em que uma instância do Aurora Serverless v2 é retomada devido a muitas solicitações de conexão simultâneas. Nesse caso, algumas conexões podem levar mais tempo do que o normal para serem processadas ou podem exceder o limite de conexões simultâneas na primeira tentativa.

Durante o desenvolvimento e a depuração da aplicação, não deixe sessões de cliente ou ferramentas de programação com conexões abertas ao banco de dados. O Aurora não pausará uma instância se houver quaisquer conexões abertas iniciadas pelo usuário, independentemente de as conexões não estarem executando nenhuma instrução ou transação SQL. Quando uma instância do Aurora Serverless v2 em um cluster do Aurora não pode pausar, outras instâncias no cluster também podem ser impedidas de pausar. Para ter mais informações, consulte Situações em que o Aurora Serverless v2 não faz pausas automáticas.

O Aurora emite eventos quando uma instância de banco de dados do Aurora Serverless v2 começa a retomar, termina de retomar e se a instância não puder retomar por algum motivo. Consulte Eventos de instância de banco de dados para obter detalhes sobre esses eventos.

Migrar para o Aurora Serverless v2

Para converter um cluster de banco de dados existente para usar o Aurora Serverless v2, é possível fazer o seguinte:

- Realizar upgrade usando um cluster de banco de dados provisionado do Aurora.
- Realizar a atualização a partir de um cluster do Aurora Serverless v1.
- Migrar de um banco de dados on-premises para um cluster do Aurora Serverless v2.

Quando o cluster atualizado estiver executando a versão do mecanismo apropriada, conforme listado em Requisitos e limitações do Aurora Serverless v2, você poderá começar a adicionar instâncias de banco de dados do Aurora Serverless v2 a ele. A primeira instância de banco de dados adicionada ao cluster atualizado deve ser uma instância de banco de dados provisionada. Depois, você pode alternar o processamento para a workload de gravação, a workload de leitura ou ambas para as instâncias de banco de dados do Aurora Serverless v2.

Sumário

- Atualizar ou alternar clusters existentes para usar o Aurora Serverless v2
  - Caminhos de atualização para que clusters compatíveis com MySQL usem o Aurora Serverless v2
  - Caminhos de atualização para que clusters compatíveis com PostgreSQL usem o Aurora Serverless v2
- Alternar de um cluster provisionado para o Aurora Serverless v2
- Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1
  - Comparação entre os requisitos do Aurora Serverless v2 e do Aurora Serverless v1
  - Comparação de escalabilidade e disponibilidade entre o Aurora Serverless v2 e o Aurora Serverless v1
  - Comparação de compatibilidade de recursos entre o Aurora Serverless v2 e o Aurora Serverless v1
  - Adaptar casos de uso do Aurora Serverless v1 para o Aurora Serverless v2
- Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2
  - Clusters de banco de dados compatíveis com o Aurora MySQL
  - Clusters de banco de dados compatíveis com o Aurora PostgreSQL
- Migrar de um banco de dados on-premises para o Aurora Serverless v2

> **Note**
>
> Esses tópicos descrevem como converter um cluster de banco de dados existente. Para obter informações sobre como criar um cluster de banco de dados do Aurora Serverless v2, consulte Criar um cluster de banco de dados que usa o Aurora Serverless v2.

Atualizar ou alternar clusters existentes para usar o Aurora Serverless v2

Se o cluster provisionado tiver uma versão do mecanismo que seja compatível com o Aurora Serverless v2, alternar para o Aurora Serverless v2 não exigirá atualização. Nesse caso, você pode adicionar instâncias de banco de dados do Aurora Serverless v2 ao cluster original. Você pode alternar o cluster para usar todas as instâncias de banco de dados do Aurora Serverless v2. Também é possível usar uma combinação de Aurora Serverless v2 e instâncias de banco de dados provisionadas no mesmo cluster de banco de dados. Sobre as versões do mecanismo do Aurora que são compatíveis com o Aurora Serverless v2, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com o Aurora Serverless v2.

Se você estiver executando uma versão inferior do mecanismo que não seja compatível com o Aurora Serverless v2, realize estas etapas gerais:

1. Atualize o cluster.
2. Crie uma instância de banco de dados do gravador provisionada para o cluster atualizado.
3. Modifique o cluster para usar instâncias de banco de dados do Aurora Serverless v2.

> **Important**
>
> Ao realizar uma atualização de versão principal para uma versão compatível com o Aurora Serverless v2 usando restauração ou clonagem de snapshot, a primeira instância de banco de dados que você adicionar ao novo cluster deverá ser uma instância de banco de dados provisionada. Essa adição inicia a fase final do processo de atualização.
> Até que esse estágio final aconteça, o cluster não terá a infraestrutura necessária para compatibilidade com o Aurora Serverless v2. Dessa forma, esses clusters atualizados sempre começam com uma instância de banco de dados do gravador provisionada. Depois, você pode converter ou fazer failover da instância de banco de dados provisionada para um Aurora Serverless v2.

A atualização do Aurora Serverless v1 para o Aurora Serverless v2 envolve a criação de um cluster provisionado como etapa intermediária. Depois, realize as mesmas etapas de atualização que são realizadas quando você começa com um cluster provisionado.

Caminhos de atualização para que clusters compatíveis com MySQL usem o Aurora Serverless v2

Se o cluster original estiver executando o Aurora MySQL, escolha o procedimento apropriado dependendo da versão do mecanismo e do modo de mecanismo do cluster.

Se o cluster original do Aurora MySQL for este	Faça isso para alternar para o Aurora Serverless v2
Cluster provisionado executando o Aurora MySQL versão 3, compatível com o MySQL 8.0	Esta é a fase final de todas as conversões de clusters existentes do Aurora MySQL. Se necessário, realize uma atualização de versão secundária para a versão 3.02.0 ou posterior. Use uma instância de banco de dados provisionada para a instância de banco de dados do gravador. Adicione uma instância de banco de dados do leitor do Aurora Serverless v2. Depois, realize um failover para transformá-la na instância de banco de dados do gravador. (Opcional) Converta outras instâncias de banco de dados provisionadas no cluster em Aurora Serverless v2. Ou adicione novas instâncias de banco de dados do Aurora Serverless v2 e remova as instâncias de banco de dados provisionadas. Para saber o procedimento completo e ver os exemplos, consulte Alternar de um cluster provisionado para o Aurora Serverless v2.
Cluster provisionado executando o Aurora MySQL versão 2, compatível com o MySQL 5.7	Realize uma atualização de versão principal para o Aurora MySQL versão 3.02.0 ou

Se o cluster original do Aurora MySQL for este	Faça isso para alternar para o Aurora Serverless v2
	posterior. Depois, siga o procedimento relativo ao Aurora MySQL versão 3 para alternar o cluster para usar instâncias de banco de dados do Aurora Serverless v2.
Aurora Serverless v1Cluster do executando o Aurora MySQL versão 2, compatível com o MySQL 5.7	Para ajudar a planejar sua conversão do Aurora Serverless v1, primeiro consulte Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1. Depois, siga o procedimento em Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.

Caminhos de atualização para que clusters compatíveis com PostgreSQL usem o Aurora Serverless v2

Se o cluster original estiver executando o Aurora PostgreSQL, escolha o procedimento apropriado dependendo da versão do mecanismo e do modo de mecanismo do cluster.

Se o cluster original do Aurora PostgreSQL for este	Faça isso para alternar para o Aurora Serverless v2
Cluster provisionado executando o Aurora PostgreSQL versão 13	Esta é a fase final de todas as conversões de clusters existentes do Aurora PostgreSQL. Se necessário, realize uma atualização de versão secundária para a versão 13.6 ou superior. Adicione uma instância de banco de dados provisionada para a instância de banco de dados do gravador. Adicione uma instância de banco de dados do leitor do Aurora Serverless v2. Realize um failover para transformar essa instância do Aurora

Se o cluster original do Aurora PostgreSQL for este	Faça isso para alternar para o Aurora Serverless v2
	Serverless v2 na instância de banco de dados do gravador. (Opcional) Converta outras instâncias de banco de dados provisionadas no cluster em Aurora Serverless v2. Ou adicione novas instâncias de banco de dados do Aurora Serverless v2 e remova as instâncias de banco de dados provisionadas. Para saber o procedimento completo e ver os exemplos, consulte Alternar de um cluster provisionado para o Aurora Serverless v2.
Cluster provisionado executando o Aurora PostgreSQL versão 11 ou 12	Realize uma atualização de versão principal para o Aurora PostgreSQL versão 13.6 ou superior. Depois, siga o procedimento relativo ao Aurora PostgreSQL versão 13 para alternar o cluster para usar instâncias de banco de dados do Aurora Serverless v2.
Cluster do Aurora Serverless v1 executando o Aurora PostgreSQL versão 11 ou 13	Para ajudar a planejar sua conversão do Aurora Serverless v1, primeiro consulte Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1. Depois, siga o procedimento em Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.

Alternar de um cluster provisionado para o Aurora Serverless v2

Para alternar um cluster provisionado para usar o Aurora Serverless v2, siga estas etapas:

1. Confira se o cluster provisionado precisa ser atualizado para ser usado com instâncias de banco de dados do Aurora Serverless v2. Sobre as versões do Aurora compatíveis com o Aurora Serverless v2, consulte Requisitos e limitações do Aurora Serverless v2.

   Se o cluster provisionado estiver executando uma versão do mecanismo que não esteja disponível para o Aurora Serverless v2, atualize a versão do mecanismo do cluster:

   - Se você tiver um cluster provisionado compatível com o MySQL 5.7, siga as instruções de atualização do Aurora MySQL versão 3. Use os procedimentos em Como realizar uma atualização local.
   - Se você tiver um cluster provisionado compatível com PostgreSQL que esteja executando o PostgreSQL versão 11 ou 12, siga as instruções de atualização do Aurora PostgreSQL versão 13. Use os procedimentos em Realizar uma atualização da versão principal.
2. Configure todas as outras propriedades do cluster para corresponder aos requisitos do Aurora Serverless v2 usando o Requisitos e limitações do Aurora Serverless v2.
3. Defina a configuração de escalabilidade do cluster. Siga o procedimento em Configurar o intervalo de capacidade de Aurora Serverless v2 para um cluster.
4. Adicione uma ou mais instâncias de banco de dados do Aurora Serverless v2 ao cluster. Siga o procedimento geral em Adicionar réplicas do Aurora a um cluster de banco de dados. Sobre cada nova instância de banco de dados, especifique o nome da classe de instância de banco de dados sem servidor no Console de gerenciamento da AWS ou `db.serverless` na AWS CLI ou na API do Amazon RDS.

   Em alguns casos, talvez você já tenha uma ou mais instâncias de banco de dados do leitor provisionadas no cluster. Se esse for o caso, será possível converter um dos leitores em uma instância de banco de dados do Aurora Serverless v2 em vez de criar outra instância. Para fazer isso, siga o procedimento em Converter um gravador ou leitor provisionado em Aurora Serverless v2.
5. Realize uma operação de failover para tornar uma das instâncias de banco de dados do Aurora Serverless v2 a instância de banco de dados do gravador do cluster.
6. (Opcional) Converta todas as instâncias de banco de dados provisionadas no Aurora Serverless v2 ou os remova do cluster. Siga o procedimento geral em Converter um gravador ou leitor provisionado em Aurora Serverless v2 ou em Excluir uma instância de banco de dados de um cluster de banco de dados do Aurora.

> **Tip**
>
> A remoção das instâncias de banco de dados provisionadas não é obrigatória. É possível configurar um cluster que contenha instâncias de banco de dados provisionadas e do Aurora Serverless v2. No entanto, até você se familiarizar com as características de performance e escalabilidade das instâncias de banco de dados do Aurora Serverless v2, recomendamos que você configure seus clusters com instâncias de banco de dados do mesmo tipo.

O exemplo da AWS CLI a seguir mostra o processo de alternância usando um cluster provisionado que está executando o Aurora MySQL versão 3.02.0. O cluster é chamado `mysql-80`. O cluster começa com duas instâncias de banco de dados provisionadas chamadas `provisioned-instance-1` e `provisioned-instance-2`, um gravador e um leitor. As duas usam a classe de instância de banco de dados `db.r6g.large`.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' --output text
mysql-80
provisioned-instance-2     False
provisioned-instance-1     True

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-1 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-1     db.r6g.large

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-2 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-2     db.r6g.large
```

Criamos uma tabela com alguns dados. Dessa forma, podemos confirmar que os dados e a operação do cluster são os mesmos antes e depois da alternância.

```
mysql> create database serverless_v2_demo;
mysql> create table serverless_v2_demo.demo (s varchar(128));
mysql> insert into serverless_v2_demo.demo values ('This cluster started with a
 provisioned writer.');
Query OK, 1 row affected (0.02 sec)
```

Primeiro, adicionamos um intervalo de capacidade ao cluster. Caso contrário, obteremos um erro ao adicionar instâncias de banco de dados do Aurora Serverless v2 ao cluster. Se usarmos o Console de gerenciamento da AWS para este procedimento, esta etapa será automática quando adicionarmos a primeira instância de banco de dados do Aurora Serverless v2.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql

An error occurred (InvalidDBClusterStateFault) when calling the CreateDBInstance
 operation:
Set the Serverless v2 scaling configuration on the parent DB cluster before creating a
 Serverless v2 DB instance.

$ # The blank ServerlessV2ScalingConfiguration attribute confirms that the cluster
 doesn't have a capacity range set yet.
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 --query
 'DBClusters[*].ServerlessV2ScalingConfiguration'
[]

$ aws rds modify-db-cluster --db-cluster-identifier mysql-80 \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
{
  "DBClusterIdentifier": "mysql-80",
  "ServerlessV2ScalingConfiguration": {
    "MinCapacity": 0.5,
    "MaxCapacity": 16
  }
}
```

Criamos dois leitores do Aurora Serverless v2 para substituir as instâncias de banco de dados originais. Fazemos isso especificando a classe de instância de banco de dados do `db.serverless` para as novas instâncias de banco de dados.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 --db-
cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}
```

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-2 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-2",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ # Wait for both DB instances to finish being created before proceeding.
$ aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
 && \
  aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-2
```

Realize um failover para tornar uma das instâncias de banco de dados do Aurora Serverless v2 o novo gravador do cluster.

```
$ aws rds failover-db-cluster --db-cluster-identifier mysql-80 \
  --target-db-instance-identifier serverless-v2-instance-1
{
  "DBClusterIdentifier": "mysql-80",
  "DBClusterMembers": [
    {
      "DBInstanceIdentifier": "serverless-v2-instance-1",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "serverless-v2-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
```

```
      "DBInstanceIdentifier": "provisioned-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "Status": "available"
}
```

São necessários alguns segundos para que essa alteração entre em vigor. Nesse momento, temos um gravador do Aurora Serverless v2 e um leitor do Aurora Serverless v2. Dessa forma, não precisamos de nenhuma das instâncias de banco de dados provisionadas originais.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
mysql-80
serverless-v2-instance-1        True
serverless-v2-instance-2        False
provisioned-instance-2          False
provisioned-instance-1          False
```

A última etapa no procedimento de alternância é excluir ambas as instâncias de banco de dados provisionadas.

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-2 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-2",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}

$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-1 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-1",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
```

```
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

Como verificação final, confirmamos que a tabela original é acessível e gravável a partir da instância de banco de dados do Aurora Serverless v2.

```
mysql> select * from serverless_v2_demo.demo;
+---------------------------------------------------+
| s                                                 |
+---------------------------------------------------+
| This cluster started with a provisioned writer.   |
+---------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into serverless_v2_demo.demo values ('And it finished with a Serverless
 v2 writer.');
Query OK, 1 row affected (0.01 sec)

mysql> select * from serverless_v2_demo.demo;
+---------------------------------------------------+
| s                                                 |
+---------------------------------------------------+
| This cluster started with a provisioned writer.   |
| And it finished with a Serverless v2 writer.      |
+---------------------------------------------------+
2 rows in set (0.01 sec)
```

Também nos conectamos à instância de banco de dados do leitor do Aurora Serverless v2 e confirmamos se os dados recém-gravados estão disponíveis também.

```
mysql> select * from serverless_v2_demo.demo;
+---------------------------------------------------+
| s                                                 |
+---------------------------------------------------+
| This cluster started with a provisioned writer.   |
| And it finished with a Serverless v2 writer.      |
+---------------------------------------------------+
2 rows in set (0.01 sec)
```

Comparação entre o Aurora Serverless v2 e o Aurora Serverless v1

Se você já estiver usando o Aurora Serverless v1, saiba as principais diferenças entre o Aurora Serverless v1 e o Aurora Serverless v2. As diferenças de arquitetura, como compatibilidade com instâncias de banco de dados de leitor, abrem novos tipos de casos de uso.

Você pode usar as tabelas a seguir para entender as diferenças mais importantes entre o Aurora Serverless v2 e o Aurora Serverless v1.

Tópicos

- Comparação entre os requisitos do Aurora Serverless v2 e do Aurora Serverless v1
- Comparação de escalabilidade e disponibilidade entre o Aurora Serverless v2 e o Aurora Serverless v1
- Comparação de compatibilidade de recursos entre o Aurora Serverless v2 e o Aurora Serverless v1
- Adaptar casos de uso do Aurora Serverless v1 para o Aurora Serverless v2

Comparação entre os requisitos do Aurora Serverless v2 e do Aurora Serverless v1

A tabela a seguir resume os diferentes requisitos para executar o banco de dados usando o Aurora Serverless v2 ou o Aurora Serverless v1. O Aurora Serverless v2 oferece versões posteriores dos mecanismos de banco de dados do Aurora MySQL e do Aurora PostgreSQL em comparação com o Aurora Serverless v1.

Recurso	Aurora Serverless v2Requisito do	Aurora Serverless v1Requisito do
Mecanismos de banco de dados	Aurora MySQL, Aurora PostgreSQL	Aurora MySQL, Aurora PostgreSQL
Versões compatíveis do Aurora MySQL	Consulte Aurora Serverless v2 com o Aurora MySQL.	Consulte Aurora Serverless v1 com o Aurora MySQL.
Versões compatíveis do Aurora PostgreSQL	Consulte Aurora Serverless v2 com o Aurora PostgreSQL.	Consulte Aurora Serverless v1 com o Aurora PostgreSQL.
Atualizar um cluster de banco de dados	Da mesma forma que os clusters de banco de	As atualizações de versões secundárias são aplicadas

Recurso	Aurora Serverless v2Requisito do	Aurora Serverless v1Requisito do
	dados provisionados, você pode realizar atualizações manualmente sem esperar que o Aurora atualize o cluster de banco de dados para você. Para obter mais informações, consulte Modificar um cluster de bancos de dados Amazon Aurora. **Note** Para realizar um upgrade de versão principal de 13.x para 14.x ou 15.x para um cluster de banco de dados compatível com o Aurora PostgreSQL, a capacidade máxima de seu cluster deve ser de pelo menos 2 ACUs.	automaticamente à medida que se tornam disponíveis. Para ter mais informações, consulte Aurora Serverless v1Versões de mecanismos de banco de dados do e do Aurora. Você pode realizar atualizaç ões de versões principai s manualmente. Para ter mais informações, consulte Modificar um cluster de banco de dados do Aurora Serverles s v1.

Recurso	Aurora Serverless v2Requisito do	Aurora Serverless v1Requisito do
Converter do cluster de banco de dados provisionado	É possível usar os seguintes métodos: • Adicione uma ou mais instâncias de banco de dados de leitor do Aurora Serverless v2 para um cluster provisionado existente. Para usar o Aurora Serverless v2 para o gravador, realize um failover para uma das instância s de banco de dados do Aurora Serverless v2. Para o cluster inteiro usar instâncias de banco de dados do Aurora Serverles s v2, remova todas as instâncias de banco de dados do gravador provision adas depois de promover a instância de banco de dados do Aurora Serverless v2 para o gravador. • Crie um cluster com o mecanismo de banco de dados e versão do mecanismo de banco de dados apropriados. Use qualquer um dos métodos padrão. Por exemplo, restaure um snapshot de cluster ou crie um clone de um cluster existente.	Restaurar o snapshot do cluster provisionado para criar um cluster do Aurora Serverless v1

Recurso	Aurora Serverless v2Requisito do	Aurora Serverless v1Requisito do
	Escolha Aurora Serverless v2 para algumas ou todas as instâncias de banco de dados do novo cluster. Se você criar o cluster por meio da clonagem, não poderá atualizar a versão do mecanismo simultane amente. Verifique se o cluster original já está executando uma versão do mecanismo compatível com o Aurora Serverless v2.	
Converter do cluster do Aurora Serverless v1	Siga o procedimento em Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.	Não aplicável
Classes disponíveis da instância de banco de dados	A classe da instância de banco de dados especial `db.serverless` . No Console de gerenciamento da AWS, é rotulado como Serverless (Sem servidor).	Não aplicável. O Aurora Serverless v1 usa o modo de mecanismo `serverless` .
Porta	Qualquer porta compatíve l com o MySQL ou o PostgreSQL	Somente porta do MySQL ou do PostgreSQL padrão
Endereço IP público permitido ?	Sim	Não

Recurso	Aurora Serverless v2Requisito do	Aurora Serverless v1Requisito do
Uma nuvem privada virtual (VPC) é obrigatória?	Sim	Sim. Cada cluster do Aurora Serverless v1 consome dois endpoints do Gateway Load Balancer alocados para sua VPC.

Comparação de escalabilidade e disponibilidade entre o Aurora Serverless v2 e o Aurora Serverless v1

A tabela a seguir resume as diferenças de escalabilidade e disponibilidade entre o Aurora Serverless v2 e o Aurora Serverless v1.

A escalabilidade do Aurora Serverless v2 é mais responsiva, mais granular e menos disruptivo da que a escalabilidade do Aurora Serverless v1. O Aurora Serverless v2 pode ser escalado alterando o tamanho da instância de banco de dados e adicionando mais instâncias ao cluster de banco de dados. Ele também pode ser escalado adicionando clusters em outro Regiões da AWSa um banco de dados global do Aurora. Em contrapartida, o Aurora Serverless v1 só é escalado aumentando ou diminuindo a capacidade do gravador. Toda a computação para um cluster do Aurora Serverless v1 é executada em uma única zona de disponibilidade e uma única Região da AWS.

Recurso de escalabilidade e alta disponibilidade	Comportamento do Aurora Serverless v2	Comportamento do Aurora Serverless v1
Unidades de capacidade mínima do Aurora (ACUs) (Aurora MySQL)	0,5 quando o cluster estiver em execução, 0 quando o cluster estiver pausado.	1 quando o cluster estiver em execução, 0 quando o cluster estiver pausado.
Mínimo de ACUs (Aurora PostgreSQL)	0,5 quando o cluster estiver em execução, 0 quando o cluster estiver pausado.	2 quando o cluster estiver em execução, 0 quando o cluster estiver pausado.
Máximo de ACUs (Aurora MySQL)	256	256

Recurso de escalabilidade e alta disponibilidade	Comportamento do Aurora Serverless v2	Comportamento do Aurora Serverless v1
Mínimo de ACUs (Aurora PostgreSQL)	256	384
Interromper um cluster	Você pode interromper e iniciar manualmente o cluster usando o mesmo recurso de parada e início de cluster que os clusters provisionados.	O cluster é pausado automatic amente após um tempo limite. Leva algum tempo para se tornar disponível quando a atividade é retomada.
Escalabilidade para instâncias de banco de dados	Aumente e reduza a escala verticalmente com incremento mínimo de 0,5 ACU.	Aumente e reduza a escala verticalmente dobrando as ACUs ou diminuindo-as pela metade.
Número de instâncias de banco de dado	O mesmo que um cluster provisionado: uma instância de banco de dados do gravador, até 15 instâncias de banco de dados do leitor.	Uma instância de banco de dados que processa leituras e gravações.
A escalabilidade pode acontecer enquanto as instruções SQL estão em execução?	Sim. O Aurora Serverless v2 não precisa esperar por um ponto de silêncio.	Não. Por exemplo, a escalabil idade aguarda a conclusão de transações de longa duração, tabelas temporárias e bloqueios de tabela.
As instâncias de banco de dados do leitor são escaladas com o gravador	Opcional	Não aplicável
Armazenamento máximo	128 TiB	128 TiB
Cache de buffer preservado na escalabilidade	Sim. O cache de buffer é redimensionado dinamicam ente.	Não. O cache de buffer é reaquecido após a escalabil idade.

Recurso de escalabilidade e alta disponibilidade	Comportamento do Aurora Serverless v2	Comportamento do Aurora Serverless v1
Failover	Sim, o mesmo que para clusters provisionados.	Apenas o melhor esforço, sujeito à disponibilidade de capacidade. Mais lento do que em Aurora Serverless v2.
Capacidade multi-AZ	Sim, o mesmo que para provisionados. Um cluster multi-AZ requer uma instância de banco de dados do leitor em uma segunda zona de disponibilidade (AZ). Para um cluster multi-AZ, o Aurora realiza failover multi-AZ em caso de falha da AZ.	Aurora Serverless v1Os clusters do executam toda a computação em uma única AZ. A recuperação em caso de falha da AZ é apenas o melhor esforço e está sujeita à disponibilidade de capacidade.
Bancos de dados globais do Aurora	Sim	Não
Escalabilidade baseada na pressão da memória	Sim	Não
Escalabilidade com base na carga da CPU	Sim	Sim
Escalabilidade com base no tráfego de rede	Sim, com base na sobrecarg a de memória e CPU do tráfego de rede. O parâmetro `max_connections` permanece constante para evitar a queda de conexões ao diminuir a escala na vertical.	Sim, com base no número de conexões.
Ação de tempo limite para eventos de escalabilidade	Não	Sim

Recurso de escalabilidade e alta disponibilidade	Comportamento do Aurora Serverless v2	Comportamento do Aurora Serverless v1
Adicionar novas instâncias de banco de dados ao cluster por meio do AWS Auto Scaling	Não aplicável. É possível criar instâncias de banco de dados do leitor do Aurora Serverless v2 em níveis de promoção 2 a 15 e deixá–las com a escala reduzida na vertical para baixa capacidade.	Não. Instâncias de banco de dados do leitor não estão disponíveis.

Comparação de compatibilidade de recursos entre o Aurora Serverless v2 e o Aurora Serverless v1

A tabela a seguir resume o seguinte:

- Recursos que estão disponíveis no Aurora Serverless v2, mas não no Aurora Serverless v1
- Recursos que funcionam de forma diferente entre o Aurora Serverless v1 e o Aurora Serverless v2
- Recursos que não estão disponíveis no momento no Aurora Serverless v2

Aurora Serverless v2 O inclui muitos recursos de clusters provisionados que não estão disponíveis para o Aurora Serverless v1.

Recurso	Aurora Serverless v2Suporte a	Aurora Serverless v1Suporte a
Topologia de cluster	Aurora Serverless v2O é uma propriedade de instâncias de banco de dados individuais. Um cluster pode conter várias instâncias de banco de dados do Aurora Serverless v2 ou uma combinação de instâncias de banco de dados provisionadas e do Aurora Serverless v2.	Aurora Serverless v1Clusters do não usam a noção de instâncias de banco de dados. Não será possível alterar a propriedade Aurora Serverless v1 depois de criar o cluster.

Recurso	Aurora Serverless v2Suporte a	Aurora Serverless v1Suporte a
Parâmetros de configuração	Quase todos os mesmos parâmetros podem ser modificados como em clusters provisionados. Para obter detalhes, consulte Trabalhar com grupos de parâmetros para o Aurora Serverless v2.	Apenas um subconjunto de parâmetros pode ser modificado.
Grupos de parâmetros	Grupo de parâmetros de cluster e grupos de parâmetros de banco de dados. Parâmetros com o valor `provisioned` no atributo `SupportedEngineModes` estão disponíveis. São muito mais parâmetros do que no Aurora Serverless v1.	Somente o grupo de parâmetros do cluster. Parâmetros com o valor `serverless` no atributo `SupportedEngineModes` estão disponíveis.
Criptografia do volume do cluster	Opcional	Obrigatório. As limitações no Limitações dos clusters de banco de dados criptografados do Amazon Aurora se aplicam a todos os clusters do Aurora Serverless v1.
Snapshots entre regiões	Sim	O snapshot deve ser criptografado com sua própria chave AWS Key Management Service (AWS KMS).
Backups automatizados retidos após a exclusão do cluster de banco de dados	Sim	Não

Recurso	Aurora Serverless v2Suporte a	Aurora Serverless v1Suporte a
TLS/SSL	Sim. O suporte é o mesmo que para clusters provisionados. Para ter mais informações, consulte Usar TLS/SSL com o Aurora Serverless v2.	Sim. Existem algumas diferenças em relação à compatibilidade com o TLS para clusters provisionados. Para ter mais informações, consulte Usar TLS/SSL com o Aurora Serverless v1.
Clonagem	Somente entre versões do mecanismo de banco de dados compatíveis com o Aurora Serverless v2. Você não pode usar a clonagem para realizar a atualização a partir do Aurora Serverless v1 nem de uma versão anterior de um cluster provisionado.	Somente entre versões do mecanismo de banco de dados compatíveis com o Aurora Serverless v1.
Integração com o Amazon S3	Sim	Sim
Integração com AWS Secrets Manager	Sim	Não
Exportar snapshots de cluster de banco de dados para o S3	Sim	Não
Associar um perfil do IAM	Sim	Não
Carregar logs do no Amazon CloudWatch	Opcional. Você escolhe quais logs serão ativados e quais logs devem ser carregados para o CloudWatch.	Todos os logs ativados são carregados para o CloudWatch automaticamente.
API de dados disponível	Sim	Sim
Editor de consultas disponível	Sim	Sim

Recurso	Aurora Serverless v2Suporte a	Aurora Serverless v1Suporte a
Performance Insights	Sim	Não
Proxy do Amazon RDS disponível	Sim	Não
Babelfish para Aurora PostgreSQL disponível	Sim. Suporte para as versões do Aurora PostgreSQL que são compatíveis com Babelfish e Aurora Serverless v2.	Não

Adaptar casos de uso do Aurora Serverless v1 para o Aurora Serverless v2

Dependendo do caso de uso do Aurora Serverless v1, você pode adaptar essa abordagem para aproveitar os recursos do Aurora Serverless v2 da forma a seguir.

Suponha que você tenha um cluster do Aurora Serverless v1 levemente carregado e sua prioridade é manter a disponibilidade contínua enquanto minimiza os custos. Com o Aurora Serverless v2, é possível configurar uma configuração mínima de ACU menor de 0,5, em comparação com um mínimo de 1 ACU no Aurora Serverless v1. É possível aumentar a disponibilidade criando uma configuração multi-AZ, com a instância de banco de dados do leitor também tendo um mínimo de 0,5 ACUs.

Suponha que você tenha um cluster do Aurora Serverless v1 que usa em um cenário de desenvolvimento e teste. Nesse caso, o custo também é uma alta prioridade, mas o cluster não precisa estar disponível o tempo todo. Aurora Serverless v2 pode pausar automaticamente cada instância quando ela estiver completamente ociosa. Você faz isso especificando uma capacidade mínima de 0 ACUs para o cluster, conforme explicado em Escalar para zero ACUs com pausa e retomada automáticas no Aurora Serverless v2. Você também pode interromper manualmente o cluster quando ele não for necessário e iniciá-lo quando chegar a hora do próximo ciclo de teste ou desenvolvimento.

Suponha que você tenha um cluster do Aurora Serverless v1 com uma workload pesada. Um cluster equivalente usando o Aurora Serverless v2 pode ser escalado com mais granularidade. Por exemplo, o Aurora Serverless v1 é escalado dobrando a capacidade, por exemplo, de 64 para 128 ACUs. Já

a instância de banco de dados do Aurora Serverless v2 pode ser escalada em incrementos de 0,5 ACU.

Suponha que sua workload exija uma capacidade total maior do que a disponível no Aurora Serverless v1. Você pode usar várias instâncias de banco de dados do leitor do Aurora Serverless v2 para descarregar as partes de uso intenso de leitura da workload da instância de banco de dados do gravador. Você também pode dividir a workload de uso intenso de leitura entre várias instâncias de banco de dados do leitor.

Para uma workload de uso intenso de gravação, é possível configurar o cluster com uma grande instância de banco de dados provisionada como o gravador. Você pode fazer isso com uma ou mais instâncias de banco de dados do leitor do Aurora Serverless v2.

Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

O processo de atualização de um cluster de banco de dados do Aurora Serverless v1 para o Aurora Serverless v2 tem várias etapas. Isso porque não é possível fazer a conversão diretamente de Aurora Serverless v1 paraAurora Serverless v2. Sempre há uma etapa intermediária que envolve a conversão do cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado.

Clusters de banco de dados compatíveis com o Aurora MySQL

É possível converter o cluster de banco de dados do Aurora Serverless v1 em um cluster de banco de dados provisionado e, depois, usar uma implantação azul/verde para atualizá-lo e convertê-lo em um cluster de banco de dados do Aurora Serverless v2. Recomendamos esse procedimento para

ambientes de produção. Para obter mais informações, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Como utilizar uma implantação azul/verde para atualizar um cluster do Aurora Serverless v1 que execute o Aurora MySQL versão 2 (compatível com o MySQL 5.7)

1. Converta o cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado do Aurora MySQL versão 2. Siga o procedimento em Conversão de Aurora Serverless v1 em provisionado.
2. Crie uma implantação azul/verde. Siga o procedimento em Criar uma implantação azul/verde no Amazon Aurora.
3. Escolha uma versão do Aurora MySQL para o cluster verde que seja compatível com o Aurora Serverless v2, por exemplo, 3.04.1.

   Para saber as versões compatíveis, consulte Aurora Serverless v2 com o Aurora MySQL.
4. Modifique a instância de banco de dados de gravador do cluster verde para usar a classe de instância de banco de dados do Serverless v2 (db.serverless).

   Para obter detalhes, consulte Converter um gravador ou leitor provisionado em Aurora Serverless v2.
5. Quando o cluster de banco de dados do Aurora Serverless v2 atualizado estiver disponível, mude do cluster azul para o cluster verde.

Clusters de banco de dados compatíveis com o Aurora PostgreSQL

É possível converter o cluster de banco de dados do Aurora Serverless v1 em um cluster de banco de dados provisionado e, depois, usar uma implantação azul/verde para atualizá-lo e convertê-lo em um cluster de banco de dados do Aurora Serverless v2. Recomendamos esse procedimento para ambientes de produção. Para obter mais informações, consulte Usar implantações azul/verde do Amazon Aurora para atualizações de banco de dados.

Como utilizar uma implantação azul/verde para atualizar um cluster do Aurora Serverless v1 que execute o Aurora PostgreSQL versão 11

1. Converta o cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado do Aurora PostgreSQL. Siga o procedimento em Conversão de Aurora Serverless v1 em provisionado.

2. Crie uma implantação azul/verde. Siga o procedimento em Criar uma implantação azul/verde no Amazon Aurora.
3. Escolha uma versão do Aurora PostgreSQL para o cluster verde que seja compatível com o Aurora Serverless v2, por exemplo, 15.3.

   Para saber as versões compatíveis, consulte Aurora Serverless v2 com o Aurora PostgreSQL.
4. Modifique a instância de banco de dados de gravador do cluster verde para usar a classe de instância de banco de dados do Serverless v2 (db.serverless).

   Para obter detalhes, consulte Converter um gravador ou leitor provisionado em Aurora Serverless v2.
5. Quando o cluster de banco de dados do Aurora Serverless v2 atualizado estiver disponível, mude do cluster azul para o cluster verde.

Também é possível atualizar o cluster de banco de dados do Aurora Serverless v1 diretamente do Aurora PostgreSQL versão 11 para a versão 13, convertê-lo em um cluster de banco de dados provisionado e, depois, converter o cluster provisionado em um cluster de banco de dados do Aurora Serverless v2.

Como atualizar e, depois, converter um cluster do Aurora Serverless v1 que execute o Aurora PostgreSQL versão 11

1. Converta o cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado do Aurora PostgreSQL. Siga o procedimento em Conversão de Aurora Serverless v1 em provisionado.
2. Atualize o cluster do Aurora Serverless v1 para uma versão do Aurora PostgreSQL versão 13 compatível com o Aurora Serverless v2, por exemplo, 13.12. Siga o procedimento em Atualizar a versão principal.

   Para saber as versões compatíveis, consulte Aurora Serverless v2 com o Aurora PostgreSQL.
3. Adicione uma instância de banco de dados de leitor do Aurora Serverless v2 ao cluster. Para obter mais informações, consulte Adicionar um leitor do Aurora Serverless v2.
4. Faça o failover para a instância de banco de dados do Aurora Serverless v2:

   a. Selecione a instância de banco de dados de gravador do cluster de banco de dados.
   b. Em Actions (Ações), selecione Failover.
   c. Na página de confirmação, selecione Failover.

Para clusters de banco de dados do Aurora Serverless v1 que executam o Aurora PostgreSQL versão 13, você deve converter o cluster do Aurora Serverless v1 em um cluster de banco de dados provisionado e, depois, converter o cluster provisionado em um cluster de banco de dados do Aurora Serverless v2.

Como atualizar um cluster do Aurora Serverless v1 executando o Aurora PostgreSQL versão 13

1. Converta o cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado do Aurora PostgreSQL. Siga o procedimento em Conversão de Aurora Serverless v1 em provisionado.
2. Adicione uma instância de banco de dados de leitor do Aurora Serverless v2 ao cluster. Para obter mais informações, consulte Adicionar um leitor do Aurora Serverless v2.
3. Faça o failover para a instância de banco de dados do Aurora Serverless v2:
    a. Selecione a instância de banco de dados de gravador do cluster de banco de dados.
    b. Em Actions (Ações), selecione Failover.
    c. Na página de confirmação, selecione Failover.
4. Remova a instância do leitor.

Migrar de um banco de dados on-premises para o Aurora Serverless v2

Você pode migrar seus bancos de dados on-premises para o Aurora Serverless v2, assim como faz com Aurora MySQL e Aurora PostgreSQL provisionados.

- Para bancos de dados MySQL, use o comando `mysqldump`. Para acessar mais informações, consulte Importing data to an Amazon RDS for MySQL database with reduced downtime no Guia do usuário do Amazon Relational Database Service.
- Para bancos de dados PostgreSQL, use os comandos `pg_dump` e `pg_restore`. Para ter mais informações, consulte a publicação no blog "Best practices for migrating PostgreSQL databases to Amazon RDS and Amazon Aurora" (Práticas recomendadas para migrar bancos de dados PostgreSQL para o Amazon RDS e o Amazon Aurora).

Usar o Amazon Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

O Amazon Aurora Serverless v1 (Amazon Aurora Serverless versão 1) é uma configuração sob demanda e de autoescalabilidade do Amazon Aurora. Um cluster de banco de dados do Aurora Serverless v1 é um cluster de banco de dados que dimensiona a capacidade computacional com base nas necessidades de sua aplicação. Isso contrasta com os clusters de banco de dados provisionados do Aurora, para os quais você gerencia a capacidade manualmente. O Aurora Serverless v1 fornece uma opção relativamente simples e econômica para workloads pouco frequentes, intermitentes ou imprevisíveis. Ele é econômico porque automaticamente inicia, escala a capacidade computacional para corresponder ao uso de sua aplicação e desliga quando não está em uso.

Para saber mais sobre preços, consulte Preços de recursos sem servidor em Edição compatível com MySQL ou Edição compatível com PostgreSQL na página Amazon Aurora pricing.

Aurora Serverless v1Os clusters do têm o mesmo tipo de volume de armazenamento de alta capacidade, distribuído e altamente disponível que é usado por clusters de banco de dados provisionados.

Para um cluster do Aurora Serverless v1, o volume do cluster é sempre criptografado. É possível escolher a chave de criptografia, mas não é possível desabilitar a criptografia. Isso significa que você pode executar as mesmas operações em um Aurora Serverless v1 que você pode em snapshots criptografados. Para obter mais informações, consulte Aurora Serverless v1 e snapshots.

Tópicos

- Disponibilidade de regiões e versões do Aurora Serverless v1
- Vantagens do Aurora Serverless v1
- Casos de uso do Aurora Serverless v1
- Limitações do Aurora Serverless v1
- Requisitos de configuração do Aurora Serverless v1
- Usar TLS/SSL com o Aurora Serverless v1
- Como funciona o Aurora Serverless v1
- Criar um cluster de banco de dados do Aurora Serverless v1
- Restaurar um cluster de banco de dados do Aurora Serverless v1
- Modificar um cluster de banco de dados do Aurora Serverless v1
- Dimensionar manualmente a capacidade do cluster de banco de dados do Aurora Serverless v1
- Visualizar clusters de banco de dados do Aurora Serverless v1
- Excluir um cluster de banco de dados do Aurora Serverless v1
- Aurora Serverless v1Versões de mecanismos de banco de dados do e do Aurora

Disponibilidade de regiões e versões do Aurora Serverless v1

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS. Para obter mais informações sobre a disponibilidade de versões e regiões com o Aurora e Aurora Serverless v1, consulte Aurora Serverless v1 da.

Vantagens do Aurora Serverless v1

Aurora Serverless v1O oferece as seguintes vantagens:

- Mais simples do que o modelo provisionado: o Aurora Serverless v1 elimina muito da complexidade e gerenciar instâncias de banco de dados e capacidade.
- Escalável: o Aurora Serverless v1 escala perfeitamente a capacidade de computação e de memória, conforme necessário, sem interrupções nas conexões do cliente.
- Econômico: ao usar o Aurora Serverless v1, você paga apenas pelos recursos de banco de dados que consome, por segundo.

- Armazenamento de alta disponibilidade: o Aurora Serverless v1 usa o mesmo sistema de armazenamento distribuído e tolerante a falhas com replicação de seis maneiras que o Aurora para se proteger contra a perda de dados.

Casos de uso do Aurora Serverless v1

Aurora Serverless v1O foi projetado para os seguintes casos de uso:

- Aplicações usadas com pouca frequência:você tem uma aplicação que é usada somente por alguns minutos várias vezes por dia ou por semana, como um site de blog de volume baixo. Com o Aurora Serverless v1, você paga apenas pelos recursos de banco de dados que consome, por segundo.
- Novas aplicações:você está implantando uma nova aplicação e não tem certeza sobre o tamanho da instância de que precisa. Com o Aurora Serverless v1, é possível criar um endpoint de banco de dados e fazer a escalabilidade automática para os requisitos de capacidade do banco de dados de sua aplicação.
- Workloads variáveis:você está executando uma aplicação pouco usada, com picos de 30 minutos a várias horas algumas vezes por dia ou várias vezes por ano. Os exemplos são aplicações de recursos humanos, orçamentos e aplicações de relatórios operacionais. Com o Aurora Serverless v1, não é mais necessário provisionar para capacidade de pico ou média.
- Workloads imprevisíveis – você está executando workloads diárias que têm aumentos repentinos e imprevisíveis na atividade. Um exemplo é um site de tráfego que tem um surto de atividades quando começa a chover. Com o Aurora Serverless v1, seu banco de dados faz escalabilidade automática da capacidade para atender às necessidades da carga de pico da aplicação e escala novamente para reduzir a capacidade quando o surto de atividades acaba.
- Bancos de dados de desenvolvimento e teste – os desenvolvedores usam os bancos de dados durante as horas de trabalho, mas não precisam deles à noite ou em fins de semana. Com o Aurora Serverless v1, seu banco de dados é desligado automaticamente quando não está em uso.
- Multi-tenant applications (Aplicações multilocatários): com o Aurora Serverless v1, não é necessário gerenciar individualmente a capacidade do banco de dados para cada aplicação em sua frota. O Aurora Serverless v1 gerencia a capacidade do banco de dados individual para você.

Limitações do Aurora Serverless v1

As limitações a seguir se aplicam ao Aurora Serverless v1:

- **Important**

  A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

- Aurora Serverless v1O não é compatível com os seguintes recursos:
  - Bancos de dados globais do Aurora
  - Réplicas do Aurora
  - AWS Identity and Access ManagementAutenticação do banco de dados do (IAM)
  - Retroceder no Aurora
  - Fluxos de atividades do banco de dados
  - Autenticação de Kerberos
  - Insights de Performance
  - RDS Proxy
  - Visualizar logs no Console de gerenciamento da AWS
- As conexões com um cluster de banco de dados do Aurora Serverless v1 são encerradas automaticamente se permanecerem abertas por mais de um dia.
- Todos os clusters de banco de dados Aurora Serverless v1 têm as seguintes limitações:
  - Não é possível exportar snapshots do Aurora Serverless v1 para buckets do Amazon S3.
  - Você não pode usar AWS Database Migration Service e Captura de dados de alteração (CDC) com clusters de banco de dados Aurora Serverless v1. Somente os clusters de bancos de dados Aurora provisionados oferecem suporte ao CDC tendo o AWS DMS como origem.
  - Não é possível salvar dados em arquivos de texto no Amazon S3 nem carregar dados de arquivos de texto para o Aurora Serverless v1 do S3.
  - Não é possível anexar um perfil do IAM a um cluster de banco de dados do Aurora Serverless v1. No entanto, você pode carregar dados no Aurora Serverless v1, por meio do Amazon S3, usando a extensão `aws_s3` com a função `aws_s3.table_import_from_s3` e o parâmetro

`credentials`. Para obter mais informações, consulte Importar dados do Amazon S3 para um cluster de banco de dados do Aurora PostgreSQL.

- Ao usar o editor de consultas, um segredo do Secrets Manager é criado para que as credenciais do banco de dados acessem o banco de dados. Se você excluir as credenciais do editor de consultas, o segredo associado também será excluído do Secrets Manager. Não é possível recuperar esse segredo após sua exclusão.

- Os clusters de banco de dados baseados em Aurora MySQL que executam o Aurora Serverless v1 não são compatíveis com:
  - Invocar funções do AWS Lambda de dentro do cluster de bancos de dados Aurora MySQL. No entanto, as funções do AWS Lambda podem fazer chamadas para o cluster de banco de dados do Aurora Serverless v1.
  - Restauração de um snapshot de uma instância de banco de dados que não seja do Aurora MySQL ou RDS para MySQL.
  - Replicação de dados usando replicação com base em logs binários (binlogs). Essa limitação é verdadeira, não importa se o cluster de banco de dados do Aurora Serverless v1 baseado em Aurora MySQL é a origem ou o destino da replicação. Para replicar dados em um cluster de banco de dados do Aurora Serverless v1 de uma instância de banco de dados MySQL fora do Aurora, como uma em execução no Amazon EC2, avalie a possibilidade de usar o AWS Database Migration Service. Para obter mais informações, consulte o Guia do usuário do AWS Database Migration Service.
  - Criar usuários com acesso baseado em host (`'username'@'IP_address'`). Isso ocorre porque o Aurora Serverless v1 usa uma frota de roteadores entre o cliente e o host do banco de dados para uma escalabilidade perfeita. O endereço IP que o cluster de banco de dados do Aurora Serverless vê é o do host do roteador, não do seu cliente. Para obter mais informações, consulte Aurora Serverless v1Arquitetura do.

    Em vez disso, use o caractere curinga (`'username'@'%'`).

- Os clusters de banco de dados baseados no Aurora PostgreSQL que executam o Aurora Serverless v1 têm as seguintes limitações:
  - O gerenciamento de planos de consulta do Aurora PostgreSQL (extensão `apg_plan_management`) não é aceito.
  - Não há suporte ao recurso de replicação lógica disponível no Amazon RDS PostgreSQL e no Aurora PostgreSQL.
  - Não há suporte a comunicações de saída como as habilitadas pelas extensões do Amazon RDS para PostgreSQL. Por exemplo, não é possível acessar dados externos com a extensão

`postgres_fdw/dblink`. Para obter mais informações sobre as extensões PostgreSQL do RDS, consulte PostgreSQL no Amazon RDS no Guia do usuário do RDS.

- Atualmente, determinadas consultas e comandos SQL não são recomendados. Estes incluem bloqueios de aconselhamento em nível de sessão, relações temporárias, notificações assíncronas (`LISTEN`) e cursores com retenção (`DECLARE` *`name`* `... CURSOR WITH HOLD FOR` *`query`*). Além disso, os comandos `NOTIFY` impedem a escalabilidade e não são recomendados.

  Para obter mais informações, consulte Autoscaling for Aurora Serverless v1.

- Não é possível definir a janela de backup automatizado preferencial para um cluster de banco de dados do Aurora Serverless v1.
- Você pode definir a janela de manutenção para um cluster de banco de dados do Aurora Serverless v1. Para obter mais informações, consulte Ajustar a janela de manutenção do cluster de banco de dados preferencial.

Requisitos de configuração do Aurora Serverless v1

Ao criar um cluster de banco de dados do Aurora Serverless v1, preste atenção aos seguintes requisitos:

- Use estes números de porta específicos para cada mecanismo de banco de dados:
  - Aurora MySQL – `3306`
  - Aurora PostgreSQL – `5432`
- Crie seu cluster de banco de dados do Aurora Serverless v1 em uma nuvem privada virtual (VPC) com base no serviço da Amazon VPC. Ao criar um cluster de banco de dados do Aurora Serverless v1 em sua VPC, você consome 2 (dois) dos 50 (cinquenta) endpoints de interface e do balanceador de carga do Gateway alocados à VPC. Esses endpoints são criados automaticamente para você. Para aumentar sua cota, você pode entrar em contato com Suporte. Para obter mais informações, consulte as Amazon VPC cotas.
- Não é possível fornecer um endereço IP público a um cluster de banco de dados do Aurora Serverless v1. Você pode acessar um cluster de banco de dados do Aurora Serverless v1 somente a partir de uma VPC.
- Crie sub-redes em diferentes zonas de disponibilidade para o grupo de sub-redes de banco de dados que você usa para seu cluster de banco de dados Aurora Serverless v1. Em outras palavras, não é possível ter mais do que uma sub-rede na mesma zona de disponibilidade.

- As alterações feitas em um grupo de sub-redes usado por um cluster de banco de dados do Aurora Serverless v1 não são aplicadas ao cluster.
- É possível acessar um cluster de banco de dados do Aurora Serverless v1 no AWS Lambda. Para isso, é necessário configurar a função do Lambda para ser executada na mesma VPC que o cluster de banco de dados do Aurora Serverless v1. Para obter mais informações sobre como trabalhar com o AWS Lambda, consulte Configurar uma função do Lambda para acessar recursos em uma Amazon VPC no Guia do desenvolvedor do AWS Lambda.

Usar TLS/SSL com o Aurora Serverless v1

Por padrão, o Aurora Serverless v1 usa o protocolo Transport Layer Security/Secure Sockets Layer (TLS/SSL) para criptografar as comunicações entre clientes e seu cluster de banco de dados Aurora Serverless v1. Ele é compatível com TLS/SSL versões 1.0, 1.1 e 1.2. Você não precisa configurar seu cluster de banco de dados do Aurora Serverless v1 para usar TLS/SSL.

No entanto, as seguintes limitações se aplicam:

- A compatibilidade com TLS/SSL para clusters de banco de dados do Aurora Serverless v1 não está disponível atualmente na região China (Pequim) da Região da AWS.
- Quando você cria usuários de banco de dados para um cluster de banco de dados do Aurora Serverless v1 baseado no Aurora MySQL, não use a cláusula `REQUIRE` para permissões SSL. Isso impede que os usuários se conectem à instância de bancos de dados Aurora.
- Para os utilitários MySQL Client e PostgreSQL Client, as variáveis de sessão que você pode usar em outros ambientes não têm efeito ao usar TLS/SSL entre o cliente e o Aurora Serverless v1.
- Para o MySQL Client, ao se conectar com o modo `VERIFY_IDENTITY` do TLS/SSL, atualmente você precisa usar o comando `mysql` compatível com o MySQL 8.0. Para obter mais informações, consulte Conexão a uma instância de banco de dados executando o mecanismo de banco de dados do MySQL.

Dependendo do cliente usado para se conectar ao cluster de banco de dados do Aurora Serverless v1, talvez você não precise especificar TLS/SSL para obter uma conexão criptografada. Por exemplo, para usar o cliente PostgreSQL para se conectar a um cluster de banco de dados do Aurora Serverless v1 que executa o Aurora Edição compatível com PostgreSQL, conecte-se como você normalmente faz.

```
psql -h endpoint -U user
```

Depois de digitar sua senha, o cliente PostgreSQL mostra que você vê os detalhes da conexão, incluindo a versão TLS/SSL e a cifra.

```
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1), server 10.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.
```

> **Important**
>
> Aurora Serverless v1O usa o protocolo TLS/SSL (Transport Layer Security/Secure Sockets Layer) para criptografar conexões por padrão, a menos que o TLS/SSL seja desabilitado pela aplicação cliente. A conexão TLS/SSL termina na frota de roteadores. A comunicação entre a frota de roteadores e seu cluster de banco de dados do Aurora Serverless v1 ocorre dentro do limite interno da rede do serviço.
>
> É possível conferir o status da conexão do cliente para examinar se a conexão com o Aurora Serverless v1 está criptografada com TLS/SSL. As tabelas `pg_stat_ssl` e `pg_stat_activity` do PostgreSQL e sua função `ssl_is_used` não mostram o estado TLS/SSL da comunicação entre a aplicação cliente e o Aurora Serverless v1. Da mesma forma, o estado TLS/SSL não pode ser derivado da instrução `status` do MySQL. Anteriormente, os parâmetros de cluster do Aurora `force_ssl` para PostgreSQL e `require_secure_transport` para MySQL não eram compatíveis com o Aurora Serverless v1. Esses parâmetros já estão disponíveis para o Aurora Serverless v1. Para obter uma lista completa dos parâmetros compatíveis com o Aurora Serverless v1, chame a operação de API DescribeEngineDefaultClusterParameters. Para obter mais informações sobre grupos de parâmetros e o Aurora Serverless v1, consulte Grupos de parâmetros para Aurora Serverless v1.

Para usar o MySQL Client para se conectar a um cluster de banco de dados do Aurora Serverless v1 que executa o Aurora Edição compatível com MySQL, especifique o TLS/SSL em sua solicitação. O exemplo a seguir inclui o armazenamento de confiança CA 1 raiz da Amazon baixado do Amazon Trust Services, que é necessário para que essa conexão seja bem-sucedida.

```
mysql -h endpoint -P 3306 -u user -p --ssl-ca=amazon-root-CA-1.pem --ssl-mode=REQUIRED
```

Insira sua senha quando for solicitado. Logo, o monitor MySQL será aberto. Você pode confirmar que a sessão está criptografada usando o comando `status`.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.62, for Linux (x86_64) using readline 5.1
Connection id:          19
Current database:
Current user:           ***@*******
SSL:                    Cipher in use is ECDHE-RSA-AES256-SHA
...
```

Para saber mais sobre como se conectar ao banco de dados do Aurora MySQL com o cliente MySQL, consulte Conectar-se a uma instância de banco de dados que esteja executando o mecanismo de banco de dados MySQL..

O Aurora Serverless v1 é compatível com todos os modos TLS/SSL disponíveis para o cliente do MySQL (`mysql`) e o cliente do PostgreSQL (`psql`), incluindo aqueles listados na tabela a seguir.

Descrição do modo TLS/SSL	mysql	psql
Conectar sem usar TLS/SSL.	DISABLED	desabilitar
Tente a conexão usando TLS/SSL primeiro, mas volte para não SSL, se necessário.	PREFERRED	preferir (padrão)
Imponha o uso de TLS/SSL.	REQUIRED	require
Imponha o TLS/SSL e verifique a CA.	VERIFY_CA	verify-ca
Imponha o TLS/SSL, verifique a CA e verifique o hostname da CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v1O usa certificados curinga. Se você especificar a opção “verificar CA” ou “verificar CA e nome do host da CA” ao usar TLS/SSL, primeiro baixe o armazenamento de confiança CA 1 raiz da Amazon no Amazon Trust Services. Depois de fazer isso, você pode identificar esse arquivo formatado em PEM em seu comando client. Para fazer isso usando o PostgreSQL Client:

Para Linux, macOS ou Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Para saber mais sobre como trabalhar com o banco de dados do Aurora PostgreSQL usando o cliente do Postgres, consulte Conectar-se a uma instância de banco de dados que esteja executando o mecanismo de banco de dados do PostgreSQL..

Para obter mais informações sobre como se conectar a clusters de bancos de dados Aurora em geral, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Pacotes de cifras compatíveis com clusters de banco de dados do Aurora Serverless v1

Usando conjuntos de cifras configuráveis, você pode ter mais controle sobre a segurança de suas conexões de banco de dados. É possível especificar uma lista de conjuntos de cifras que você deseja permitir para proteger conexões TLS/SSL do cliente com o banco de dados. Com conjuntos de cifras configuráveis, você pode controlar a criptografia de conexão aceita pelo servidor de banco de dados. Isso impede o uso de cifras que não são seguras ou que não são mais usadas.

Aurora Serverless v1Os clusters de banco de dados do que são baseados no Aurora MySQL são compatíveis com os mesmos conjuntos de cifras que os clusters de banco de dados provisionados pelo Aurora MySQL. Para obter informações sobre esses conjuntos de cifras, consulte Configurar conjuntos de criptografia para conexões com clusters de banco de dados do Aurora MySQL.

Aurora Serverless v1Os clusters de banco de dados do baseados no Aurora PostgreSQL não são compatíveis com conjuntos de cifras.

Como funciona o Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster

do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

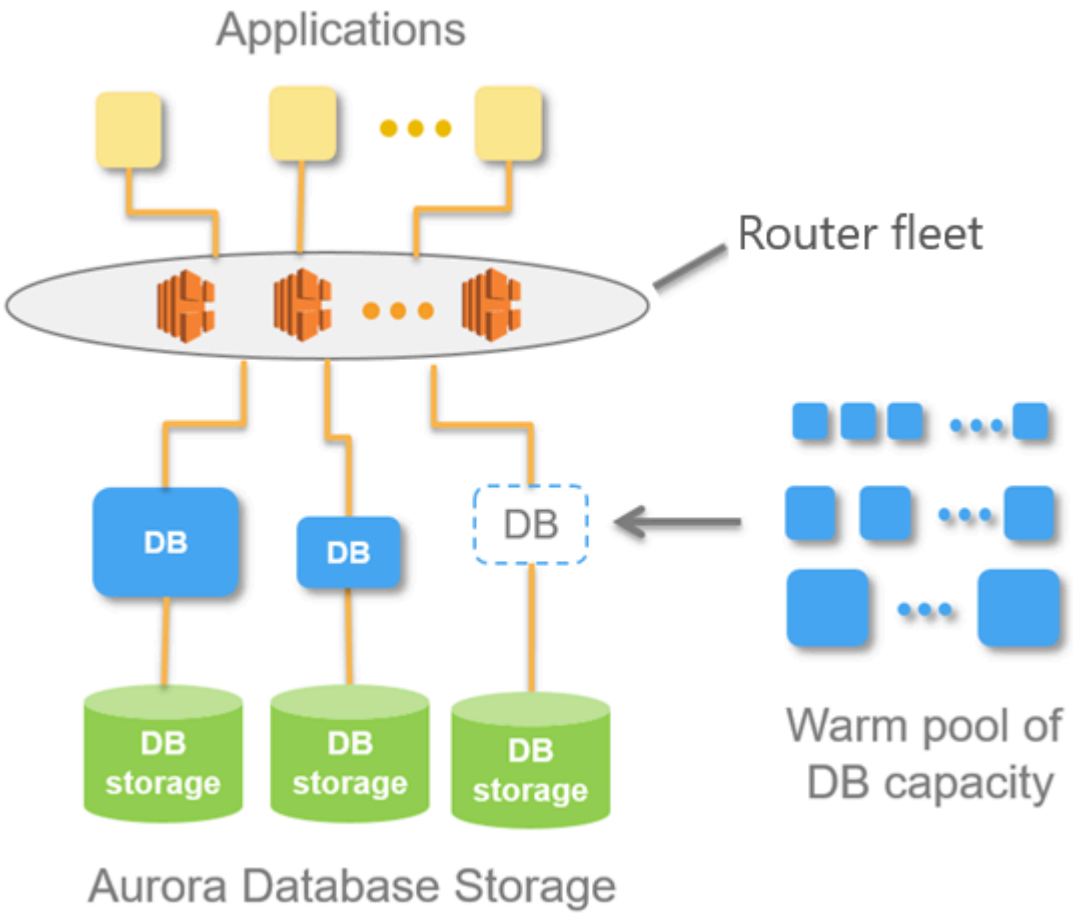
A seguir, saiba como funciona o Aurora Serverless v1.

Tópicos

- Aurora Serverless v1Arquitetura do
- Autoscaling for Aurora Serverless v1
- Ação de tempo limite para alterações na capacidade
- Pausa e retomada para o Aurora Serverless v1
- Determinar o número máximo de conexões de banco de dados do Aurora Serverless v1
- Grupos de parâmetros para Aurora Serverless v1
- Registro em log para o Aurora Serverless v1
- Aurora Serverless v1 e manutenção
- Aurora Serverless v1 e failover
- Aurora Serverless v1 e snapshots

Aurora Serverless v1Arquitetura do

A imagem a seguir mostra uma visão geral da arquitetura do Aurora Serverless v1.



Em vez de provisionar e gerenciar servidores de banco de dados, você especifica capacity units (ACUs - unidades de capacidade) do Aurora. Cada ACU é uma combinação de aproximadamente 2 gigabytes (GB) de memória, CPU correspondente e rede. O armazenamento do banco de dados escala automaticamente de 10 gibibytes (GiB) para 128 tebibytes (TiB), o mesmo que o armazenamento em um cluster de banco de dados padrão do Aurora.

Você pode especificar as ACUs mínima e máxima. A unidade de capacidade mínima do Aurora é a ACU mais baixa para a qual o cluster de banco de dados pode ser reduzido. A unidade de capacidade máxima do Aurora é a ACU mais alta para a qual o cluster de banco de dados pode ser expandido. Com base em suas configurações, o Aurora Serverless v1 cria automaticamente as regras de escalabilidade para os limites de utilização de CPU, conexões e memória disponível.

O Aurora Serverless v1 gerencia o grupo quente de recursos em uma Região da AWS para minimizar o tempo de escalabilidade. Quando o Aurora Serverless v1 adiciona recursos ao cluster de bancos de dados Aurora, ele usa a frota de roteadores para alternar as conexões de clientes ativas para os novos recursos. Em algum momento específico, você será cobrado somente pelas ACUs que estão sendo usadas ativamente em seu cluster de banco de dados do Aurora.

Autoscaling for Aurora Serverless v1

A capacidade alocada para o cluster de banco de dados do Aurora Serverless v1 é expandida ou reduzida perfeitamente com base na carga gerada pela aplicação cliente. Aqui, a carga é a utilização da CPU e a quantidade de conexões. Quando a capacidade é limitada por qualquer um desses, o Aurora Serverless v1 se expande. O Aurora Serverless v1 também se expande quando detecta problemas de performance que podem ser resolvidos ao fazê-lo.

Você pode visualizar eventos de escalabilidade de seu cluster do Aurora Serverless v1 no Console de gerenciamento da AWS. Durante a escalabilidade automática, o Aurora Serverless v1 redefine a métrica `EngineUptime`. O valor da métrica de redefinição não significa que a escalabilidade contínua tenha problemas nem que o Aurora Serverless v1 descartou conexões. É simplesmente o ponto de partida para o tempo de atividade na nova capacidade. Para saber mais sobre as métricas, consulte Monitorar métricas em um cluster do Amazon Aurora.

Quando o seu cluster de banco de dados do Aurora Serverless v1 não tem conexões ativas, ele pode reduzir até a capacidade zero (0 ACUs). Para saber mais, consulte Pausa e retomada para o Aurora Serverless v1.

Quando é preciso executar uma operação de escalabilidade, o Aurora Serverless v1 primeiro tenta identificar um ponto de escalabilidade, um momento em que nenhuma consulta esteja em processamento. O Aurora Serverless v1 talvez não consiga encontrar um ponto de escalabilidade pelos seguintes motivos:

- Consultas de longa duração
- Transações em andamento
- As tabelas temporárias ou bloqueios de tabela estão em uso

Para aumentar a taxa de sucesso em encontrar um ponto de escalabilidade de seu cluster de banco de dados Aurora Serverless v1, recomendamos evitar consultas e transações de longa duração. Para saber mais sobre operações que impedem a escalabilidade e como evitá-las, consulte Práticas recomendadas para trabalhar com o Aurora Serverless v1.

Por padrão, o Aurora Serverless v1 tenta encontrar um ponto de escalabilidade por cinco minutos (300 segundos). É possível especificar um período de tempo limite diferente ao criar ou modificar o cluster. O tempo limite pode ter entre 60 segundos e 10 minutos (600 segundos). Se o Aurora Serverless v1 não conseguir encontrar um ponto de escalabilidade no período especificado, o tempo limite da operação de autoescalabilidade é excedido.

Por padrão, se a autoescalabilidade não encontrar um ponto de escalabilidade antes do tempo limite, o Aurora Serverless v1 manterá o cluster na capacidade atual. Você pode alterar esse comportamento padrão ao criar ou modificar o seu cluster de banco de dados do Aurora Serverless v1 selecionando a opção Force the capacity change (Forçar a alteração da capacidade). Para ter mais informações, consulte Ação de tempo limite para alterações na capacidade.

Ação de tempo limite para alterações na capacidade

Se a autoescalabilidade ultrapassar o tempo limite sem encontrar um ponto de escalabilidade, por padrão, o Aurora manterá a capacidade atual. Você pode optar por deixar o Aurora forçar a alteração selecionando a opção Force the capacity change (Forçar a alteração de capacidade). Essa opção está disponível na seção Autoscaling timeout and action (Tempo limite e ação de autoescalabilidade) da página Create database (Criar banco de dados), quando você cria o cluster.

Por padrão, a opção Force the capacity change (Forçar a alteração de capacidade) não é selecionada. Deixe essa opção desmarcada para que a capacidade do cluster de banco de dados do Aurora Serverless v1 permaneça inalterada, se a operação de escalabilidade ultrapassar o tempo limite sem encontrar um ponto de escalabilidade.

Selecionar essa opção faz com que o cluster de banco de dados do Aurora Serverless v1 imponha a alteração de capacidade, mesmo sem um ponto de escalabilidade. Antes de selecionar esta opção, lembre-se consequências dessa seleção:

- Quaisquer transações em andamento são interrompidas e a seguinte mensagem de erro é exibida.

  Aurora MySQL versão 2 – ERRO 1105 (HY000): A última transação foi interrompida devido à escalabilidade simples. Tente novamente.

  Você pode reenviar as transações assim que seu cluster de banco de dados do Aurora Serverless v1 estiver disponível.
- As conexões com tabelas temporárias e bloqueios são descartadas.

  Recomendamos que você selecione a opção Force the capacity change (Forçar a alteração da capacidade) somente se sua aplicação puder se recuperar de conexões descartadas ou transações incompletas.

As escolhas que você faz no Console de gerenciamento da AWS ao criar um cluster de banco de dados do Aurora Serverless v1 são armazenadas no objeto `ScalingConfigurationInfo`,

nas propriedades `SecondsBeforeTimeout` e `TimeoutAction`. O valor da propriedade `TimeoutAction` é definido como um dos seguintes valores quando você cria o cluster:

- `RollbackCapacityChange`: este valor é definido quando você seleciona a opção Roll back the capacity change (Reverter a alteração na capacidade). Esse é o comportamento padrão.
- `ForceApplyCapacityChange`: este valor é definido quando você seleciona a opção Force the capacity change (Forçar a alteração da capacidade).

Você pode obter o valor dessa propriedade em um cluster de banco de dados do Aurora Serverless v1 existente usando o comando describe-db-clusters da AWS CLI, como mostrado a seguir.

Para Linux, macOS ou Unix:

```
aws rds describe-db-clusters --region region \
  --db-cluster-identifier your-cluster-name \
  --query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

Para Windows:

```
aws rds describe-db-clusters --region region ^
  --db-cluster-identifier your-cluster-name ^
  --query "*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}"
```

O exemplo a seguir mostra a consulta e a resposta de um cluster de banco de dados do Aurora Serverless v1 chamado `west-coast-sles` na região Oeste dos EUA (Norte da Califórnia).

```
$ aws rds describe-db-clusters --region us-west-1 --db-cluster-identifier west-coast-
sles
--query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'

[
    {
        "ScalingConfigurationInfo": {
            "MinCapacity": 1,
            "MaxCapacity": 64,
            "AutoPause": false,
            "SecondsBeforeTimeout": 300,
            "SecondsUntilAutoPause": 300,
            "TimeoutAction": "RollbackCapacityChange"
        }
    }
```

```
]
```

Como mostra a resposta, esse cluster de banco de dados do Aurora Serverless v1 usa a configuração padrão.

Para ter mais informações, consulte Criar um cluster de banco de dados do Aurora Serverless v1. Após criar seu Aurora Serverless v1, você também pode modificar a ação de tempo limite e outras configurações de capacidade a qualquer momento. Para saber como, consulte Modificar um cluster de banco de dados do Aurora Serverless v1.

Pausa e retomada para o Aurora Serverless v1

É possível escolher pausar o cluster de banco de dados do Aurora Serverless v1 depois de um certo tempo sem atividade. Você especifica o tempo sem atividade para que o cluster de banco seja pausado. Ao selecionar essa opção, o tempo de inatividade padrão é cinco minutos, mas é possível alterar esse valor. Esta definição é opcional.

Quando o cluster de banco é pausado, não ocorre nenhuma atividade de computação ou de memória, e você é cobrado somente pelo armazenamento. Se forem solicitadas conexões ao banco de dados quando um cluster de banco de dados do Aurora Serverless v1 estiver pausado, o cluster de banco de dados será retomado automaticamente e atenderá às solicitações de conexão.

Quando o cluster de banco de dados retoma a atividade, ele tem a mesma capacidade que tinha quando o Aurora pausou o cluster. O número de ACUs depende de quanto o Aurora expandiu ou reduziu o cluster antes de pausá-lo.

> **Note**
>
> Se um cluster de banco de dados for pausado por mais de sete dias, o backup do cluster de banco de dados poderá ser feito com um snapshot. Nesse caso, o Aurora restaura o cluster de banco de dados do snapshot quando há uma solicitação de conexão.

Determinar o número máximo de conexões de banco de dados do Aurora Serverless v1

Os exemplos a seguir são de um cluster de banco de dados do Aurora Serverless v1 compatível com o MySQL 5.7. Você pode usar um cliente do MySQL ou o editor de consultas, se tiver configurado o acesso a ele. Para ter mais informações, consulte Executar consultas no editor de consultas.

Como encontrar o número máximo de conexões de banco de dados

1. Encontre o intervalo de capacidade de seu cluster de banco de dados do Aurora Serverless v1 usando a AWS CLI.

```
aws rds describe-db-clusters \
    --db-cluster-identifier my-serverless-57-cluster \
    --query 'DBClusters[*].ScalingConfigurationInfo|[0]'
```

   O resultado mostra que seu intervalo de capacidade é de 1 a 4 ACUs.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

2. Realize a seguinte consulta SQL para encontrar o número máximo de conexões.

```
select @@max_connections;
```

   O resultado mostrado é da capacidade mínima do cluster, 1 ACU.

```
@@max_connections
90
```

3. Dimensione o cluster para 8 a 32 ACUs.

   Para ter mais informações sobre a escalabilidade, consulte Modificar um cluster de banco de dados do Aurora Serverless v1.

4. Confirme o intervalo de capacidade.

```
{
    "MinCapacity": 8,
    "AutoPause": true,
    "MaxCapacity": 32,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
```

```
}
```

5. Encontre o número máximo de conexões.

```
select @@max_connections;
```

O resultado mostrado é da capacidade mínima do cluster, 8 ACUs.

```
@@max_connections
1000
```

6. Dimensione o cluster para o máximo possível, 256 a 256 ACUs.
7. Confirme o intervalo de capacidade.

```
{
    "MinCapacity": 256,
    "AutoPause": true,
    "MaxCapacity": 256,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

8. Encontre o número máximo de conexões.

```
select @@max_connections;
```

O resultado mostrado é de 256 ACUs.

```
@@max_connections
6000
```

> **Note**
>
> O valor `max_connections` não é dimensionado linearmente com o número de ACUs.

9. Reduza a escala do cluster verticalmente para 1 a 4 ACUs.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
```

```
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

Dessa vez, o valor `max_connections` é de 4 ACUs.

```
@@max_connections
270
```

10. Reduza a escala do cluster verticalmente para 2 ACUs.

```
@@max_connections
180
```

Se você configurou o cluster para pausar após determinado período de tempo ocioso, a escala será reduzida verticalmente para 0 ACUs. No entanto, o valor `max_connections` não fica abaixo de 1 ACU.

```
@@max_connections
90
```

Grupos de parâmetros para Aurora Serverless v1

Quando você cria o seu cluster de banco de dados do Aurora Serverless v1, você escolhe um mecanismo de banco de dados do Aurora específico e um grupo de parâmetros de cluster de banco de dados associado. Ao contrário dos clusters de bancos de dados Aurora provisionado, um cluster de banco de dados do Aurora Serverless v1 tem uma única instância de banco de dados de leitura/gravação configurada apenas com um grupo de parâmetros de cluster de banco de dados. Ele não tem um grupo de parâmetros de banco de dados separado. Durante o autoscaling, o Aurora Serverless v1 precisa ser capaz de alterar os parâmetros para que o cluster funcione melhor com a capacidade aumentada ou reduzida. Assim, com um cluster de banco de dados do Aurora Serverless v1, algumas das alterações que você pode fazer nos parâmetros para um determinado tipo de mecanismo de banco de dados podem não se aplicar.

Por exemplo, um cluster de banco de dados do Aurora Serverless v1 baseado no Aurora PostgreSQL não pode usar `apg_plan_mgmt.capture_plan_baselines` e outros parâmetros

para gerenciamento de planos de consulta que podem ser usados em clusters de bancos de dados Aurora PostgreSQL provisionados.

Você pode obter uma lista de valores padrão para os grupos de parâmetros padrão dos vários mecanismos de banco de dados do Aurora usando o comando da CLI describe-engine-default-cluster-parameters e consultando a Região da AWS. A seguir estão os valores que você pode usar para a opção `--db-parameter-group-family`.

Aurora MySQL versão 2	`aurora-mysql5.7`
Aurora PostgreSQL versão 11	`aurora-postgresql11`
Aurora PostgreSQL versão 13	`aurora-postgresql13`

Recomendamos que você configure a AWS CLI com o seu ID de chave de acesso da AWS e a chave de acesso secreta da AWS, além de definir a sua Região da AWS antes de usar os comandos da AWS CLI. Fornecer a região a sua configuração de CLI torna desnecessário inserir o parâmetro `--region` ao executar comandos. Para saber mais sobre como configurar a AWS CLI, consulte Noções básicas de configuração no Guia do usuário da AWS Command Line Interface.

O exemplo a seguir obtém uma lista de parâmetros do grupo de clusters de banco de dados padrão para Aurora MySQL versão 2.

Para Linux, macOS ou Unix:

```
aws rds describe-engine-default-cluster-parameters \
  --db-parameter-group-family aurora-mysql5.7 --query \
  'EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, `serverless`) == `true`] | [*].{param:ParameterName}' \
  --output text
```

Para Windows:

```
aws rds describe-engine-default-cluster-parameters ^
   --db-parameter-group-family aurora-mysql5.7 --query ^
   "EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, 'serverless') == `true`] | [*].{param:ParameterName}" ^
```

```
    --output text
```

Modificação de valores de parâmetro para o Aurora Serverless v1

Como explicado em Grupos de parâmetros para Amazon Aurora, você não pode alterar diretamente os valores em um grupo de parâmetros padrão, independentemente do tipo (grupo de parâmetros de cluster de banco de dados, grupo de parâmetros de banco de dados). Em vez disso, você cria um grupo de parâmetros personalizado com base no grupo de parâmetros do cluster de banco de dados padrão para o seu mecanismo de banco de dados do Aurora e altera as configurações nesse grupo de parâmetros, conforme necessário. Por exemplo, você pode querer alterar algumas das configurações do cluster de banco de dados do Aurora Serverless v1 para consultas de log do ou para fazer upload de logs específicos do mecanismo de banco de dados para o Amazon CloudWatch.

Para criar um grupo de parâmetros de cluster de banco de dados personalizado

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Grupos de parâmetros.
3. Escolha Create parameter group (Criar grupo de parâmetros) para abrir o painel Parameter group details (Detalhes do grupo de parâmetros).
4. Escolha o grupo de cluster de banco de dados padrão apropriado para o mecanismo de banco de dados que você deseja usar para seu cluster de banco de dados do Aurora Serverless v1. Escolha as seguintes opções:

    a. Para Parameter group family (Família de grupo de parâmetros), escolha a família apropriada para o mecanismo de banco de dados escolhido. Sua seleção deve ter o prefixo `aurora-` no nome.

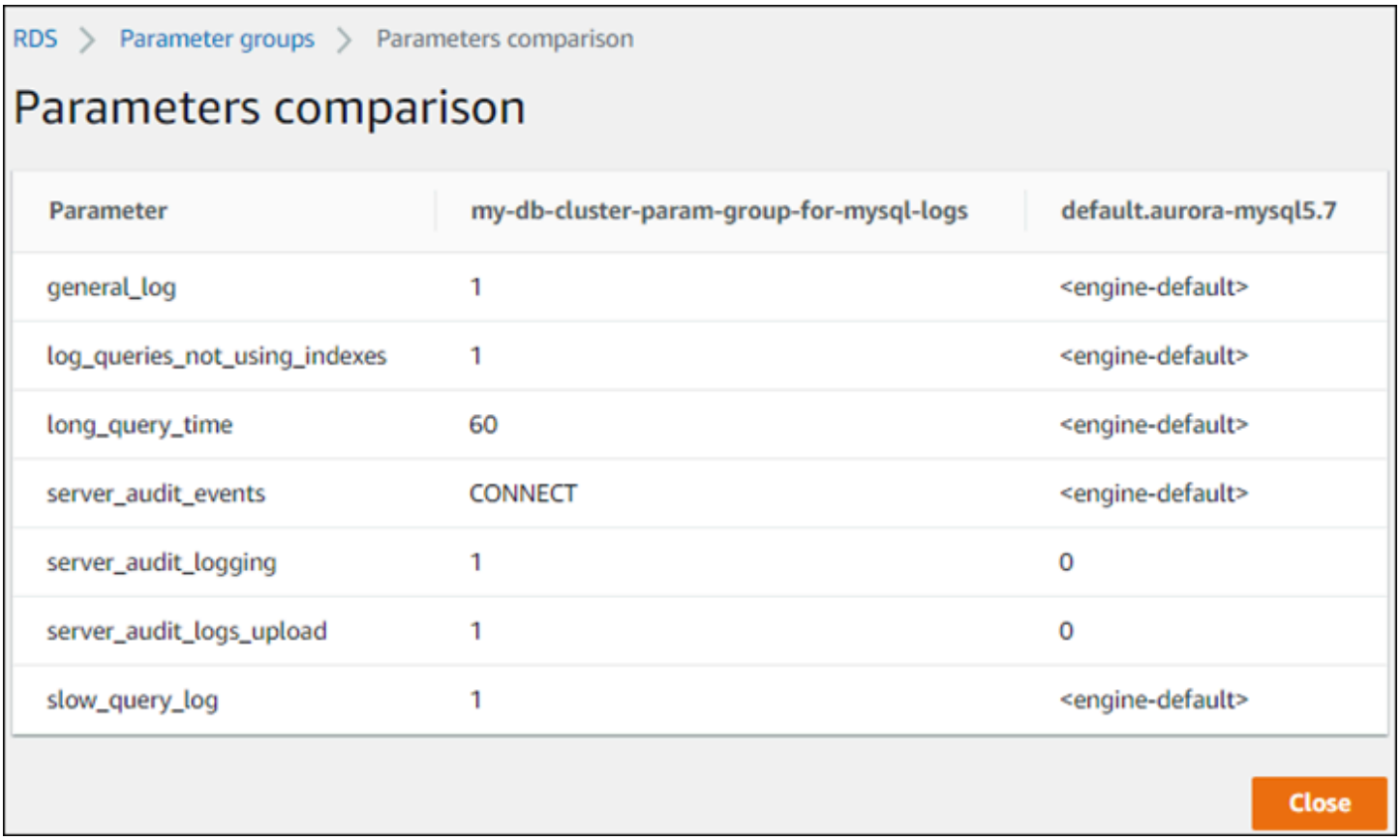
    b. Em Type (Tipo), escolha DB Cluster Parameter Group (Grupo de parâmetros do cluster de banco de dados).

    c. Em Group name (Nome do grupo) e Description (Descrição), insira nomes significativos para você ou outras pessoas que possam precisar trabalhar com seu cluster de banco de dados do Aurora Serverless v1 e respectivos parâmetros.

    d. Escolha Criar.

Seu grupo de parâmetros de cluster de banco de dados personalizado é adicionado à lista de grupos de parâmetros disponíveis em sua Região da AWS. Você pode usar seu grupo de parâmetros de cluster de banco de dados personalizado ao criar clusters de banco de dados do Aurora Serverless v1. Você também pode modificar seus clusters de banco de dados do Aurora Serverless v1 existentes para usar seu grupo de parâmetros de cluster de banco de dados personalizado. Depois que seu cluster de banco de dados do Aurora Serverless v1 começa a usar seu grupo de parâmetros do cluster de banco de dados personalizado, você pode alterar os valores de parâmetros dinâmicos usando o Console de gerenciamento da AWS ou a AWS CLI.

Você também pode usar o console para visualizar uma comparação lado a lado entre os valores no seu grupo de parâmetros do cluster de banco de dados personalizado e no cluster de banco de dados padrão, conforme mostrado no snapshot a seguir.



Quando você altera valores de parâmetros em um cluster de banco de dados ativo, o Aurora Serverless v1 inicia uma escalabilidade perfeita para aplicar as alterações de parâmetros. Se o seu cluster de banco de dados do Aurora Serverless v1 estiver em um estado pausado, ele retomará e começará a escalar para que possa fazer a alteração. A operação de escalabilidade para uma alteração de grupo de parâmetros sempre força a alteração de capacidade, portanto esteja ciente

de que a modificação de parâmetros poderá ocasionar conexões descartadas se um ponto de escalabilidade não puder ser encontrado durante o período de escalabilidade.

Registro em log para o Aurora Serverless v1

Por padrão, logs de erros para o Aurora Serverless v1 são ativados e carregados automaticamente no Amazon CloudWatch. Você também pode fazer com que seu cluster de banco de dados do Aurora Serverless v1 carregue logs específicos do mecanismo de banco de dados do Aurora para o CloudWatch. Para fazer isso, habilite os parâmetros de configuração em seu grupo de parâmetros do seu cluster de banco de dados personalizado. Depois, seu cluster de banco de dados do Aurora Serverless v1 carrega todos os logs disponíveis para o Amazon CloudWatch. Nesse momento, é possível usar o CloudWatch para analisar os dados de log, criar alarmes e visualizar métricas.

No caso do Aurora MySQL, a tabela a seguir mostra os logs que você pode habilitar. Quando habilitados, é feito upload deles automaticamente do cluster de banco de dados do Aurora Serverless v1 para o Amazon CloudWatch.

Log do Aurora MySQL	Descrição
`general_log`	Cria o log geral. Defina como 1 para ativar. O padrão é desativado (0).
`log_queries_not_using_indexes`	Registra todas as consultas no log de consultas lentas que não usam um índice. O padrão é desativado (0). Defina como 1 para ativar esse log.
`long_query_time`	Impede que consultas em execução rápida sejam registradas no log de consultas lentas. Pode ser definido como um float entre 0 e 3.1536.000. O padrão é 0 (não ativo).
`server_audit_events`	A lista de eventos a serem capturados nos logs. Os valores compatíveis são `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Defina como 1 para ativar o log de auditoria do servidor. Se você ativar isso, você

Log do Aurora MySQL	Descrição
	poderá especificar os eventos de auditoria a serem enviados, CloudWatch listando-os no `server_audit_events` parâmetro.
`slow_query_log`	Cria um log de consulta lento. Defina como 1 para ativar o log de consulta lenta. O padrão é desativado (0).

Para obter mais informações, consulte Como utilizar a auditoria avançada em um cluster de banco de dados do Amazon Aurora MySQL.

No caso do Aurora PostgreSQL, a tabela a seguir mostra os logs que você pode habilitar. Quando habilitados, é feito upload deles automaticamente do cluster de banco de dados do Aurora Serverless v1 para o Amazon CloudWatch com os logs de erro regulares.

Log do Aurora PostgreSQL	Descrição
`log_connections`	Ativado por padrão e não pode ser alterado. Ele registra detalhes para todas as novas conexões de clientes.
`log_disconnections`	Ativado por padrão e não pode ser alterado. Registra todas as desconexões de clientes.
`log_hostname`	Desativado por padrão e não pode ser alterado. Nomes de host não são registrados no log.
`log_lock_waits`	O padrão é 0 (desativado). Defina como 1 para registrar esperas por bloqueio.
`log_min_duration_statement`	A duração mínima (em milissegundos) para que uma instrução seja executada antes de ser registrada.
`log_min_messages`	Define os níveis de mensagem registrados. Os valores compatíveis são `debug5`, `debug4`,

Log do Aurora PostgreSQL	Descrição
	`debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Para registrar dados de performance no log do `postgres`, defina o valor como `debug1`.
`log_temp_files`	Registra o uso de arquivos temporários que estão acima dos kilobytes (kB) especificados.
`log_statement`	Controla as instruções SQL específicas que são registradas. Os valores compatíveis são `none`, `ddl`, `mod` e `all`. O padrão é `none`.

Depois de ativar os logs para Aurora MySQL ou Aurora PostgreSQL para o cluster de banco de dados do Aurora Serverless v1, você pode visualizá-los no CloudWatch.

Visualizar logs do Aurora Serverless v1 com o Amazon CloudWatch

Aurora Serverless v1O carrega automaticamente (“publica”) no Amazon CloudWatch todos os logs que estão ativados em seu grupo de parâmetros de cluster de banco de dados personalizado. Você não precisa escolher ou especificar os tipos de log. O upload de logs começa assim que você habilita o parâmetro de configuração de log. Se, mais tarde, você desabilitar o parâmetro de log, os uploads serão interrompidos. No entanto, todos os logs que já foram publicados no CloudWatch permanecem até que você os exclua.

Para mais informações sobre como usar o CloudWatch com os logs do Aurora MySQL, consulte Monitorar eventos de log no Amazon CloudWatch.

Para ter mais informações sobre CloudWatch e Aurora PostgreSQL, consulte Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs.

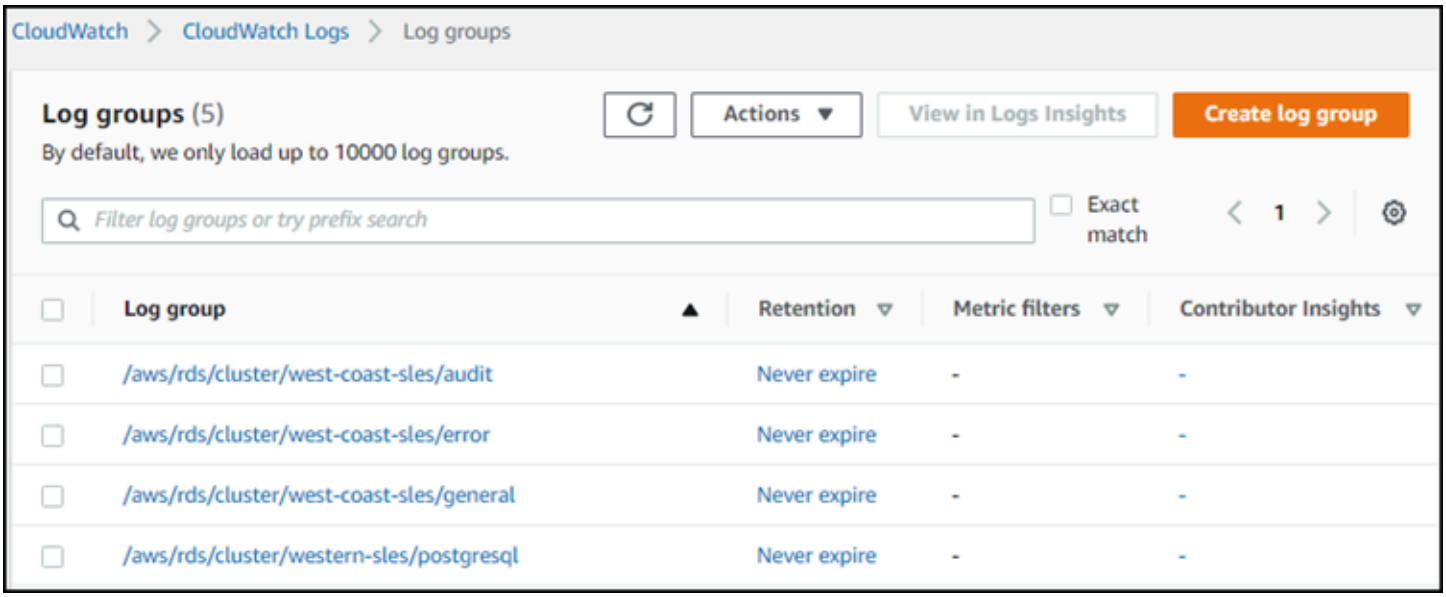
Para visualizar logs do cluster de banco de dados do Aurora Serverless v1

1. Abra o console do CloudWatch, em https://console.aws.amazon.com/cloudwatch/.
2. Escolha a Região da AWS.
3. Escolha Grupos de logs.

4. Escolha o log do cluster de banco de dados do Aurora Serverless v1 na lista. Para logs de erros, o padrão de nomenclatura é o seguinte.

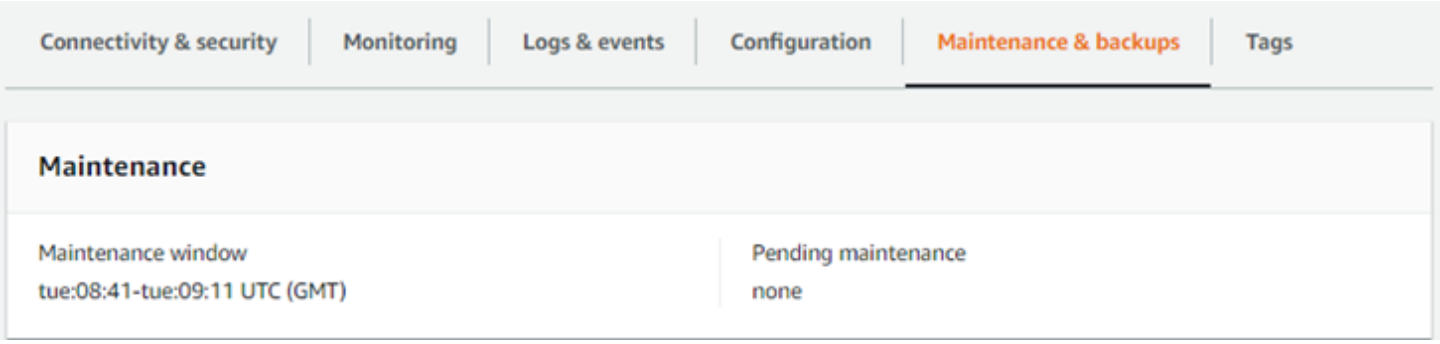
```
/aws/rds/cluster/cluster-name/error
```

Por exemplo, no snapshot a seguir, você pode encontrar listagens para logs publicados para um cluster de banco de dados do Aurora Serverless v1 para Aurora PostgreSQL denominado `western-sles`. Você também pode encontrar várias listas para cluster de banco de dados do Aurora Serverless v1 para Aurora MySQL, `west-coast-sles`. Escolha o log de interesse para começar a explorar seu conteúdo.



Aurora Serverless v1 e manutenção

A manutenção do cluster de banco de dados do Aurora Serverless v1, como a aplicação dos recursos, das correções e das atualizações de segurança mais recentes, é realizada automaticamente para você. O Aurora Serverless v1 tem uma janela de manutenção que você pode visualizar no Console de gerenciamento da AWS em Manutenção e backups para seu cluster de banco de dados do Aurora Serverless v1. É possível encontrar a data e a hora em que a manutenção pode ser realizada e se alguma manutenção está pendente para o cluster de banco de dados do Aurora Serverless v1, conforme mostrado na figura a seguir.



Você pode definir a janela de manutenção ao criar o cluster de banco de dados do Aurora Serverless v1 e pode modificá-la posteriormente. Para ter mais informações, consulte Ajustar a janela de manutenção do cluster de banco de dados preferencial.

As janelas de manutenção são usadas para atualizações programadas de versões principais. Atualizações de versões secundárias e patches são aplicados imediatamente durante o ajuste de escala. O ajuste de escala ocorre de acordo com a configuração de `TimeoutAction`:

- `ForceApplyCapacityChange`: a alteração é aplicada imediatamente.
- `RollbackCapacityChange`: o Aurora atualizará o cluster à força após três dias a partir da primeira tentativa.

Tal como acontece com qualquer alteração que seja forçada sem um ponto de escala apropriado, isso pode interromper sua workload.

Sempre que possível, o Aurora Serverless v1 realiza a manutenção sem causar disrupções. Quando a manutenção é necessária, o cluster de banco de dados do Aurora Serverless v1 dimensiona sua capacidade para lidar com as operações necessárias. Antes de escalar, o Aurora Serverless v1 procura um ponto de escalabilidade. Se necessário, isso será realizado por até três dias.

No fim de cada dia em que o Aurora Serverless v1 não consegue encontrar um ponto de escalabilidade, ele cria um evento de cluster. Esse evento notifica você sobre a manutenção pendente e a necessidade de dimensionar para executar a manutenção. A notificação inclui a data em que o Aurora Serverless v1 pode forçar o cluster de banco de dados a escalar.

Para obter mais informações, consulte Ação de tempo limite para alterações na capacidade.

Aurora Serverless v1 e failover

Se a instância de banco de dados de um cluster de banco de dados do Aurora Serverless v1 ficar indisponível ou a zona de disponibilidade (AZ) em que ela está estiver com falha, o Aurora recriará

a instância de banco de dados em uma AZ diferente. No entanto, o cluster do Aurora Serverless v1 não é um cluster multi-AZ. O motivo disso é que ele consiste em uma única instância de banco de dados em uma única AZ. Dessa forma, esse mecanismo de failover leva mais tempo do que um cluster do Aurora com instâncias do Aurora Serverless v2 ou provisionadas. O tempo de failover do Aurora Serverless v1 é indefinido no momento, porque depende da demanda e da disponibilidade da capacidade em outras AZs na Região da AWS indicada.

Como o Aurora separa a capacidade computacional e o armazenamento, o volume de armazenamento do cluster é distribuído em várias AZs. Seus dados permanecem disponíveis mesmo que as interrupções afetem a instância de banco de dados ou a AZ associada.

Aurora Serverless v1 e snapshots

O volume do cluster de um cluster do Aurora Serverless v1 é sempre criptografado. É possível escolher a chave de criptografia, mas não é possível desabilitar a criptografia. Para copiar ou compartilhar um snapshot de um cluster do Aurora Serverless v1, criptografe o snapshot usando a sua própria AWS KMS key. Para ter mais informações, consulte Cópia de snapshot de cluster de banco de dados. Para saber mais sobre criptografia e o Amazon Aurora, consulte Criptografar um cluster de banco de dados do Amazon Aurora

Criar um cluster de banco de dados do Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Exporte estendido do Amazon RDS. Para ter mais informações, consulte Suporte estendido do RDS.

O procedimento a seguir cria um cluster do Aurora Serverless v1 sem nenhum dos objetos nem dados do esquema. Se você deseja criar um cluster do Aurora Serverless v1 seja uma duplicata de um cluster provisionado existente ou do Aurora Serverless v1, realize uma operação de restauração ou clonagem de snapshots. Para obter esses detalhes, consulte Restauração de um snapshot de um cluster de banco de dados e Clonar um volume para um cluster de banco de dados do Amazon

Aurora. Não é possível converter um cluster provisionado existente em Aurora Serverless v1. Também não é possível converter um cluster do Aurora Serverless v1 em um cluster provisionado.

Ao criar um cluster de banco de dados do Aurora Serverless v1, configure as capacidades mínima e máxima para o cluster. Uma unidade de capacidade é equivalente a uma configuração específica de computação e memória. O Aurora Serverless v1 cria regras de escalabilidade de limites para utilização de CPU, conexões e memória disponível e escala perfeitamente para uma série de unidades de capacidade, conforme o necessário às suas aplicações. Para ter mais informações, consulte Aurora Serverless v1Arquitetura do.

É possível definir os seguintes valores específicos para seu cluster de banco de dados do Aurora Serverless v1:

- Minimum Aurora capacity unit (Unidade de capacidade mínima do Aurora): o Aurora Serverless v1 pode reduzir a capacidade até essa unidade de capacidade.
- Maximum Aurora capacity unit (Unidade de capacidade máxima do Aurora): o Aurora Serverless v1 pode aumentar a capacidade até essa unidade de capacidade.

Também é possível escolher as seguintes opções opcionais de configuração de escalabilidade:

- Force scaling the capacity to the specified values when the timeout is reached (Forçar escalabilidade da capacidade para os valores especificados quando o tempo limite for atingido) é possível escolher essa configuração se quiser que o Aurora Serverless v1 force o Aurora Serverless v1 a escalar, mesmo que não consiga encontrar um ponto de escalabilidade antes do tempo limite. Se você quiser que o Aurora Serverless v1 cancele as alterações de capacidade se não conseguir encontrar um ponto de escalabilidade, não escolha essa definição. Para ter mais informações, consulte Ação de tempo limite para alterações na capacidade.
- Pause compute capacity after consecutive minutes of inactivity (Pausar a capacidade computacional após minutos consecutivos de inatividade): é possível escolher essa configuração se você quiser que o Aurora Serverless v1 escale para zero quando não houver atividade no cluster de banco de dados por um período especificado. Com essa configuração ativada, o cluster de banco de dados do Aurora Serverless v1 retoma o processamento automaticamente e escala para a capacidade necessária para lidar com a workload quando o tráfego do banco de dados for retomado. Para saber mais, consulte Pausa e retomada para o Aurora Serverless v1.

Antes de criar um cluster de banco de dados do Aurora Serverless v1, é necessária uma conta da AWS. Também é necessário concluir as tarefas de configuração para trabalhar com o Amazon

Aurora. Para ter mais informações, consulte Configuração de seu ambiente para Amazon Aurora. Também é necessário concluir outras etapas preliminares para criar qualquer cluster de bancos de dados Aurora. Para saber mais, consulte Criar um cluster de bancos de dados do Amazon Aurora.

O Aurora Serverless v1 está disponível em determinadas Regiões da AWS e somente para versões específicas do Aurora MySQL e do Aurora PostgreSQL. Para ter mais informações, consulte Aurora Serverless v1 da.

> **Note**
>
> O volume do cluster de um cluster do Aurora Serverless v1 é sempre criptografado. Ao criar seu cluster de banco de dados do Aurora Serverless v1, não é possível desativar a criptografia, mas pode optar por usar sua própria chave de criptografia. Com o Aurora Serverless v2, você pode escolher se deseja criptografar o volume do cluster.

É possível criar um novo cluster de banco de dados do Aurora Serverless v1 com a AWS CLI ou a API do RDS.

> **Note**
>
> Se você receber a seguinte mensagem de erro ao tentar criar seu cluster, sua conta precisará de permissões adicionais.
> `Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.`
> Consulte Usar funções vinculadas ao serviço do Amazon Aurora para obter mais informações.

Você não pode se conectar diretamente à instância de banco de dados em seu cluster de banco de dados do Aurora Serverless v1. Para se conectar a um cluster de banco de dados do Aurora Serverless v1, use o endpoint do banco de dados. Você pode encontrar o endpoint do seu cluster de banco de dados do Aurora Serverless v1 na guia Connectivity & security (Conectividade e segurança) do cluster no Console de gerenciamento da AWS. Para obter mais informações, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

AWS CLI

Para criar um novo cluster de banco de dados do Aurora Serverless v1 com a AWS CLI, execute o comando create-db-cluster e especifique `serverless` na opção `--engine-mode`.

Opcionalmente, você pode especificar a opção `--scaling-configuration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões.

Os exemplos de comando a seguir criam um novo cluster de banco de dados Serverless configurando a opção `--engine-mode` como `serverless`. O exemplo também especifica valores para a opção `--scaling-configuration`.

Exemplo de Aurora MySQL

O comando a seguir cria um cluster de banco de dados sem servidor compatível com o Aurora MySQL. Os valores de capacidade válidos para Aurora MySQL são 1, 2, 4, 8, 16, 32, 64, 128 e 256.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

Exemplo de Aurora PostgreSQL

O comando a seguir cria um cluster de banco de dados sem servidor compatível com o PostgreSQL 13.9. Os valores de capacidade válidos para o Aurora PostgreSQL são 2, 4, 8, 16, 32, 64, 192 e 384.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-postgresql --engine-version 13.9 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-postgresql --engine-version 13.9 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

API do RDS

Para criar um cluster de banco de dados do Aurora Serverless v1 com a API do RDS, execute a operação CreateDBCluster e especifique `serverless` para o parâmetro `EngineMode`.

Opcionalmente, você pode especificar o parâmetro `ScalingConfiguration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Restaurar um cluster de banco de dados do Aurora Serverless v1

Important

A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente.

Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Exporte estendido do Amazon RDS. Para ter mais informações, consulte Suporte estendido do RDS.

É possível configurar um cluster de banco de dados do Aurora Serverless v1 ao restaurar um snapshot de cluster de banco de dados provisionado com a AWS CLI ou a API do RDS.

Ao restaurar um snapshot em um cluster de banco de dados do Aurora Serverless v1, é possível definir os seguintes valores específicos:

- Minimum Aurora capacity unit (Unidade de capacidade mínima do Aurora): o Aurora Serverless v1 pode reduzir a capacidade até essa unidade de capacidade.
- Maximum Aurora capacity unit (Unidade de capacidade máxima do Aurora): o Aurora Serverless v1 pode aumentar a capacidade até essa unidade de capacidade.
- Timeout action (Ação de tempo limite): a ação a ser executada quando uma modificação de capacidade expira porque não consegue encontrar um ponto de escalabilidade. Aurora Serverless v1 O cluster de banco de dados poderá forçar o cluster de banco de dados para as novas configurações de capacidade se definir a opção Force scaling the capacity to the specified values... (Forçar escalabilidade da capacidade para os valores especificados...). Ou ele poderá reverter a alteração de capacidade para cancelá-la se você não escolher a opção. Para obter mais informações, consulte Ação de tempo limite para alterações na capacidade.
- Pause after inactivity (Pausar depois de inatividade) – a quantidade de tempo sem tráfego no banco de dados que determina o redimensionamento da capacidade de processamento para zero. Quando o tráfego no banco de dados é retomado, o Aurora retoma automaticamente a capacidade de processamento e escala para tratar o tráfego.

Para obter mais informações gerais sobre como restaurar um cluster de banco de dados a partir de um snapshot, consulte Restauração de um snapshot de um cluster de banco de dados.

AWS CLI

É possível configurar um cluster de banco de dados do Aurora Serverless ao restaurar um snapshot de cluster de banco de dados provisionado com o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Ao restaurar um snapshot em um cluster de banco de dados do Aurora Serverless, é possível definir os seguintes valores específicos:

- Minimum Aurora capacity unit (Unidade de capacidade mínima do Aurora): o Aurora Serverless pode reduzir a capacidade até essa unidade de capacidade.
- Maximum Aurora capacity unit (Unidade de capacidade máxima do Aurora): o Aurora Serverless pode aumentar a capacidade até essa unidade de capacidade.
- Timeout action (Ação de tempo limite): a ação a ser executada quando uma modificação de capacidade expira porque não consegue encontrar um ponto de escalabilidade. Aurora Serverless v1 O cluster de banco de dados poderá forçar o cluster de banco de dados para as novas configurações de capacidade se definir a opção Force scaling the capacity to the specified values... (Forçar escalabilidade da capacidade para os valores especificados...). Ou ele poderá reverter a alteração de capacidade para cancelá-la se você não escolher a opção. Para obter mais informações, consulte Ação de tempo limite para alterações na capacidade.
- Pause after inactivity (Pausar depois de inatividade) – a quantidade de tempo sem tráfego no banco de dados que determina o redimensionamento da capacidade de processamento para zero. Quando o tráfego no banco de dados é retomado, o Aurora retoma automaticamente a capacidade de processamento e escala para tratar o tráfego.

> **Note**
>
> A versão do snapshot do cluster de banco de dados deve ser compatível com o Aurora Serverless v1. Para obter a lista de versões compatíveis, consulte Aurora Serverless v1 da.

Para restaurar um snapshot para um cluster do Aurora Serverless v1 com compatibilidade com o MySQL 5.7, inclua os seguintes parâmetros adicionais:

- `--engine` **`aurora-mysql`**
- `--engine-version` **`5.7`**

Os parâmetros `--engine` e `--engine-version` permitem que você crie um cluster do Aurora Serverless v1 compatível com MySQL 5.7 a partir de um snapshot do Aurora compatível com o MySQL 5.6 ou do Aurora Serverless v1 . O exemplo a seguir restaura um snapshot de um cluster compatível com MySQL 5.6 chamado *`mydbclustersnapshot`* para um cluster Aurora Serverless v1 compatível com MySQL 5.7 chamado *`mynewdbcluster`*.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless \
    --engine aurora-mysql \
    --engine-version 5.7
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql ^
    --engine-version 5.7
```

Opcionalmente, você pode especificar a opção `--scaling-configuration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

No exemplo a seguir, você restaura de um snapshot de cluster de banco de dados criado anteriormente chamado *mydbclustersnapshot* para um novo cluster de banco de dados chamado *mynewdbcluster*. Você define o `--scaling-configuration` para que o novo cluster de banco de dados do Aurora Serverless v1 possa escalar de 8 ACUs para 64 ACUs (unidades de capacidade do Aurora) conforme necessário para processar a workload. Após a conclusão do processamento e após 1000 segundos sem nenhuma conexão compatível, o cluster é encerrado até que as solicitações de conexão peçam para reiniciar.

Para Linux, macOS ou Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

API do RDS

Para configurar um cluster de banco de dados do Aurora Serverless v1 ao restaurar um cluster de banco de dados usando a API do RDS, execute a operação RestoreDBClusterFromSnapshot e especifique `serverless` no parâmetro `EngineMode`.

Opcionalmente, você pode especificar o parâmetro `ScalingConfiguration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Modificar um cluster de banco de dados do Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

Depois de configurar um cluster de banco de dados do Aurora Serverless v1, é possível modificar determinadas propriedades com o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. A maioria das propriedades que você pode modificar é a mesma de outros tipos de clusters do Aurora.

As alterações mais relevantes do Aurora Serverless v1 são as seguintes:

- Modificar a configuração de ajuste de escala
- Atualizar a versão principal
- Conversão de Aurora Serverless v1 em provisionado

Modificar a configuração de ajuste de escala de um cluster de banco de dados do Aurora Serverless v1

As capacidades mínima e máxima podem ser definidas para o cluster de banco de dados. Cada unidade de capacidade é equivalente a uma configuração específica de computação e de memória. O Aurora Serverless cria automaticamente regras de escalabilidade de limites para utilização de CPU, conexões e memória disponível. Também é possível definir se o Aurora Serverless pausa o banco de dados quando não há nenhuma atividade e se retoma quando a atividade começa novamente.

É possível definir os seguintes valores específicos para a configuração de escalabilidade:

- Minimum Aurora capacity unit (Unidade de capacidade mínima do Aurora): o Aurora Serverless pode reduzir a capacidade até essa unidade de capacidade.
- Maximum Aurora capacity unit (Unidade de capacidade máxima do Aurora): o Aurora Serverless pode aumentar a capacidade até essa unidade de capacidade.
- Autoscaling timeout and action (Tempo limite e ação de autoescalabilidade): esta seção especifica quanto tempo o Aurora Serverless aguarda até encontrar um ponto de escalabilidade antes de expirar. Também especifica a ação a ser executada quando uma alteração de capacidade expira, pois não consegue encontrar um ponto de escalabilidade. O Aurora pode forçar a alteração na capacidade para definir a capacidade para o valor especificado assim que possível. Ou, pode reverter a alteração na capacidade para cancelá-la. Para obter mais informações, consulte Ação de tempo limite para alterações na capacidade.
- Pausa após a inatividade: use a configuração opcional Ajustar a escala da capacidade para 0 ACU quando o cluster estiver ocioso para ajustar a escala do banco de dados para zero capacidade de processamento enquanto ele estiver inativo. Quando o tráfego no banco de dados é retomado, o Aurora retoma automaticamente a capacidade de processamento e escala para tratar o tráfego.

> **Note**
>
> Quando você modifica o intervalo de capacidade de um cluster de banco de dados do Aurora Serverless, a alteração ocorre imediatamente, independentemente de você optar por aplicá-la imediatamente ou durante a próxima janela de manutenção programada.

Console

Você pode modificar a configuração de escalabilidade de um cluster de bancos de dados Aurora com o Console de gerenciamento da AWS.

Como modificar um cluster de banco de dados do Aurora Serverless v1

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados do Aurora Serverless v1 que você deseja modificar.
4. Em Actions (Ações), escolha Modify cluster (Modificar cluster).
5. Na seção Capacity settings (Configurações da capacidade), modifique a configuração de escalabilidade.
6. Escolha Continuar.
7. Na página Modificar cluster de banco de dados, revise suas modificações e escolha quando aplicá-las.
8. Escolha Modificar Cluster.

AWS CLI

Para modificar a configuração de escalabilidade do cluster de banco de dados do Aurora Serverless v1 usando a AWS CLI, execute o comando modify-db-cluster da AWS CLI. Especifique a opção `--scaling-configuration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Neste exemplo, modifique a configuração de escalabilidade de um cluster de banco de dados do Aurora Serverless v1 chamado *sample-cluster*.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

API do RDS

É possível modificar a configuração de escalabilidade de um cluster de bancos de dados Aurora com a operação da API ModifyDBCluster. Especifique o parâmetro `ScalingConfiguration` para configurar a capacidade mínima, a capacidade máxima e a pausa automática quando não houver conexões. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Atualizar a versão principal de um cluster de banco de dados do Aurora Serverless v1

Important

A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte

estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

Você pode fazer upgrade da versão principal de um cluster de banco de dados do Aurora Serverless v1 compatível com o PostgreSQL 11 para uma versão correspondente compatível com o PostgreSQL 13.

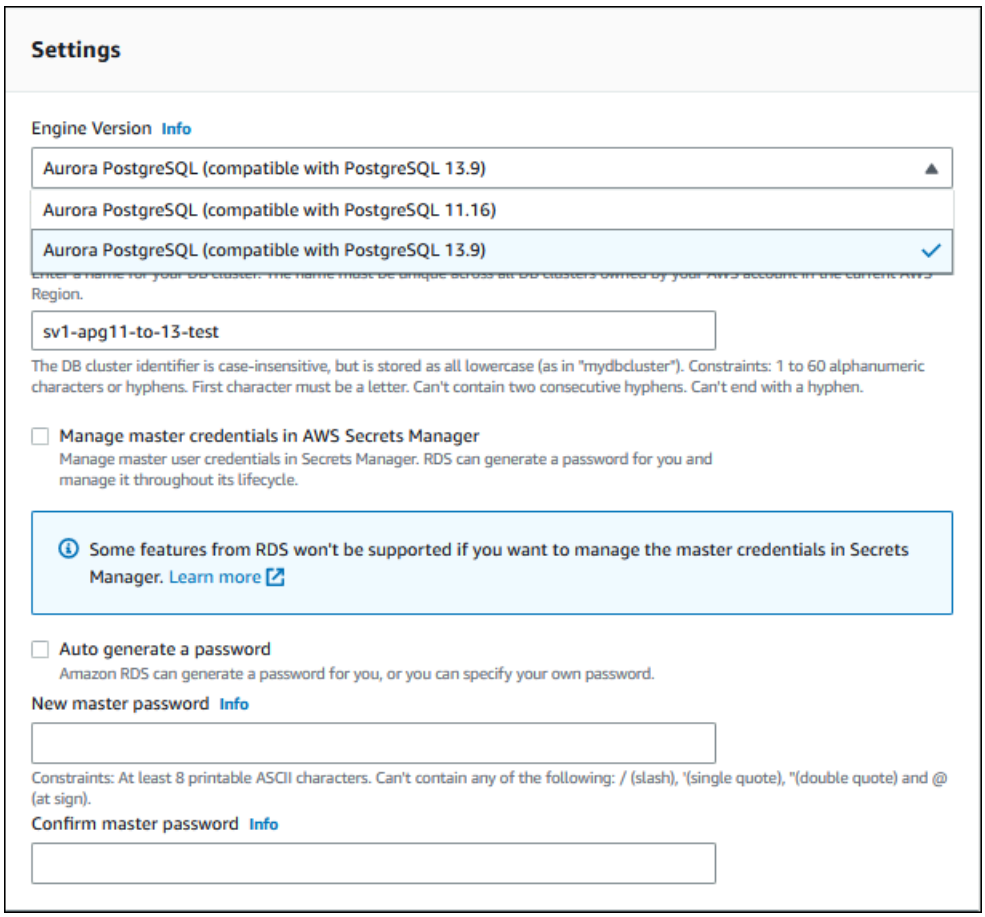
Console

Você pode realizar uma atualização no local de um cluster de banco de dados do Aurora Serverless v1 usando o Console de gerenciamento da AWS.

Como atualizar um cluster de banco de dados do Aurora Serverless v1

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione o cluster de banco de dados do Aurora Serverless v1 que você deseja atualizar.
4. Em Actions (Ações), escolha Modify cluster (Modificar cluster).
5. Em Versão, escolha uma versão 13 do Aurora PostgreSQL.

   O exemplo a seguir mostra uma atualização no local do Aurora PostgreSQL 11.16 para 13.9.



Se você realizar uma atualização de versão principal, deixe todas as outras propriedades iguais. Para alterar qualquer uma das outras propriedades, realize outra operação Modificar após o término da atualização.

6. Escolha Continuar.
7. Na página Modificar cluster de banco de dados, revise suas modificações e escolha quando aplicá-las.
8. Escolha Modificar Cluster.

AWS CLI

Para realizar uma atualização no local de um cluster de banco de dados do Aurora Serverless v1 compatível com o PostgreSQL 11 para um compatível com o PostgreSQL 13, especifique o parâmetro `--engine-version` com um número de versão do Aurora PostgreSQL versão 13 compatível com o Aurora Serverless v1. Inclua também o parâmetro `--allow-major-version-upgrade`.

Neste exemplo, modifique a versão principal de um cluster de banco de dados do Aurora Serverless v1 compatível com o PostgreSQL 11 denominado `sample-cluster`. Com esse procedimento, é realizado uma atualização no local para um cluster de banco de dados do Aurora Serverless v1 compatível com o PostgreSQL 13.

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine-version 13.serverless_12 \
    --allow-major-version-upgrade
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine-version 13.serverless_12 ^
    --allow-major-version-upgrade
```

API do RDS

Para realizar uma atualização no local de um cluster de banco de dados do Aurora Serverless v1 compatível com o PostgreSQL 11 para um compatível com o PostgreSQL 13, especifique o parâmetro `EngineVersion` com um número de versão do Aurora PostgreSQL versão 13 compatível com o Aurora Serverless v1. Inclua também o parâmetro `AllowMajorVersionUpgrade`.

Realizar a conversão de um cluster de banco de dados do Aurora Serverless v1 provisionado

É possível converter um cluster de banco de dados do Aurora Serverless v1 em um cluster de banco de dados provisionado. Para realizar a conversão, use a AWS CLI ou a API do Amazon RES para alterar a classe da instância de banco de dados para Provisionada. Use as etapas abaixo para modificar a classe de instância de banco de dados.

O exemplo a seguir demonstra como usar a AWS CLI para converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado.

AWS CLI

Para converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado, execute o comando modify-db-cluster da AWS CLI.

Os seguintes parâmetros são obrigatórios:

- `--db-cluster-identifier`: o cluster de banco de dados do Aurora Serverless v1 que você está convertendo em provisionado.
- `--engine-mode`: use o valor `provisioned`.
- `--allow-engine-mode-change`
- `--db-cluster-instance-class`: escolha a classe da instância de banco de dados para o cluster de banco de dados provisionado com base na capacidade do cluster de banco de dados do Aurora Serverless v1.

Neste exemplo, você converte um cluster de banco de dados do Aurora Serverless v1 chamado `sample-cluster` e usa a classe de instância de banco de dados `db.r5.xlarge`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier sample-cluster \
--engine-mode provisioned \
--allow-engine-mode-change \
--db-cluster-instance-class db.r5.xlarge
```

Para Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--engine-mode provisioned ^
--allow-engine-mode-change ^
--db-cluster-instance-class db.r5.xlarge
```

O exemplo a seguir demonstra como usar a API do Amazon RDS para converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado.

API do RDS

Para converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado, use a operação ModifyDBCluster da API.

Os seguintes parâmetros são obrigatórios:

- `DBClusterIdentifier`: o cluster de banco de dados do Aurora Serverless v1 que você está convertendo em provisionado.

- `EngineMode`: use o valor `provisioned`.
- `AllowEngineModeChange`
- `DBClusterInstanceClass`: escolha a classe da instância de banco de dados para o cluster de banco de dados provisionado com base na capacidade do cluster de banco de dados do Aurora Serverless v1.

Considerações ao converter um cluster de banco de dados do Aurora Serverless v1 em um cluster provisionado

As seguintes considerações se aplicam quando um cluster de banco de dados do Aurora Serverless v1 é convertido em um cluster provisionado:

- Você pode usar essa conversão como parte da atualização do cluster de banco de dados de Aurora Serverless v1 para Aurora Serverless v2. Para ter mais informações, consulte Atualizar a partir de um cluster do Aurora Serverless v1 para o Aurora Serverless v2.
- O processo de conversão cria uma instância de banco de dados do leitor no cluster de banco de dados, promove a instância do leitor a uma instância de gravador e, depois, exclui a instância original do Aurora Serverless v1. Ao converter o cluster de banco de dados, você não pode realizar nenhuma outra modificação ao mesmo tempo, como alterar a versão do mecanismo de banco de dados ou o grupo de parâmetros do cluster de banco de dados. A operação de conversão é aplicada imediatamente e não pode ser desfeita.
- Durante a conversão, um snapshot do cluster de banco de dados de backup é obtido do cluster de banco de dados caso ocorra um erro. O identificador do snapshot de cluster de banco de dados tem o formato `pre-modify-engine-mode-`*`DB_cluster_identifier`*`-`*`timestamp`*.
- O Aurora usa a versão secundária atual padrão do mecanismo de banco de dados para o cluster de banco de dados provisionado.
- Se você não fornecer uma classe de instância de banco de dados para o cluster de banco de dados convertido, o Aurora recomendará uma com base na capacidade máxima do cluster de banco de dados original do Aurora Serverless v1. A capacidade recomendada para os mapeamentos de classe de instância é mostrada na tabela a seguir.

Capacidade máxima do Serverless (ACUs)	Classe de instância de banco de dados provisionada
1	db.t3.small

Capacidade máxima do Serverless (ACUs)	Classe de instância de banco de dados provisionada
2	db.t3.medium
4	db.t3.large
8	db.r5.large
16	db.r5.xlarge
32	db.r5.2xlarge
64	db.r5.4xlarge
128	db.r5.8xlarge
192	db.r5.12xlarge
256	db.r5.16xlarge
384	db.r5.24xlarge

Note

Dependendo da classe de instância de banco de dados escolhida e do uso do banco de dados, você pode ver custos diferentes para um cluster de banco de dados provisionado em comparação com o Aurora Serverless v1.

Se você converter o cluster de banco de dados do Aurora Serverless v1 em uma classe de instância de banco de dados expansível (db.t*), poderá gerar custos adicionais pelo uso do cluster de banco de dados. Para obter mais informações, consulte Tipos de classe de instância de banco de dados.

Dimensionar manualmente a capacidade do cluster de banco de dados do Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Exporte estendido do Amazon RDS. Para ter mais informações, consulte Suporte estendido do RDS.

Normalmente, os clusters de banco de dados do Aurora Serverless v1 são dimensionados perfeitamente com base na workload. No entanto, a capacidade nem sempre pode escalar rapidamente o suficiente para atender a extremos súbitos, como um aumento exponencial nas transações. Nesses casos, você pode iniciar a operação de escalabilidade manualmente definindo um novo valor de capacidade. Depois de definir a capacidade explicitamente, o Aurora Serverless v1 escalará o cluster de banco de dados automaticamente. Ele faz isso baseado no período de resfriamento para redução.

Você pode definir explicitamente a capacidade de um cluster de banco de dados do Aurora Serverless v1 como um valor específico com o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Você pode definir a capacidade de um cluster de bancos de dados Aurora com o Console de gerenciamento da AWS.

Como modificar um cluster de banco de dados do Aurora Serverless v1

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados do Aurora Serverless v1 que você deseja modificar.
4. Em Actions (Ações), escolha Set capacity (Definir capacidade).

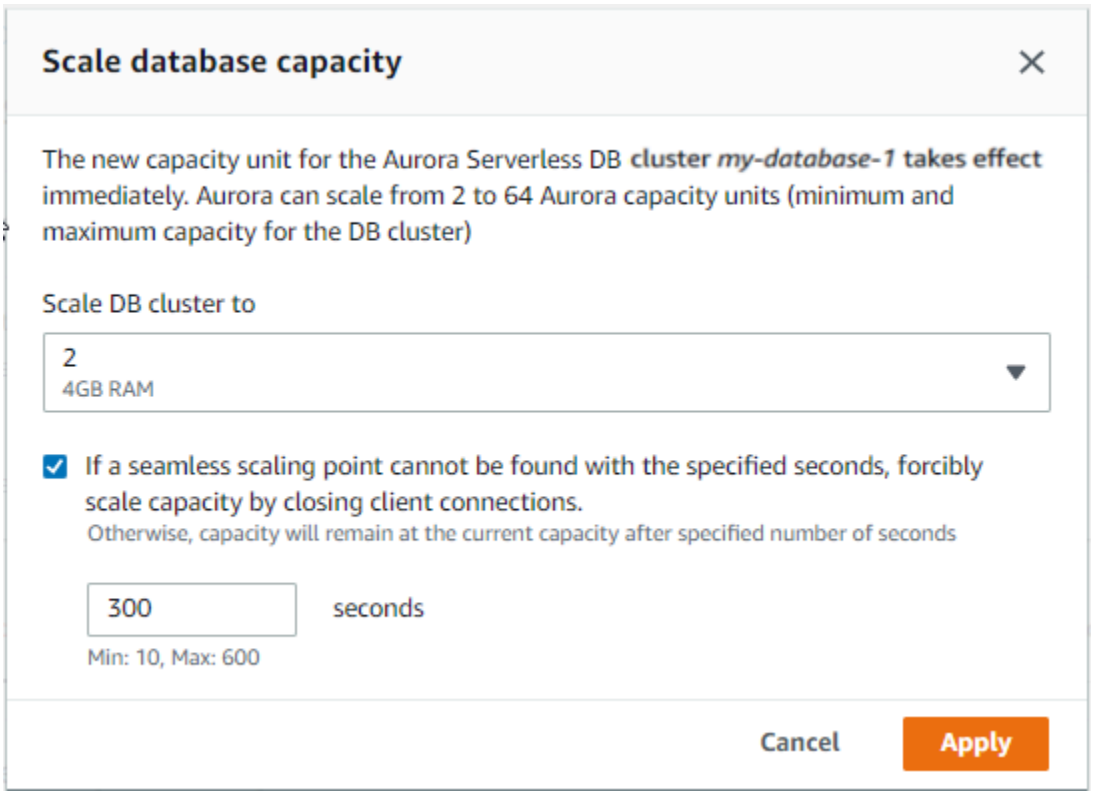
5. Na janela Scale database capacity (Dimensionar capacidade do banco de dados), escolha o seguinte:

   a. Para o seletor suspenso Scale DB cluster to (Dimensionar cluster de banco de dados para), escolha a nova capacidade desejada para o cluster de banco de dados.

   b. Na caixa de seleção If a seamless scaling point cannot be found (Se não for possível encontrar um ponto de escalabilidade perfeito), escolha o comportamento desejado para a configuração `TimeoutAction` do cluster de banco de dados do Aurora Serverless v1 da seguinte forma:

      - Desmarque essa opção se quiser que a capacidade permaneça inalterada se o Aurora Serverless v1 não conseguir encontrar um ponto de escalabilidade antes do tempo limite.
      - Marque essa opção se quiser forçar o cluster de banco de dados do Aurora Serverless v1 a alterar sua capacidade, mesmo que ele não encontre um ponto de escalabilidade antes do tempo limite. Essa opção pode resultar na liberação de conexões pelo Aurora Serverless v1 que impedem que ele encontre um ponto de escala.

   c. Em seconds (segundos), insira a quantidade de tempo que você deseja permitir que seu cluster de banco de dados do Aurora Serverless v1 procure um ponto de escalabilidade antes do tempo limite. Você pode especificar em qualquer lugar de 10 segundos a 600 segundos (10 minutos). O padrão é cinco minutos (300 segundos). Este exemplo a seguir força o cluster de banco de dados do Aurora Serverless v1 banco de dados a escalar até 2 ACUs, mesmo que não encontre um ponto de escala dentro de cinco minutos.



6. Escolha Apply (Aplicar).

Para saber mais sobre pontos de escalabilidade, `TimeoutAction` e períodos de resfriamento, consulte Autoscaling for Aurora Serverless v1.

AWS CLI

Para definir a capacidade de um cluster de banco de dados do Aurora Serverless v1 usando a AWS CLI, execute o comando modify-current-db-cluster-capacity da AWS CLI e especifique a opção `--capacity`. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Neste exemplo, defina a capacidade de um cluster de banco de dados do Aurora Serverless v1 chamado *`sample-cluster`* como *`64`*.

```
aws rds modify-current-db-cluster-capacity --db-cluster-identifier sample-cluster --
capacity 64
```

API do RDS

É possível definir a capacidade de um cluster de bancos de dados Aurora com a operação da API ModifyCurrentDBClusterCapacity. Especifique o parâmetro `Capacity`. Entre os valores de capacidade válidos estão os seguintes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Visualizar clusters de banco de dados do Aurora Serverless v1

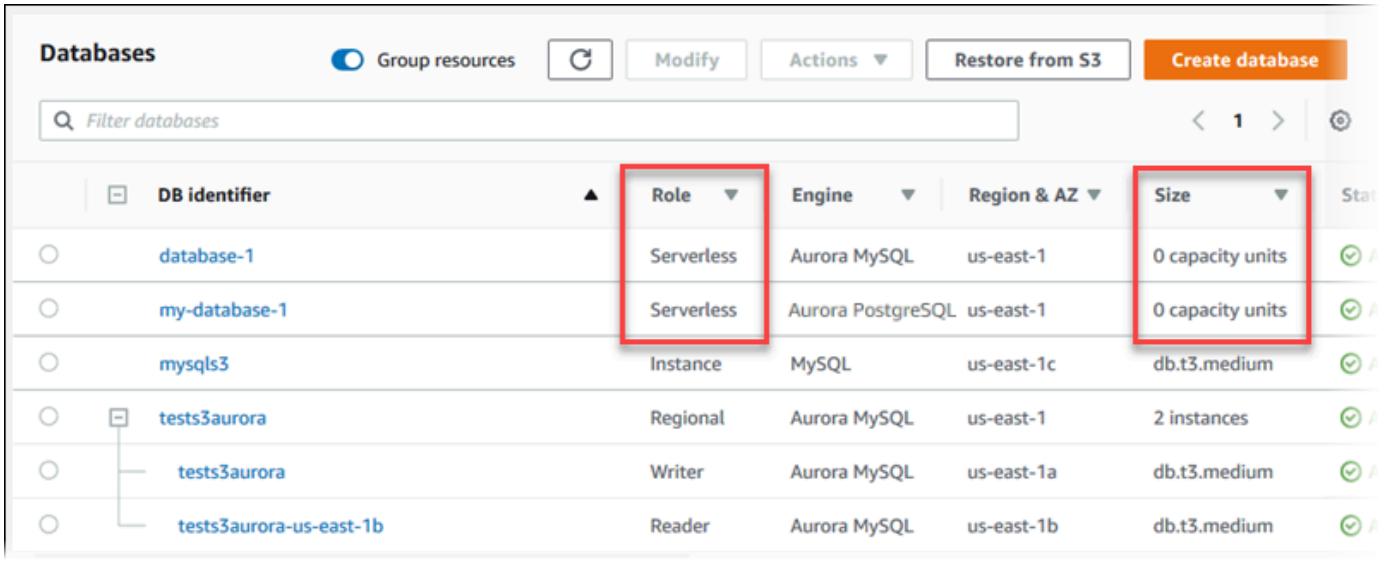
> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Exporte estendido do Amazon RDS. Para ter mais informações, consulte Suporte estendido do RDS.

Depois de criar um ou mais clusters de banco de dados do Aurora Serverless v1, será possível visualizar quais clusters são do tipo Serverless (Sem servidor) e quais são do tipo Instance (Instância). Também é possível visualizar o número atual de unidades de capacidade do Aurora (ACUs) que cada cluster de banco de dados do Aurora Serverless v1 está usando. Cada ACU é uma combinação da capacidade de processamento (CPU) e de memória (RAM).

Como visualizar os clusters de banco de dados do Aurora Serverless v1

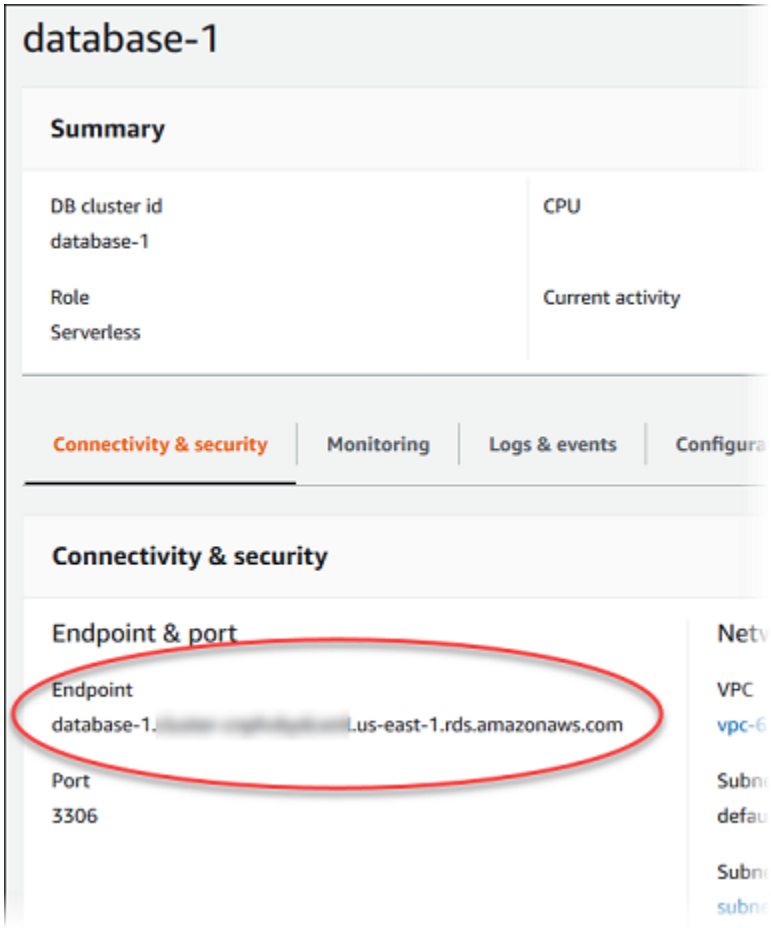
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a Região da AWS em que você criou os clusters de banco de dados do Aurora Serverless v1.
3. No painel de navegação, escolha Bancos de dados.

Para cada cluster de banco de dados, o tipo de cluster de banco de dados é mostrado sob Role (Função). Os clusters de banco de dados do Aurora Serverless v1 mostram Serverless (Sem servidor) como o tipo. É possível visualizar a capacidade atual de um cluster de banco de dados do Aurora Serverless v1 em Size (Tamanho).

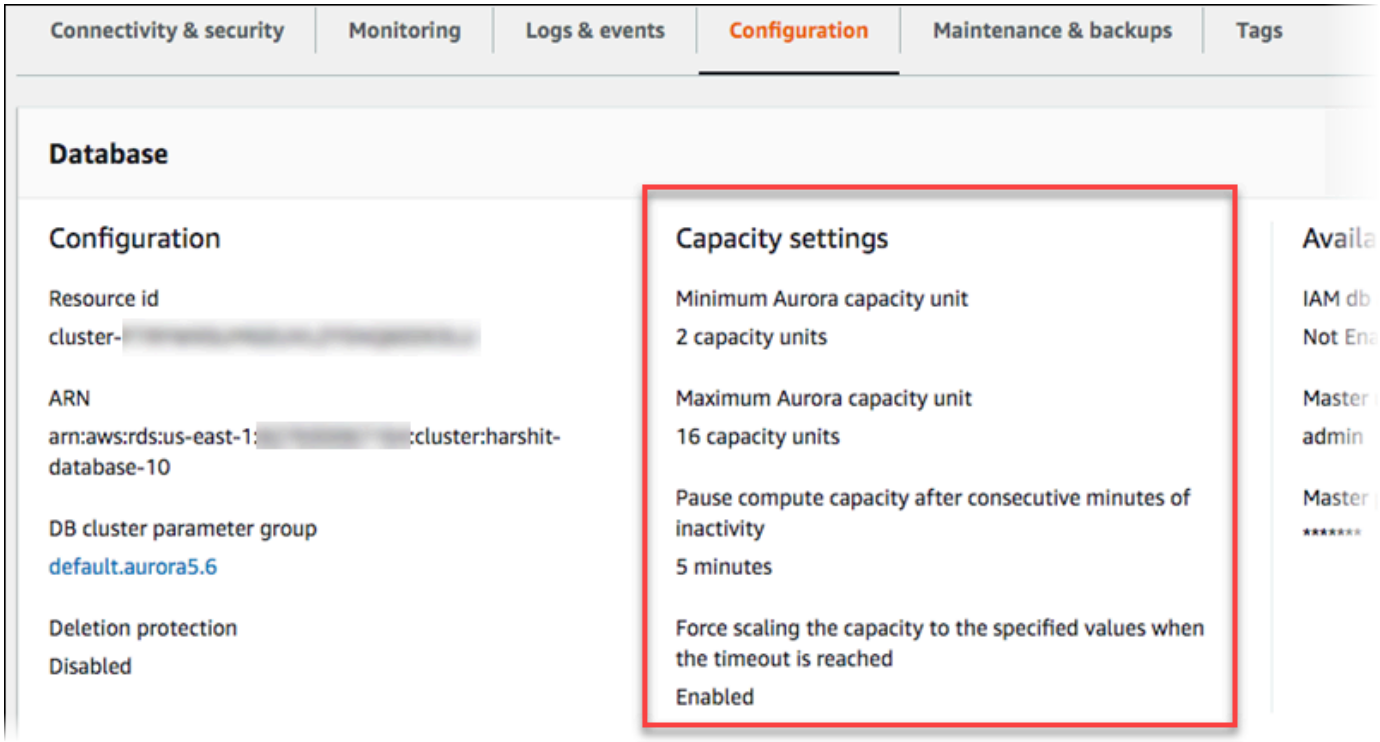


4. Escolha o nome de um cluster de banco de dados do Aurora Serverless v1 para exibir seus detalhes.

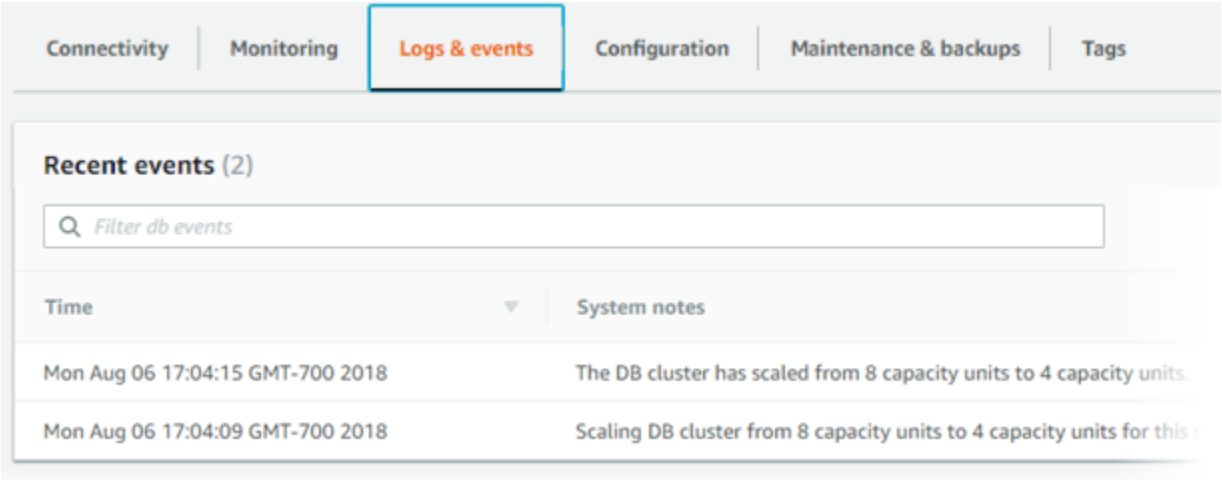
   Na guia Connectivity & security (Conectividade e segurança), observe o endpoint do banco de dados. Use esse endpoint para se conectar ao cluster de banco de dados do Aurora Serverless v1.



Escolha a guia Configuration (Configuração) para visualizar as configurações de capacidade.



Um evento de escalabilidade é gerado quando o cluster de banco de dados é expandido, reduzido, pausado ou retomado. Escolha a guia Logs & events (Logs e eventos) para ver eventos recentes. A imagem a seguir mostra exemplos desses eventos.



Monitorar a capacidade e eventos de escalabilidade do seu cluster de banco de dados do Aurora Serverless v1

Você pode visualizar seu cluster de banco de dados do Aurora Serverless v1 no CloudWatch para monitorar a capacidade alocada ao cluster de banco de dados com a métrica

`ServerlessDatabaseCapacity`. Você também pode monitorar todas as métricas padrão do CloudWatch para o Aurora, como `CPUUtilization`, `DatabaseConnections`, `Queries` e assim por diante.

Você pode fazer com que o Aurora publique alguns ou todos os logs de banco de dados no CloudWatch. Você seleciona os logs a serem publicados habilitando os parâmetros de configuração , como `general_log` e `slow_query_log`, no grupo de parâmetros do cluster de banco de dados associado ao cluster do Aurora Serverless v1. Ao contrário dos clusters provisionados, os clusters do Aurora Serverless v1 não necessitam que você especifique nas configurações do cluster de banco de dados que tipos de log ser carregados no CloudWatch. Os clusters do Aurora Serverless v1 carregam automaticamente todos os logs disponíveis. Quando você desativa um parâmetro de configuração de log, a publicação do log no CloudWatch é interrompida. Você também pode excluir os logs no CloudWatch se eles não forem mais necessários.

Para começar a usar o Amazon CloudWatch para seu cluster de banco de dados do Aurora Serverless v1, consulte Visualizar logs do Aurora Serverless v1 com o Amazon CloudWatch. Para saber mais sobre como monitorar clusters de Aurora banco de dados por meio de CloudWatch, consulte Monitorar eventos de log no Amazon CloudWatch.

Para se conectar a um cluster de banco de dados do Aurora Serverless v1, use o endpoint do banco de dados. Para obter mais informações, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

> **Note**
>
> Não é possível conectar diretamente a instâncias de banco de dados específicas em seus clusters de banco de dados do Aurora Serverless v1.

Excluir um cluster de banco de dados do Aurora Serverless v1

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente.

Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Exporte estendido do Amazon RDS. Para ter mais informações, consulte Suporte estendido do RDS.

Dependendo de como você cria um cluster de banco de dados do Aurora Serverless v1, a proteção contra exclusão pode estar ativada por padrão. Você não pode excluir imediatamente um cluster de Aurora Serverless v1 banco de dados que tenha a proteção de exclusão ativada. Para excluir clusters de Aurora Serverless v1 banco de dados que têm proteção contra exclusão usando o Console de gerenciamento da AWS, primeiro modifique o cluster para remover essa proteção. Para obter informações sobre como usar o AWS CLI para esta tarefa, consulte AWS CLI.

Para desabilitar a proteção contra exclusão usando o Console de gerenciamento da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha DB clusters (Clusters de banco de dados).
3. Escolha seu cluster de Aurora Serverless v1 banco de dados na lista.
4. Escolha Modificar para abrir a configuração do cluster de banco de dados. A página Modificar cluster de banco de dados abre as configurações, configurações de capacidade e outros detalhes de configuração para seu cluster de Aurora Serverless v1 banco de dados. A proteção contra exclusão está na seção Configuração adicional .
5. Desmarque a caixa de seleção Enable deletion protection (Habilitar proteção contra exclusão) no cartão de propriedades Additional configuration (Configuração adicional).
6. Escolha Continue. Você verá o Summary of modifications (Resumo das modificações).
7. Escolha Modify cluster (Modificar cluster) para aceitar o resumo das modificações. Você também pode escolher BAck (Voltar) para modificar suas alterações ou Cancel (Cancelar) para descartar suas alterações.

Depois que a proteção contra exclusão não estiver mais ativa, você poderá excluir seu cluster de Aurora Serverless v1 banco de dados usando Console de gerenciamento da AWSo.

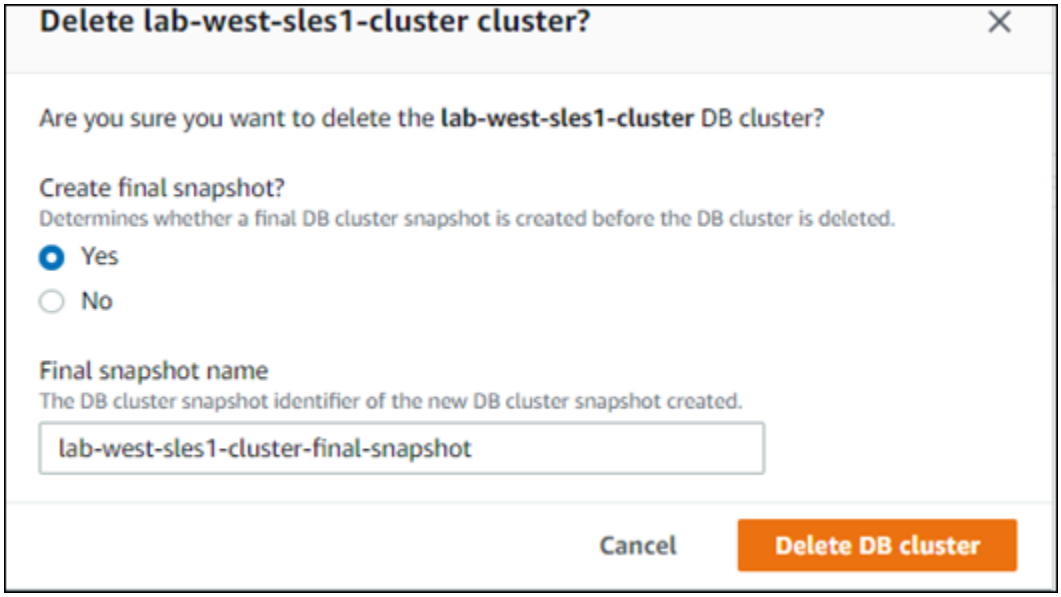
Console

Para excluir um cluster de banco de dados do Aurora Serverless v1

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

2. Na seção Resources (Recursos), escolha DB Clusters (Clusters de banco de dados).
3. Escolha o cluster de banco de dados do Aurora Serverless v1 que você deseja excluir.
4. Em Actions, escolha Delete. Você será solicitado a confirmar se deseja excluir seu cluster de Aurora Serverless v1 banco de dados.
5. Recomendamos que você mantenha as opções pré-selecionadas:

   - Yes (Sim) para Create final snapshot? (Criar snapshot final?)
   - O nome do cluster de banco de dados do Aurora Serverless v1 mais `-final-snapshot` para Final snapshot name (Nome do snapshot final). No entanto, você pode alterar o nome do snapshot final neste campo.



   Se você escolher No (Não) para Create final snapshot? (Criar snapshot final?) você não pode restaurar seu cluster de banco de dados usando snapshots ou recuperação em um ponto anterior no tempo.
6. Escolha Delete DB cluster (Excluir cluster de banco de dados).

Aurora Serverless v1 exclui seu cluster de banco de dados. Se você optar por ter um snapshot final, verá o status do cluster de Aurora Serverless v1 banco de dados mudar para “Backing-up” antes de ser excluído e não aparecer mais na lista.

AWS CLI

Antes de começar, configure sua AWS CLI com o ID da chave de acesso da AWS, a chave de acesso secreta da AWS e a Região da AWS onde está seu cluster de banco de dados do Aurora Serverless v1. Para obter mais informações, consulte Conceitos básicos de configuração no Guia do usuário da AWS Command Line Interface.

Você não pode excluir um cluster de Aurora Serverless v1 banco de dados até depois de desativar a proteção contra exclusão para clusters configurados com essa opção. Se você tentar excluir um cluster que tenha essa opção de proteção ativada, verá a seguinte mensagem de erro.

```
An error occurred (InvalidParameterCombination) when calling the DeleteDBCluster
  operation: Cannot delete protected Cluster, please disable deletion protection and
 try again.
```

Você pode alterar a configuração de proteção contra exclusão do cluster de Aurora Serverless v1 banco de dados usando o AWS CLI comando modify-db-cluster , conforme mostrado no seguinte:

```
aws rds modify-db-cluster --db-cluster-identifier your-cluster-name --no-deletion-
protection
```

Este comando retorna as propriedades revisadas para o cluster de banco de dados especificado. Agora você pode excluir seu cluster de Aurora Serverless v1 banco de dados.

Recomendamos que você sempre crie um snapshot final sempre que excluir um cluster de Aurora Serverless v1 banco de dados. O exemplo a seguir de usar o AWS CLI delete-db-cluster mostra como. Você fornece o nome do cluster de banco de dados e um nome para o snapshot.

Para Linux, macOS ou Unix:

```
aws rds delete-db-cluster --db-cluster-identifier \
  your-cluster-name --no-skip-final-snapshot \
  --final-db-snapshot-identifier name-your-snapshot
```

Para Windows:

```
aws rds delete-db-cluster --db-cluster-identifier ^
  your-cluster-name --no-skip-final-snapshot ^
  --final-db-snapshot-identifier name-your-snapshot
```

Aurora Serverless v1Versões de mecanismos de banco de dados do e do Aurora

> **Important**
>
> A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Todos os clusters do Aurora Serverless v1 que não forem migrados até 31 de março de 2025 serão migrados para o Aurora Serverless v2 durante a janela de manutenção. Se a atualização falhar, durante a janela de manutenção do Amazon Aurora converterá o cluster do Sem Servidor v1 em um cluster provisionado com a versão de mecanismo equivalente. Se aplicável, o Amazon Aurora inscreverá o cluster provisionado convertido no Suporte estendido do Amazon RDS. Para obter mais informações, consulte Suporte estendido do RDS.

O Aurora Serverless v1 está disponível em determinadas Regiões da AWS e somente para versões específicas do Aurora MySQL e do Aurora PostgreSQL. Para obter a lista atual de Regiões da AWS que são compatíveis com o Aurora Serverless v1 e as versões específicas do Aurora MySQL e do Aurora PostgreSQL disponíveis em cada região, consulte Aurora Serverless v1 da.

Aurora Serverless v1O usa seu mecanismo de banco de dados do Aurora associado para identificar versões compatíveis específicas para cada mecanismo de banco de dados compatível, deste modo:

- Aurora MySQL Serverless
- Aurora PostgreSQL Serverless

Quando versões secundárias dos mecanismos de banco de dados ficam disponíveis para o Aurora Serverless v1, elas são aplicadas automaticamente nas várias Regiões da AWS onde o Aurora Serverless v1 está disponível. Em outras palavras, você não precisa atualizar seu cluster de Aurora Serverless v1 banco de dados para obter uma nova versão secundária para o mecanismo de banco de dados do cluster quando ele estiver disponível para Aurora Serverless v1.

Aurora MySQL Serverless

O Aurora Serverless v1 está disponível em determinadas Regiões da AWS e somente para versões específicas do Aurora MySQL. Para obter a lista atual de Regiões da AWS compatíveis com o Aurora

Serverless v1 e as versões específicas do Aurora MySQL disponíveis em cada região, consulte Aurora Serverless v1 com o Aurora MySQL.

Para saber mais sobre aprimoramentos e correções de erros para o Aurora MySQL versão 2, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL nas Notas de versão do Aurora MySQL.

Para usar uma versão mais recente do Aurora MySQL, é possível usar o Aurora Serverless v2. Para saber quais Regiões da AWS e versões do Aurora MySQL podem ser usadas com o Aurora Serverless v2, consulte Aurora Serverless v2 com o Aurora MySQL. Para ter informações de uso do Aurora Serverless v2, consulte Usar o Aurora Serverless v2.

Aurora PostgreSQL Serverless

O Aurora Serverless v1 está disponível em determinadas Regiões da AWS e somente para versões específicas do Aurora PostgreSQL. Para obter a lista atual de Regiões da AWS compatíveis com o Aurora Serverless v1 e as versões específicas do Aurora PostgreSQL disponíveis em cada região, consulte Aurora Serverless v1 com o Aurora PostgreSQL.

Se você quiser usar o Aurora PostgreSQL para o cluster de banco de dados do Aurora Serverless v1, poderá usar as versões compatíveis com o Aurora PostgreSQL 13. Versões menores para Aurora Edição compatível com PostgreSQL incluir apenas alterações compatíveis com versões anteriores. Seu cluster de banco de dados do Aurora Serverless v1 é atualizado de forma transparente quando uma versão secundária do Aurora PostgreSQL se torna disponível para o Aurora Serverless v1 em sua Região da AWS.

Para usar uma versão mais recente do Aurora PostgreSQL, é possível usar o Aurora Serverless v2. Para saber quais Regiões da AWS e versões do Aurora PostgreSQL podem ser usadas com o Aurora Serverless v2, consulte Aurora Serverless v2 com o Aurora PostgreSQL. Para ter informações de uso do Aurora Serverless v2, consulte Usar o Aurora Serverless v2.

Atualizações automáticas da versão secundária do Aurora Serverless v1

Quando versões secundárias dos mecanismos de banco de dados ficam disponíveis para o Aurora Serverless v1, elas são aplicadas automaticamente nas várias Regiões da AWS onde o Aurora Serverless v1 está disponível. Em outras palavras, você não precisa atualizar seu cluster de Aurora Serverless v1 banco de dados para obter uma nova versão secundária para o mecanismo de banco de dados do cluster quando ele estiver disponível para Aurora Serverless v1.

Usar a API de dados do Amazon RDS

Usando a API de dados do RDS, é possível trabalhar com uma interface de serviços web para o cluster de banco de dados do Aurora. A API de dados não exige uma conexão persistente com o cluster do banco de dados. Em vez disso, ela oferece um endpoint HTTP seguro e uma integração com SDKs da AWS. Você pode usar o endpoint para executar instruções SQL sem gerenciar conexões.

Os usuários não precisam transmitir as credenciais com chamadas para a API de dados, porque a API de dados usa credenciais de banco de dados armazenadas no AWS Secrets Manager. Para armazenar credenciais no Secrets Manager, os usuários devem receber as permissões apropriadas para usar o Secrets Manager, bem como a API de dados. Para obter mais informações sobre como autorizar usuários, consulte Autorizar acesso à API de dados do Amazon RDS.

Também é possível usar a API de dados para integrar o Amazon Aurora a outras aplicações da AWS, como AWS Lambda, AWS AppSync e AWS Cloud9. A API de dados oferece uma forma mais segura de usar o AWS Lambda. Ela permite que você acesse o cluster de banco de dados sem precisar configurar uma função do Lambda para acessar recursos em uma nuvem privada virtual (VPC). Para obter mais informações, consulte AWS Lambda, AWS AppSync e AWS Cloud9.

É possível habilitar a API de dados ao criar o cluster de banco de dados do Aurora. Também é possível modificar a configuração posteriormente. Para obter mais informações, consulte Habilitar a API de dados do Amazon RDS.

Depois de habilitar a API de dados, também é possível usar o editor de consultas para executar consultas ad hoc sem configurar uma ferramenta de consulta para acessar o Aurora em uma VPC. Para obter mais informações, consulte Como usar o editor de consultas do Aurora.

Tópicos

- Disponibilidade de região e versão para a API de dados do Amazon RDS
- Usar o IPv6 com a API de dados do Amazon RDS
- Limitações da API de dados do Amazon RDS
- Comparar os comportamentos da API de dados do Amazon RDS para clusters do Aurora Serverless v2 e provisionados com clusters do Aurora Serverless v1
- Autorizar acesso à API de dados do Amazon RDS

- Habilitar a API de dados do Amazon RDS
- Criar um endpoint da Amazon VPC para a API de dados do Amazon RDS (AWS PrivateLink)
- Chamar a API de dados do Amazon RDS
- Usar a biblioteca do cliente Java para a API de dados do RDS
- Processar resultados de consulta da API de dados do Amazon RDS no formato JSON
- Solução de problemas com a API de dados do Amazon RDS
- Registrar em log chamadas da API de dados do Amazon RDS com o AWS CloudTrail
- Monitorar consultas da API de dados do RDS com o Insights de Performance

Disponibilidade de região e versão para a API de dados do Amazon RDS

Para ter informações sobre as regiões e as versões do mecanismo disponíveis para a API de dados, consulte as seções a seguir.

Tipo de cluster	Disponibilidade de regiões e versões
Aurora PostgreSQL provisionado e Sem Servidor v2	API de dados do RDS para o Aurora PostgreSQL Sem Servidor v2 e provisionado
Aurora MySQL provisionado e Sem Servidor v2	API de dados com o Aurora MySQL Sem Servidor v2 e provisionado
Aurora PostgreSQL Sem Servidor v1	API de dados com o Aurora PostgreSQL Sem Servidor v1
Aurora MySQL Sem Servidor v1	API de dados com o Aurora MySQL Sem Servidor v1

Se forem necessários módulos criptográficos validados pelo FIPS 140-2 ao acessar a API de dados por meio de uma interface de linha de comandos ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-2.

Usar o IPv6 com a API de dados do Amazon RDS

A API de dados do Amazon RDS aceita a conectividade IPv6 por meio de endpoints de pilha dupla. Isso permite que você se conecte à API de dados usando endereços IPv6, mantendo a compatibilidade com versões anteriores com IPv4.

Suporte a endpoints IPv6

A API de dados fornece endpoints de pilha dupla que aceitam conexões IPv4 e IPv6. Esses endpoints usam o domínio `.aws` em vez do `.amazonaws.com` tradicional.

Tipos de endpoint disponíveis

Endpoints de pilha dupla públicos

- Formato: `rds-data.region.api.aws`
- Exemplo: `rds-data.us-east-1.api.aws`

Endpoints de pilha dupla FIPS

- Formato: `rds-data-fips.region.api.aws`
- Exemplo: `rds-data-fips.us-east-1.api.aws`

Endpoints IPv6 do PrivateLink

- Disponível por meio de endpoints da VPC com suporte a IPv6
- Permite conectividade IPv6 privada em sua VPC

Endpoints legados somente para IPv4

Os endpoints `.amazonaws.com` existentes continuam aceitando conexões somente IPv4:

- `rds-data.region.amazonaws.com`
- `rds-data-fips.region.amazonaws.com`

> **Note**
>
> Os endpoints legados permanecem inalterados para garantir a compatibilidade com versões anteriores das aplicações existentes.

Usar endpoints IPv6

Para usar o IPv6 com a API de dados, atualize sua aplicação para usar os novos endpoints de pilha dupla. Sua aplicação usará automaticamente o IPv6, se disponível, ou retornará ao IPv4.

Para receber orientação geral sobre como configurar o IPv6 em sua VPC, consulte Migrar para IPv6 no Manual do usuário da Amazon VPC.

É possível configurar endpoints IPv6 de duas maneiras:

- Usando a variável de ambiente: defina `AWS_USE_DUALSTACK_ENDPOINT=true` em seu ambiente IPv6. Os SDKs AWS CLI e AWS construirão automaticamente os endpoints da `api.aws` apropriados sem exigir que você especifique manualmente os URLs dos endpoints.
- Usando URLs de endpoint explícitos: especifique o URL do endpoint de pilha dupla diretamente em seus comandos da AWS CLI ou na configuração do SDK, conforme mostrado nos exemplos abaixo.

AWS CLIConfiguração da

Configure a AWS CLI para usar endpoints IPv6 especificando o URL do endpoint:

Para Linux, macOS ou Unix:

```
aws rds-data execute-statement \
    --endpoint-url https://rds-data.us-east-1.api.aws \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" \
    --database "mydb" \
    --sql "SELECT * FROM users LIMIT 10"
```

Para Windows:

```
aws rds-data execute-statement ^
```

```
 --endpoint-url https://rds-data.us-east-1.api.aws ^
 --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" ^
 --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" ^
 --database "mydb" ^
 --sql "SELECT * FROM users LIMIT 10"
```

Configuração do AWS SDK

Configure os AWS SDKs para usar endpoints de pilha dupla:

Python

```
import boto3

# Create RDS Data API client with IPv6 dual-stack endpoint
client = boto3.client(
    'rds-data',
    endpoint_url='https://rds-data.us-east-1.api.aws'
)

# Execute a SQL statement
response = client.execute_statement(
    resourceArn='arn:aws:rds:us-east-1:123456789012:cluster:my-cluster',
    secretArn='arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret',
    database='mydb',
    sql='SELECT * FROM users LIMIT 10'
)

print(response['records'])
```

Java

```
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import java.net.URI;

// Create RDS Data API client with IPv6 dual-stack endpoint
RdsDataClient client = RdsDataClient.builder()
    .endpointOverride(URI.create("https://rds-data.us-east-1.api.aws"))
    .build();

// Execute a SQL statement
```

```
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .resourceArn("arn:aws:rds:us-east-1:123456789012:cluster:my-cluster")
    .secretArn("arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret")
    .database("mydb")
    .sql("SELECT * FROM users LIMIT 10")
    .build();

ExecuteStatementResponse response = client.executeStatement(request);
System.out.println(response.records());
```

JavaScript

```
const { RDSDataClient, ExecuteStatementCommand } = require("@aws-sdk/client-rds-
data");

// Create RDS Data API client with IPv6 dual-stack endpoint
const client = new RDSDataClient({
    endpoint: "https://rds-data.us-east-1.api.aws"
});

// Execute a SQL statement
const command = new ExecuteStatementCommand({
    resourceArn: "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster",
    secretArn: "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret",
    database: "mydb",
    sql: "SELECT * FROM users LIMIT 10"
});

const response = await client.send(command);
console.log(response.records);
```

Usar AWS PrivateLink com o IPv6

Você pode criar endpoints da VPC para a API de dados que comportem a conectividade IPv6 em sua VPC. Para receber instruções detalhadas sobre a criação de endpoints da VPC para a API de dados, consulte Criar um endpoint da Amazon VPC para a API de dados do Amazon RDS (AWS PrivateLink).

Ao criar um endpoint da VPC para suporte a IPv6, garanta que:

- Sua VPC e sub-redes estão configuradas para comportar IPv6.

- Grupos de segurança permitam tráfego IPv6 nas portas necessárias (normalmente 443 para HTTPS).
- As ACLs de rede sejam configuradas para permitir tráfego IPv6.

Considerações sobre a migração

Ao migrar para endpoints IPv6, pense no seguinte:

- Migração gradual: você pode migrar aplicações gradualmente, uma por vez, atualizando os URLs dos endpoints.
- Compatibilidade de rede: garanta que sua infraestrutura de rede aceite IPv6 antes de migrar.
- Políticas de segurança: atualize as regras do grupo de segurança e as ACLs de rede para permitir o tráfego IPv6, se necessário.
- Monitoramento: atualize as configurações de monitoramento e registro em log para lidar com endereços IPv6.

> **Note**
>
> Endereços de conexão de banco de dados: ao usar endpoints IPv6 para API de dados, as conexões de banco de dados subjacentes e os logs de banco de dados ainda mostrarão endereços IPv4. Esse é o comportamento esperado e não afeta a funcionalidade das aplicações habilitadas para IPv6.

Solução de problemas de conectividade IPv6

Se você tiver problemas com a conectividade IPv6, confira o seguinte:

Configuração de rede

Confira se sua rede comporta IPv6 e se o roteamento IPv6 está configurado corretamente.

Resolução do DNS

Verifique se seu resolvedor de DNS consegue resolver registros AAAA para os endpoints de pilha dupla.

Grupos de segurança

Atualize os grupos de segurança para permitirem tráfego IPv6 nas portas necessárias (normalmente 443 para HTTPS).

Bibliotecas do cliente

Verifique se as bibliotecas do cliente HTTP aceitam conectividade IPv6 e de pilha dupla.

Limitações da API de dados do Amazon RDS

A API de dados do RDS tem as seguintes limitações:

- Só é possível executar consultas da API de dados em instâncias de gravador em um cluster de banco de dados. No entanto, as instâncias de gravador podem aceitar consultas de gravação e leitura.
- Com os bancos de dados globais do Aurora, é possível habilitar a API de dados em clusters de banco de dados primários e secundários. No entanto, um cluster secundário não terá uma instância de gravador até ser promovido a primário. A API de dados requer acesso à instância de gravador para processamento de consultas, mesmo para consultas de leitura. Como resultado, as consultas de leitura e gravação enviadas ao cluster secundário falham enquanto ele não possui uma instância de gravador. Quando um cluster secundário é promovido e tem uma instância de gravador disponível, as consultas da API de dados nessa instância de banco de dados são bem-sucedidas.
- A API de dados não é compatível com as classes de instância de banco de dados T.
- Em relação ao Aurora Serverless v2 e aos clusters de banco de dados provisionados, a API de dados do RDS não é compatível com alguns tipos de dados. Para ver uma lista de tipos compatíveis consulte the section called “API de dados com o Aurora Serverless v2 em comparação com o Aurora Serverless v1”.
- Em bancos de dados do Aurora PostgreSQL versão 14 e posterior, a API de dados só é compatível com `scram-sha-256` para criptografia de senha.
- O limite de tamanho da resposta é de 1 MiB. Se a chamada retornar mais que 1 MiB de dados de resposta, ela será encerrada.
- No Aurora Serverless v1, o número máximo de solicitações por segundo é mil. Para todos os outros bancos de dados compatíveis, não há limite.
- O limite de tamanho da API de dados é de 64 KB por linha no conjunto de resultados obtido pelo banco de dados. Verifique se cada linha em um conjunto de resultados tem até 64 KB.

Comparar os comportamentos da API de dados do Amazon RDS para clusters do Aurora Serverless v2 e provisionados com clusters do Aurora Serverless v1

Os aprimoramentos mais recentes das APIs de dados do Amazon RDS as tornam disponíveis para clusters que usam versões recentes dos mecanismos do PostgreSQL ou do MySQL. Esses clusters podem ser configurados para usar o Aurora Serverless v2 ou classes de instância provisionadas, como `db.r6g` ou `db.r6i`.

As seções a seguir descrevem as diferenças da API de dados do Amazon RDS entre clusters de banco de dados do Aurora Serverless v2 e provisionados e clusters de banco de dados do Aurora Serverless v1. Os clusters de banco de dados do Aurora Serverless v1 usam o modo de mecanismo `serverless`. Os clusters de banco de dados provisionados usam o modo de mecanismo `provisioned`. Um cluster de banco de dados Aurora Serverless v2 também usa o modo de mecanismo `provisioned` e contém uma ou mais instâncias de banco de dados Aurora Serverless v2 com a classe de instância `db.serverless`.

Número máximo de solicitações por segundo

Aurora Serverless v1

As APIs de dados podem fazer até mil solicitações por segundo.

Aurora Serverless v2

As APIs de dados podem fazer um número ilimitado de solicitações por segundo.

Habilitar ou desabilitar a API de dados do Amazon RDS em um banco de dados existente

Aurora Serverless v1

- Com a API do Amazon RDS: use a operação `ModifyCluster` e especifique `True` ou `False`, conforme aplicável, para o parâmetro `EnableHttpEndpoint`.
- Com a AWS CLI: use a operação `modify-db-cluster` com a opção `--enable-http-endpoint` ou `--no-enable-http-endpoint`, conforme aplicável.

Aurora Serverless v2

- Com a API do Amazon RDS: use as operações `EnableHttpEndpoint` e `DisableHttpEndpoint`.
- Com a AWS CLI: use as operações `enable-http-endpoint` e `disable-http-endpoint`.

Eventos do CloudTrail

Aurora Serverless v1

Os eventos das chamadas da API de dados são eventos de gerenciamento. Esses eventos são incluídos automaticamente em uma trilha por padrão. Para obter mais informações, consulte the section called “Excluir eventos da API de dados de uma trilha do CloudTrail (somente o Aurora Serverless v1)”.

Aurora Serverless v2

Os eventos das chamadas da API de dados são eventos de dados. Esses eventos são excluídos automaticamente em uma trilha por padrão. Para obter mais informações, consulte the section called “Incluir eventos da API de dados em uma trilha do CloudTrail”.

Suporte a várias declarações

Aurora Serverless v1

- No Aurora MySQL, não há suporte para várias declarações.
- Para o Aurora PostgreSQL, várias declarações exibem somente a primeira resposta da consulta.

Aurora Serverless v2

Não há suporte a várias declarações. A tentativa de executar várias instruções em uma única chamada de API exibe “`An error occurred (ValidationException) when calling the ExecuteStatement operation: Multistatements aren't supported.`”. Para executar várias instruções, faça chamadas de API `ExecuteStatement` separadas ou use `BatchExecuteStatement` para processamento em lote.

O exemplo a seguir mostra a mensagem de erro resultante de uma chamada de API que tenta executar várias instruções.

```
 aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table; Select * FROM next_table;

                                "An error occurred (ValidationException) when calling
 the ExecuteStatement operation: Multistatements aren't supported.
```

O exemplo a seguir executa várias instruções com chamadas de API `ExecuteStatement` separadas.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table;"

aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM next_table;"
```

Solicitações simultâneas para o mesmo ID de transação

Aurora Serverless v1

As solicitações subsequentes aguardam até que a solicitação atual seja concluída. Sua aplicação precisará lidar com erros de tempo limite se o período de espera for muito longo.

Aurora Serverless v2

Quando a API de dados recebe várias solicitações com o mesmo ID de transação, ela retorna imediatamente este erro:

`DatabaseErrorException: Transaction is still running a query`

Esse erro ocorre em duas situações:

- A aplicação faz solicitações assíncronas (como promessas de JavaScript) usando o mesmo ID de transação.

- Uma solicitação anterior com esse ID de transação ainda está sendo processada.

O exemplo a seguir mostra todas as solicitações executadas em paralelo com `promise.all()`.

```
const api_calls = [];
for (let i = 0; i < 10; i++) {
api_calls.push(
    client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    )
);
}
await Promise.all(api_calls);
```

Para resolver esse erro, aguarde a conclusão da solicitação atual antes de enviar outra solicitação com o mesmo ID de transação ou remova o ID da transação para permitir solicitações paralelas.

O exemplo a seguir mostra uma chamada de API que usa execução sequencial com o mesmo ID de transação.

```
 for (let i = 0; i < 10; i++) {
    await client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    ).promise()
);
}
```

Comportamento de BatchExecuteStatement

Para ter mais informações sobre `BatchExecuteStatement`, consulte BatchExecuteStatement.

Aurora Serverless v1

O objeto de campos gerado no resultado da atualização inclui valores inseridos.

Aurora Serverless v2

- Para o Aurora MySQL, o objeto de campos gerado no resultado da atualização tem valores inseridos.
- Para o Aurora PostgreSQL, o objeto de campos gerado não contém valores.

Comportamento do ExecuteSQL

Para ter mais informações sobre `ExecuteSQL`, consulte ExecuteSQL.

Aurora Serverless v1

A operação `ExecuteSQL` está obsoleta.

Aurora Serverless v2

A operação `ExecuteSQL` não é compatível.

Comportamento de ExecuteStatement

Para ter mais informações sobre `ExecuteStatement`, consulte ExecuteStatement.

Aurora Serverless v1

O parâmetro `ExecuteStatement` é compatível com a recuperação de colunas de matriz multidimensionais e todos os tipos de dados avançados.

Aurora Serverless v2

O parâmetro `ExecuteStatement` não é compatível com colunas de matriz multidimensionais. Ele também não é compatível com determinados tipos de dados do PostgreSQL, como tipos geométricos e monetários. Quando uma API de dados encontra um tipo de dados incompatível, ela retorna este erro: `UnsupportedResultException: The result contains the unsupported data type data_type`.

Para contornar esse problema, converta o tipo de dados não aceito em `TEXT`. O exemplo a seguir converte um tipo de dados não compatível em `TEXT`.

```
SELECT custom_type::TEXT FROM my_table;--
ORSELECT CAST(custom_type AS TEXT) FROM my_table;
```

Para obter uma lista dos tipos de dados compatíveis com cada mecanismo de banco de dados do Aurora, consulte Referência de operações da API de dados do Amazon RDS.

Comportamento do parâmetro de esquema

Aurora Serverless v1

O parâmetro `Schema` não é compatível. Quando você inclui o parâmetro `Schema` em uma chamada de API, a API de dados ignora o parâmetro.

Aurora Serverless v2

O parâmetro `Schema` está obsoleto. Quando você inclui o parâmetro `Schema` em uma chamada de API, a API de Dados retorna este erro: `ValidationException: The schema parameter isn't supported`. O exemplo a seguir mostra uma chamada da API de dados que retorna o erro `ValidationException`.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--schema "your_schema" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Para resolver esse problema, remova o parâmetro `Schema` da chamada de API.

O exemplo a seguir mostra uma chamada da API de dados com o parâmetro `Schema` removido.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Autorizar acesso à API de dados do Amazon RDS

Os usuários poderão invocar operações da API de dados do Amazon RDS (API de dados) somente se estiverem autorizados a fazê-lo. É possível conceder permissão a um usuário para usar a API

de dados anexando uma política do AWS Identity and Access Management (IAM) que defina os privilégios. Também será possível anexar a política a um perfil se você estiver usando perfis do IAM. Uma política gerenciada da AWS, `AmazonRDSDataFullAccess`, inclui permissões para a API de dados.

A política `AmazonRDSDataFullAccess` também inclui permissões para que o usuário obtenha o valor de um segredo do AWS Secrets Manager. Os usuários precisam usar o Secrets Manager para armazenar os segredos que eles podem usar nas chamadas para a API de dados. Com o uso de segredos, os usuários não precisam incluir credenciais de banco de dados para os recursos que eles têm como destino nas chamadas para a API de dados. A API de dados chama de forma transparente o Secrets Manager, que permite (ou nega) a solicitação do usuário para o segredo. Para ter informações sobre como configurar segredos para usar com a API de dados, consulte Armazenar credenciais de banco de dados no AWS Secrets Manager.

A política `AmazonRDSDataFullAccess` concede acesso completo (por meio da API de dados) aos recursos. É possível restringir o escopo definindo suas próprias políticas que especificam o Nome de recurso da Amazon (ARN) de um recurso.

Por exemplo, a política a seguir mostra um exemplo das permissões mínimas necessárias para um usuário acessar a API de dados para o cluster de banco de dados identificado pelo ARN. A política inclui as permissões necessárias para acessar o Secrets Manager e obter autorização para a instância de banco de dados para o usuário.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:prod"
        }
    ]
}
```

Recomendamos usar um ARN específico para o elemento “Recursos” em suas instruções de política (conforme mostrado no exemplo), em vez de um caractere curinga (*).

Trabalhar com autorização baseada em tags

A API de dados do RDS (API de dados) e o Secrets Manager são compatíveis com a autorização baseada em tags. As tags são pares de chave-valor que rotulam um recurso, como um cluster do RDS, com um valor de cadeia de caracteres adicional, por exemplo:

- `environment:production`
- `environment:development`

É possível aplicar tags aos recursos para alocação de custos, suporte de operações, controle de acesso e muitas outras finalidades. (Se você ainda não tiver tags em seus recursos e quiser aplicá-las, saiba mais em Marcar recursos do Amazon RDS.) É possível usar as tags nas instruções de política para limitar o acesso aos clusters do RDS rotulados com essas tags. Como exemplo, um cluster de bancos de dados Aurora pode ter tags que identificam o ambiente como produção ou desenvolvimento.

O exemplo a seguir mostra como é possível usar tags em suas instruções de política. Esta instrução exige que o cluster e o segredo passados na solicitação da API de dados tenham uma tag `environment:production`.

Veja como a política é aplicada: quando um usuário faz uma chamada usando a API de dados, a solicitação é enviada ao serviço. A API de dados primeiro verifica se o ARN de cluster transmitido na solicitação está marcado com `environment:production`. Em seguida, ele chama o Secrets Manager para recuperar o valor do segredo do usuário na solicitação. O Secrets Manager também

verifica se o segredo do usuário está marcado com `environment:production`. Em caso afirmativo, a API de dados usa o valor recuperado para a senha de banco de dados do usuário. Por fim, se isso também estiver correto, a solicitação da API de dados é invocada com êxito para o usuário.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                 "secretsmanager:GetSecretValue"
               ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                  "rds-data:*"
               ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
         }
     ]
}
```

O exemplo mostra ações separadas para `rds-data` e `secretsmanager` para a API de dados e o Secrets Manager. No entanto, é possível combinar ações e definir condições de tag de várias maneiras diferentes para oferecer suporte aos seus casos de uso específicos. Para obter mais informações, consulte Usar políticas baseadas em identidade (políticas do IAM) para o Secrets Manager.

No elemento “Condição” da política, é possível escolher chaves de tag entre as seguintes:

- `aws:TagKeys`
- `aws:ResourceTag/${TagKey}`

Para saber mais sobre tags de recursos e como usar `aws:TagKeys`, consulte Como controlar o acesso a recursos da AWS usando tags de recurso.

> **Note**
>
> A API de dados e o AWS Secrets Manager autorizam usuários. Se não tiver permissões para todas as ações definidas em uma política, você receberá um erro `AccessDeniedException`.

Armazenar credenciais de banco de dados no AWS Secrets Manager

Ao chamar a API de dados do Amazon RDS (API de dados), transmita as credenciais ao cluster de banco de dados do Aurora usando um segredo no Secrets Manager. Para passar credenciais dessa maneira, especifique o nome do segredo ou o nome de recurso da Amazon (ARN) do segredo.

Para armazenar credenciais de cluster de banco de dados em um segredo

1. Use o Secrets Manager para criar um segredo que contenha credenciais para o cluster de bancos de dados Aurora.

   Para obter instruções, consulte Criar um segredo de banco de dados no Guia do usuário do AWS Secrets Manager.
2. Use o console do Secrets Manager para visualizar os detalhes do segredo criado ou execute o comando `aws secretsmanager describe-secret` da AWS CLI.

   Anote o nome e o ARN do segredo. É possível usá-los em chamadas para a API de dados.

Para obter mais informações sobre o uso do Secrets Manager, consulte o Manual do usuário do AWS.

Para entender como o Amazon Aurora lida com o gerenciamento de identidade e acesso, consulte Como o Amazon Aurora funciona com o IAM.

Para obter mais informações sobre como criar uma política do IAM, consulte Criar políticas do IAM no Manual do usuário do IAM. Para obter informações sobre como adicionar uma política do IAM a um usuário, consulte Adicionar e remover permissões de identidade do IAM no Manual do usuário do IAM.

Habilitar a API de dados do Amazon RDS

Para usar a API de dados do Amazon RDS (API de dados), habilite-a para o cluster de banco de dados do Aurora. É possível habilitar a API de dados ao criar ou modificar o cluster de banco de dados.

> **Note**
>
> A disponibilidade da API de dados para o cluster depende da versão do Aurora, do mecanismo de banco de dados e da região da AWS. Em versões mais antigas do Aurora, a API de dados só funciona com clusters do Aurora Serverless v1. Em versões mais recentes do Aurora, a API de dados funciona com clusters que usam tanto instâncias provisionadas quanto do Aurora Serverless v2. Verifique se o cluster pode usar a API de dados em Regiões e mecanismos de banco de dados do Aurora compatíveis com a API de dados do RDS.

Tópicos

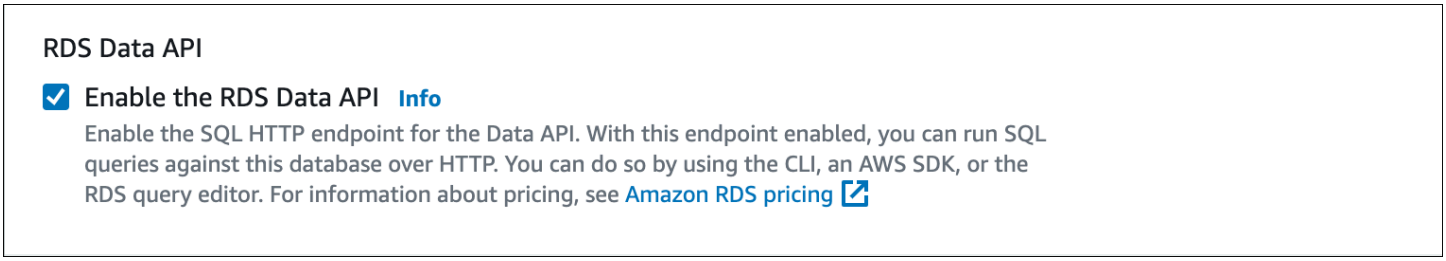
- Habilitar a API de dados do RDS ao criar um banco de dados
- Habilitar a API Data do RDS em um banco de dados existente

Habilitar a API de dados do RDS ao criar um banco de dados

Ao criar um banco de dados compatível com a API de dados do RDS (API de dados), é possível habilitar esse recurso. Os procedimentos a seguir descrevem como fazer isso ao usar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS.

Console

Para habilitar a API de dados ao criar um cluster de banco de dados, marque a caixa de seleção Habilitar a API de dados do RDS na seção Conectividade da página Criar banco de dados, como no screenshot a seguir.



Para ter instruções sobre como criar um cluster de banco de dados do Aurora que possa usar a API de dados do RDS, consulte o seguinte:

- Para clusters do Aurora Serverless v2 e provisionados: Criar um cluster de bancos de dados do Amazon Aurora
- Para Aurora Serverless v1: Criar um cluster de banco de dados do Aurora Serverless v1

AWS CLI

Para habilitar a API de dados ao criar um cluster de banco de dados do Aurora, execute o comando AWS CLIcreate-db-cluster com a opção `--enable-http-endpoint`.

O exemplo a seguir cria um cluster de banco de dados do Aurora PostgreSQL com a API de dados habilitada.

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier my_pg_cluster \
    --engine aurora-postgresql \
    --enable-http-endpoint
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my_pg_cluster ^
    --engine aurora-postgresql ^
```

```
    --enable-http-endpoint
```

API do RDS

Para habilitar a API de dados ao criar um cluster de banco de dados do Aurora, use a operação CreatedBCluster com o valor do parâmetro `EnableHttpEndpoint` definido como `true`.

Habilitar a API Data do RDS em um banco de dados existente

É possível modificar um cluster de banco de dados compatível com a API de dados do RDS (API de dados) para habilitar ou desabilitar esse recurso.

Tópicos

- Habilitar ou desabilitar a API de dados (Aurora Serverless v2 e provisionado)
- Habilitar ou desabilitar a API de dados (somente o Aurora Serverless v1)

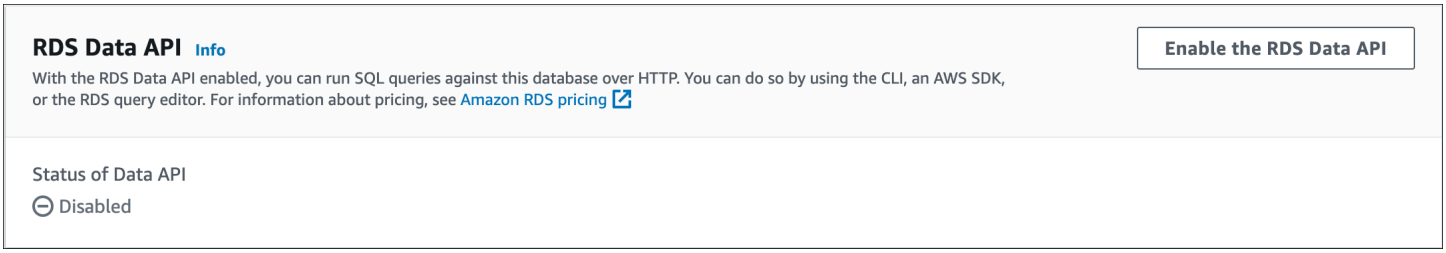
Habilitar ou desabilitar a API de dados (Aurora Serverless v2 e provisionado)

Use os procedimentos a seguir para habilitar ou desabilitar a API de dados em bancos de dados do Aurora Serverless v2 e provisionados. Para habilitar ou desabilitar a API de dados em bancos de dados do Aurora Serverless v1, use os procedimentos em the section called “Habilitar ou desabilitar a API de dados (somente o Aurora Serverless v1)”.

Console

É possível habilitar ou desabilitar a API de dados usando o console do RDS para um cluster de banco de dados que seja compatível com esse recurso. Para fazer isso, abra a página de detalhes do cluster do banco de dados no qual você deseja habilitar ou desabilitar a API de dados e, na guia Conectividade e segurança, vá para a seção API de dados do RDS. Esta seção exibe o status da API de dados e permite habilitá-la ou desabilitá-la.

O screenshot a seguir mostra que a API de dados do RDS não está habilitada.



AWS CLI

Para habilitar ou desabilitar a API de dados em um banco de dados existente, execute o comando enable-http-endpoint ou disable-http-endpoint da AWS CLI e especifique o ARN do cluster de banco de dados.

O exemplo a seguir habilita a API de dados.

Para Linux, macOS ou Unix:

```
aws rds enable-http-endpoint \
    --resource-arn cluster_arn
```

Para Windows:

```
aws rds enable-http-endpoint ^
    --resource-arn cluster_arn
```

API do RDS

Para habilitar ou desabilitar a API de dados em um banco de dados existente, use as operações EnableHttpEndpoint e DisableHttpEndpoint.

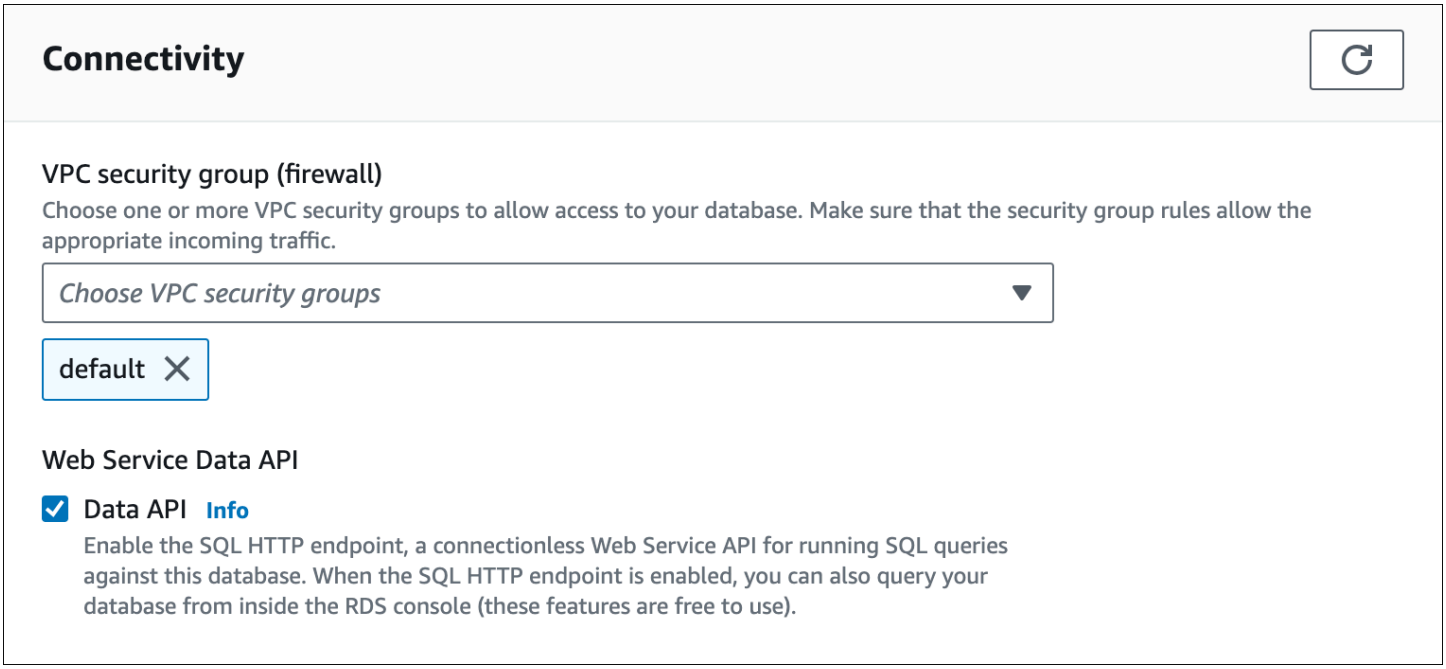
Habilitar ou desabilitar a API de dados (somente o Aurora Serverless v1)

Use os procedimentos a seguir para habilitar ou desabilitar a API de dados em bancos de dados do Aurora Serverless v1. Para habilitar ou desabilitar a API de dados em bancos de dados do Aurora Serverless v2 e provisionados, use os procedimentos em the section called “Habilitar ou desabilitar a API de dados”.

Console

Ao modificar um cluster de banco de dados do Aurora Serverless v1, habilite a API de dados na seção Conectividade do console do RDS.

O screenshot a seguir mostra a API de dados habilitada ao modificar um cluster de banco de dados do Aurora.



Para ter instruções sobre como modificar um cluster de banco de dados do Aurora Serverless v1, consulte Modificar um cluster de banco de dados do Aurora Serverless v1.

AWS CLI

Para habilitar ou desativar a API de dados, execute o comando modify-db-cluster da AWS CLI com o `--enable-http-endpoint` ou o `--no-enable-http-endpoint`, conforme aplicável.

O exemplo a seguir habilita a API de dados no `sample-cluster`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --enable-http-endpoint
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --enable-http-endpoint
```

API do RDS

Para habilitar a API de dados, use a operação ModifyDBCluster e defina o valor de `EnableHttpEndpoint` como `true` ou `false`, conforme aplicável.

Criar um endpoint da Amazon VPC para a API de dados do Amazon RDS (AWS PrivateLink)

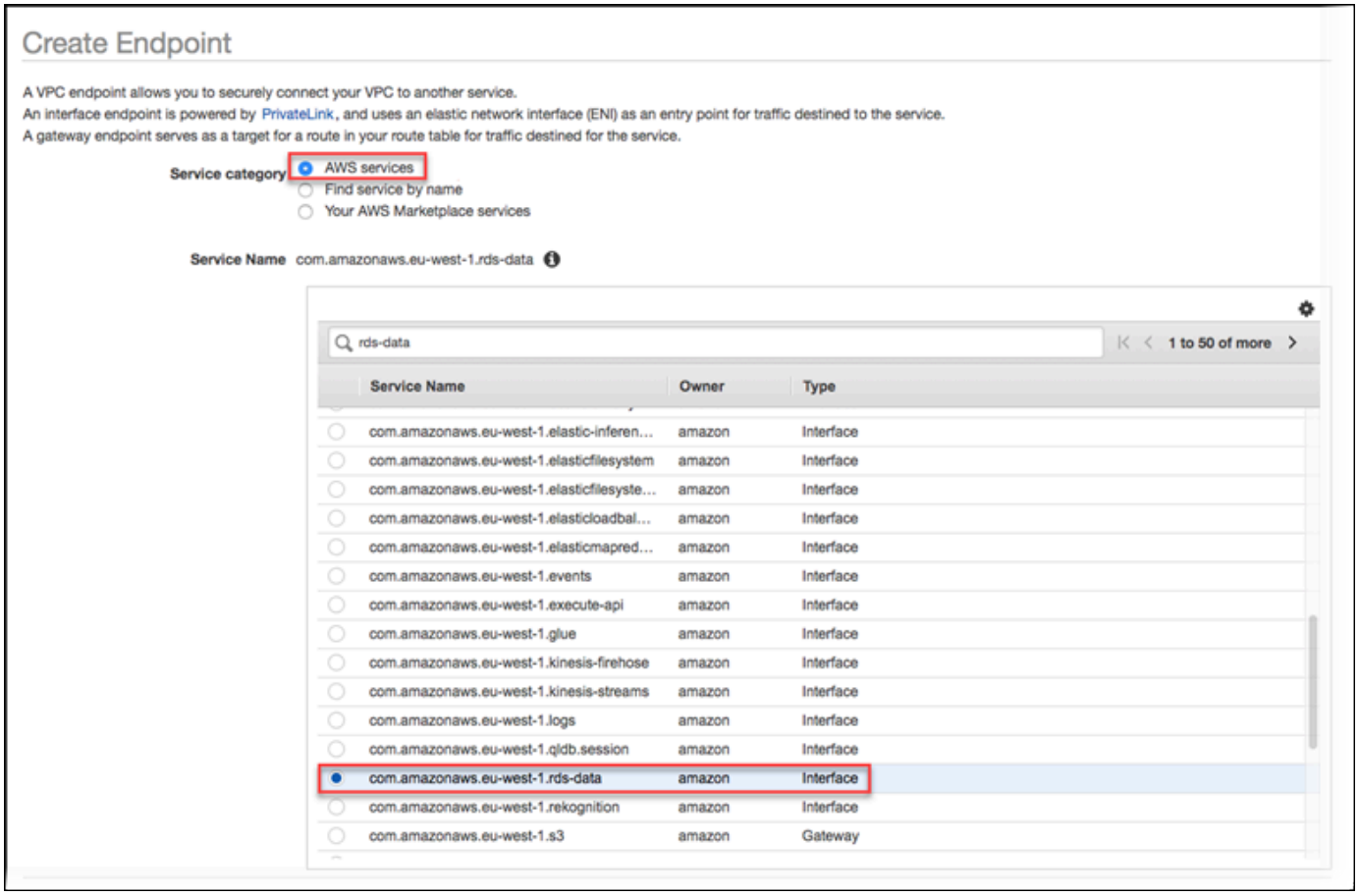
A Amazon VPC permite iniciar recursos da AWS, como clusters de banco de dados e aplicações do Aurora, em uma nuvem privada virtual (VPC). O AWS PrivateLink fornece conectividade privada entre VPCs e produtos da AWS com alta segurança na rede da Amazon. Usando o AWS PrivateLink, é possível criar endpoints da Amazon VPC que permitem que você se conecte a produtos em diferentes contas e VPCs com base na Amazon VPC. Para obter mais informações sobre o AWS PrivateLink, consulte Serviços do VPC endpoint (AWS PrivateLink) no Manual do usuário da Amazon Virtual Private Cloud.

É possível chamar a API de dados do RDS (API de dados) com endpoints da Amazon VPC. O uso de um endpoint da Amazon VPC mantém o tráfego entre aplicações na Amazon VPC e a API de dados na rede da AWS sem usar endereços IP públicos. Os endpoints do Amazon VPC podem ajudá-lo a atender aos requisitos normativos e de compatibilidade relacionados à limitação da conectividade pública com a Internet. Por exemplo, ao usar um endpoint da Amazon VPC, é possível manter o tráfego entre uma aplicação em execução em uma instância do Amazon EC2 e a API de dados nas VPCs que as contêm.

Depois de criar o Amazon VPC endpoint, você pode começar a usá-lo sem fazer alterações no código ou na configuração de sua aplicação.

Como criar um endpoint da Amazon VPC para a API de dados

1. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha Endpoints e Create Endpoint (Criar endpoint).
3. Na página Criar endpoint, para a Categoria de serviço, escolha Serviços da AWS. Em Service Name (Nome do serviço), escolha rds-data.



4. Em VPC, escolha a VPC na qual criar o endpoint.

   Escolha a VPC que contém a aplicação que faz chamadas da API de dados.

5. Em Sub-redes, escolha a sub-rede de cada zona de disponibilidade (AZ) usada pelo serviço da AWS que está executando a aplicação.

   Para criar um Amazon VPC endpoint, especifique o intervalo de endereços IP privados no qual o endpoint estará acessível. Para fazer isso, escolha a sub-rede de cada zona de disponibilidade. Isso restringe o VPC endpoint ao intervalo de endereços IP privados específico para cada zona de disponibilidade e também cria um Amazon VPC endpoint em cada zona de disponibilidade.

6. Em Enable DNS Name (Habilitar nome DNS), selecione Enable for this endpoint (Habilitar para este endpoint).

   O DNS privado resolve o nome de host DNS da API de dados padrão (`https://rds-data.`*`region`*`.amazonaws.com`) para os endereços IP privados associados ao nome de host DNS específico a seu Amazon VPC endpoint. Como resultado, é possível acessar o endpoint da VPC da API de dados usando a AWS CLI ou os SDKs da AWS sem fazer alterações no código nem na configuração para atualizar o URL do endpoint da API de dados.

7. Em Security group (Grupo de segurança), escolha um grupo de segurança para associar ao Amazon VPC endpoint.

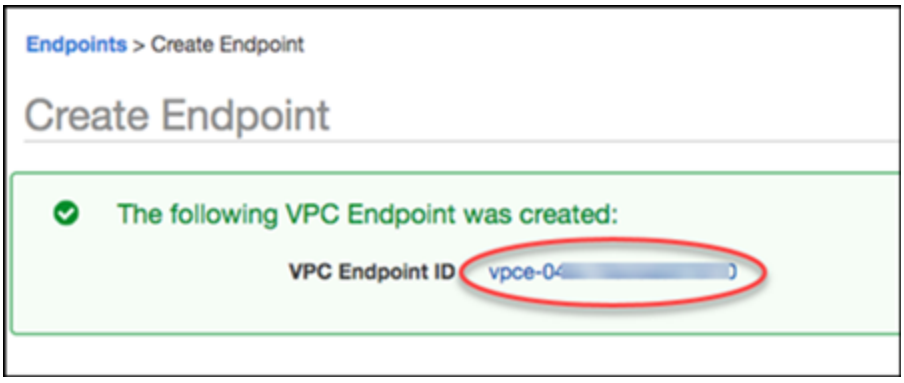
   Escolha o grupo de segurança que permite o acesso ao serviço da AWS que está executando sua aplicação. Por exemplo, se uma instância do Amazon EC2 estiver executando sua aplicação, escolha o grupo de segurança que permite o acesso à instância do Amazon EC2. O grupo de segurança permite que você controle o tráfego para o Amazon VPC endpoint nos recursos em sua VPC.

8. Em Policy (Política), escolha Full Access (Acesso total) para permitir que qualquer pessoa dentro da Amazon VPC acesse a API de dados por meio desse endpoint. Ou escolha Custom (Personalizada) para especificar uma política que limite o acesso.

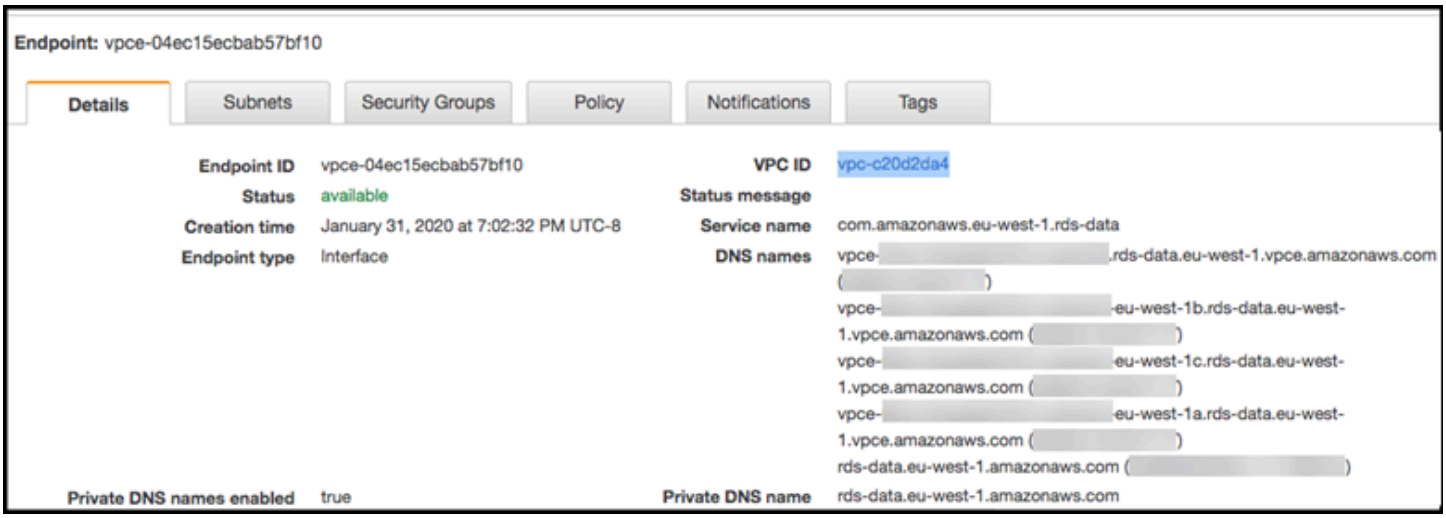
   Se você escolher Custom (Personalizada), insira a política na ferramenta de criação de política.

9. Escolha Create endpoint (Criar endpoint).

Depois que o endpoint for criado, escolha o link no Console de gerenciamento da AWS para visualizar os detalhes do endpoint.



A guia Details (Detalhes) do endpoint mostra os nomes de host de DNS que foram gerados durante a criação do Amazon VPC endpoint.



Você pode usar o endpoint padrão (`rds-data.`*`region`*`.amazonaws.com`) ou um dos endpoints específicos da VPC para chamar a API de dados dentro da Amazon VPC. O endpoint padrão da API de dados roteia automaticamente para o Amazon VPC endpoint. Esse roteamento ocorre porque o nome de host DNS privado foi habilitado quando o Amazon VPC endpoint foi criado.

Ao usar um endpoint da Amazon VPC em uma chamada da API de dados, todo tráfego entre a aplicação e a API de dados permanece nas Amazon VPCs que as contêm. Você pode usar um Amazon VPC endpoint para qualquer tipo de chamada da API de dados. Para ter informações sobre como chamar a API de dados, consulte Chamar a API de dados do Amazon RDS.

Chamar a API de dados do Amazon RDS

Com a API de dados do Amazon RDS (API de dados) habilitada no cluster de banco de dados do Aurora, é possível executar instruções SQL no cluster de bancos de dados do Aurora usando a API de dados ou a AWS CLI. A API de dados é compatível com linguagens de programação aceitas pelos SDKs da AWS. Para obter mais informações, consulte o tópico sobre Ferramentas para criar na AWS.

Tópicos

- Referência de operações da API de dados do Amazon RDS
- Chamar a API de dados do Amazon RDS com a AWS CLI
- Chamar a API de dados do Amazon RDS por meio de uma aplicação Python
- Chamar a API de dados do Amazon RDS por meio de uma aplicação Java

- Controlar o comportamento de tempo limite da API de dados

Referência de operações da API de dados do Amazon RDS

A API de dados do Amazon RDS oferece as operações a seguir para executar instruções SQL.

Operação da API de dados	AWS CLI command	Descrição
ExecuteStatement	aws rds-data execute-statement	Executa uma instrução SQL em um banco de dados.
BatchExecuteStatement	aws rds-data batch-execute-statement	Executa uma instrução SQL em lote em uma matriz de dados para atualização em massa e operações de inserção. Você pode executar uma instrução de linguagem de manipulação de dados (DML) com uma matriz de conjuntos de parâmetros. Uma instrução SQL em lote pode fornecer uma melhoria significativa de performance em relação a instruções de atualização e inserção individuais.

Você pode usar a operação para executar instruções SQL individuais ou para executar transações. Para transações, a API de dados fornece as operações a seguir.

Operação da API de dados	AWS CLI command	Descrição
BeginTransaction	aws rds-data begin-transaction	Inicia uma transação SQL.
CommitTransaction	aws rds-data commit-transaction	Encerra uma transação SQL e confirma as alterações.

Operação da API de dados	AWS CLI command	Descrição
RollbackTransaction	aws rds-data rollback-transaction	Executa uma reversão de uma transação.

As operações para executar instruções SQL e oferecer suporte a transações têm os parâmetros comuns da API de dados e as opções de AWS CLI a seguir. Algumas operações oferecem suporte a outros parâmetros ou outras opções.

Parâmetro da operação da API de dados	AWS CLIOpção de comando da	Obrigatório	Descrição
`resourceArn`	`--resource-arn`	Sim	O nome do recurso da Amazon (ARN) do cluster de banco de dados do Aurora. O cluster deve estar na mesma Conta da AWS que o perfil ou usuário do IAM que invoca a API de dados. Para acessar um cluster em uma conta diferente, assuma um perfil nessa conta.
`secretArn`	`--secret-arn`	Sim	O nome ou o ARN do segredo que permite o acesso ao cluster de banco de dados.

A API de dados do RDS é compatível com os seguintes tipos de dados para o Aurora MySQL:

- `TINYINT(1)`, `BOOLEAN`, `BOOL`
- `TINYINT`
- `SMALLINT` [`SIGNED` | `UNSIGNED`]

- MEDIUMINT [SIGNED | UNSIGNED]
- INT [SIGNED | UNSIGNED]
- BIGINT [SIGNED | UNSIGNED]
- FLOAT
- DOUBLE
- VARCHAR, CHAR, TEXT, ENUM
- VARBINARY, BINARY, BLOB
- DATE, TIME, DATETIME, TIMESTAMP
- DECIMAL
- JSON
- BIT, BIT(N)

A API de dados do RDS é compatível com os seguintes tipos escalares do Aurora PostgreSQL:

- BOOL
- BYTEA
- DATE
- CIDR
- DECIMAL, NUMERIC
- ENUM
- FLOAT8, DOUBLE PRECISION
- INET
- INT, INT4, SERIAL
- INT2, SMALLINT, SMALLSERIAL
- INT8, BIGINT, BIGSERIAL
- JSONB, JSON
- REAL, FLOAT
- TEXT, CHAR(N), VARCHAR, NAME
- TIME
- TIMESTAMP
- UUID

- VECTOR

A API de dados do RDS é compatível com os seguintes tipos de matriz do Aurora PostgreSQL:

- BOOL[], BIT[]
- DATE[]
- DECIMAL[], NUMERIC[]
- FLOAT8[], DOUBLE PRECISION[]
- INT[], INT4[]
- INT2[]
- INT8[], BIGINT[]
- JSON[]
- REAL[], FLOAT[]
- TEXT[], CHAR(N)[], VARCHAR[], NAME[]
- TIME[]
- TIMESTAMP[]
- UUID[]

Você pode usar parâmetros nas chamadas da API de dados para `ExecuteStatement` e `BatchExecuteStatement` e quando executar os comandos AWS CLI e `execute-statement` da `batch-execute-statement`. Para usar um parâmetro, especifique um par nome-valor no tipo de dados `SqlParameter`. Você especifica o valor com o tipo de dados `Field`. A tabela a seguir mapeia tipos de dados Java Database Connectivity (JDBC) para os tipos de dados especificados nas chamadas da API de dados.

Tipo de dados JDBC	Tipo de dados da API de dados
INTEGER, TINYINT, SMALLINT, BIGINT	LONG (ou STRING)
FLOAT, REAL, DOUBLE	DOUBLE
DECIMAL	STRING
BOOLEAN, BIT	BOOLEAN

Tipo de dados JDBC	Tipo de dados da API de dados
`BLOB, BINARY, LONGVARBINARY, VARBINARY`	`BLOB`
`CLOB`	`STRING`
Outros tipos (incluindo tipos relacionados a data e hora)	`STRING`

> **Note**
>
> Você pode especificar o tipo de dados `LONG` ou `STRING` em sua chamada de API de dados para valores `LONG` retornados pelo banco de dados. Recomendamos que você faça isso para evitar perder precisão para números extremamente grandes, o que pode acontecer quando você trabalha com JavaScript.

Determinados tipos, como `DECIMAL` e `TIME`, exigem uma dica para que a API de dados transmita valores `String` ao banco de dados como o tipo correto. Para usar uma dica, inclua valores de `typeHint` no tipo de dados `SqlParameter`. Os valores possíveis para `typeHint` são os seguintes:

- `DATE`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `DATE` para o banco de dados. O formato aceito é `YYYY-MM-DD`.
- `DECIMAL`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `DECIMAL` para o banco de dados.
- `JSON`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `JSON` para o banco de dados.
- `TIME`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `TIME` para o banco de dados. O formato aceito é `HH:MM:SS[.FFF]`.
- `TIMESTAMP`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `TIMESTAMP` para o banco de dados. O formato aceito é `YYYY-MM-DD HH:MM:SS[.FFF]`.
- `UUID`: o valor de parâmetro correspondente `String` é enviado como objeto do tipo `UUID` para o banco de dados.

> **Note**
>
> No momento, a API de dados não é compatível com as matrizes de identificadores únicos universais (UUIDs).

> **Note**
>
> Para o Amazon Aurora PostgreSQL, a API de dados sempre exibe o tipo de dados `TIMESTAMPTZ` do Aurora PostgreSQL no fuso horário UTC.

Chamar a API de dados do Amazon RDS com a AWS CLI

É possível chamar a API de dados do RDS (API de dados) usando a AWS CLI.

Os exemplos a seguir usam a AWS CLI para a API de dados. Para obter mais informações, consulte AWS CLI Referência para a API de dados.

Em cada exemplo, substitua o nome do recurso da Amazon (ARN) do cluster de banco de dados pelo ARN do cluster de banco de dados do Aurora. Além disso, substitua o ARN do segredo pelo ARN do segredo no Secrets Manager que concede acesso ao cluster de banco de dados.

> **Note**
>
> A AWS CLI pode formatar respostas em JSON.

Tópicos

- Iniciar uma transação SQL
- Executar uma instrução SQL
- Executar uma instrução SQL em lote em uma matriz de dados
- Confirmar uma transação SQL
- Reverter uma transação SQL

Iniciar uma transação SQL

Você pode iniciar uma transação SQL usando o comando `aws rds-data begin-transaction` da CLI. A chamada retorna um identificador da transação.

> **Important**
>
> Com a API de dados, uma transação atinge o tempo limite quando não ocorre nenhuma chamada que usa o ID da respectiva transação no espaço de três minutos. Caso uma transação atinja o tempo limite antes de ser confirmada, a API de dados a reverte automaticamente.
> As instruções da linguagem de definição de dados (DDL) do MySQL dentro de uma transação causam uma confirmação implícita. Recomendamos que você execute cada instrução de DDL do MySQL em um comando `execute-statement` separado com a opção `--continue-after-timeout`.

Além das opções comuns, especifique a opção `--database`, que fornece o nome do banco de dados.

Por exemplo, o comando da CLI a seguir inicia uma transação SQL.

Para Linux, macOS ou Unix:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Para Windows:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Este é um exemplo da resposta.

```
{
```

```
    "transactionId": "ABC1234567890xyz"
}
```

Executar uma instrução SQL

Você pode executar uma instrução SQL usando o comando `aws rds-data execute-statement` da CLI.

Você pode executar a instrução SQL em uma transação especificando o identificador da transação com a opção `--transaction-id`. Você pode iniciar uma transação usando o comando `aws rds-data begin-transaction` da CLI. Você pode encerrar e confirmar uma transação usando o comando `aws rds-data commit-transaction` da CLI.

> **Important**
>
> Se você não especificar a opção `--transaction-id`, as alterações resultantes da chamadas serão confirmadas automaticamente.

Além das opções comuns, especifique as opções a seguir:

- `--sql` (obrigatório): uma instrução SQL para ser executada no cluster de banco de dados.
- `--transaction-id` (opcional): o identificador de uma transação que foi iniciada usando o comando `begin-transaction` da CLI. Especifique o ID da transação que você deseja incluir na instrução SQL.
- `--parameters` (opcional): os parâmetros para a instrução SQL.
- `--include-result-metadata | --no-include-result-metadata` (opcional): um valor que indica se os metadados devem ou não ser incluídos no resultado. O padrão é `--no-include-result-metadata`.
- `--database` (opcional): o nome do banco de dados.

  A opção `--database` pode não funcionar quando você executa uma instrução SQL depois de executar `--sql "use database_name;"` na solicitação anterior. Recomendamos que você use a opção `--database` em vez de executar instruções `--sql "use database_name;"`.
- `--continue-after-timeout | --no-continue-after-timeout` (opcional): um valor que indica se a instrução deve continuar a ser executada depois que a chamada excede o intervalo de 45 segundos do tempo limite da API de dados. O padrão é `--no-continue-after-timeout`.

Para instruções DDL (linguagem de definição de dados, data definition language), recomendamos continuar a executar a instrução depois que a chamada expira para evitar erros e a possibilidade de estruturas de dados corrompidos.

- `--format-records-as "JSON"|"NONE"`: um valor opcional que especifica se o conjunto de resultados será formatado como uma string JSON. O padrão é `"NONE"`. Para obter informações de uso sobre o processamento de conjuntos de resultados JSON, consulte Processar resultados de consulta da API de dados do Amazon RDS no formato JSON.

O cluster de banco de dados retorna uma resposta para a chamada.

> **Note**
>
> O limite de tamanho da resposta é de 1 MiB. Se a chamada retornar mais que 1 MiB de dados de resposta, ela será encerrada.
> No Aurora Serverless v1, o número máximo de solicitações por segundo é mil. Para todos os outros bancos de dados compatíveis, não há limite.

Por exemplo, o comando da CLI a seguir executa uma única instrução SQL e omite os metadados nos resultados (o padrão).

Para Linux, macOS ou Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "select * from mytable"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "select * from mytable"
```

Este é um exemplo da resposta.

```
{
    "numberOfRecordsUpdated": 0,
    "records": [
        [
            {
                "longValue": 1
            },
            {
                "stringValue": "ValueOne"
            }
        ],
        [
            {
                "longValue": 2
            },
            {
                "stringValue": "ValueTwo"
            }
        ],
        [
            {
                "longValue": 3
            },
            {
                "stringValue": "ValueThree"
            }
        ]
    ]
}
```

O comando da CLI a seguir executa uma única instrução SQL em uma transação especificando a opção `--transaction-id`.

Para Linux, macOS ou Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Este é um exemplo da resposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

O comando da CLI a seguir executa uma única instrução SQL com parâmetros.

Para Linux, macOS ou Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Este é um exemplo da resposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

O comando da CLI a seguir executa uma instrução SQL da linguagem de definição de dados (DDL). A instrução DDL renomeia a coluna `job` para coluna `role`.

> **Important**
>
> Para instruções DDL, recomendamos continuar a executar a instrução depois que a chamada expira. Quando uma instrução DDL é encerrada antes que ela termine de ser executada, podem ocorrer erros e possivelmente estruturas de dados corrompidos. Para continuar a executar uma instrução depois que uma chamada excede o intervalo de 45 segundos do tempo limite da API de dados do RDS, especifique a opção `--continue-after-timeout`.

Para Linux, macOS ou Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Este é um exemplo da resposta.

```
{
    "generatedFields": [],
    "numberOfRecordsUpdated": 0
}
```

> **Note**
>
> Os dados `generatedFields` não são compatíveis com o Aurora PostgreSQL. Para obter os valores de campos gerados, use a cláusula `RETURNING`. Para obter mais informações, consulte Returning data from modified rows na documentação do PostgreSQL.

Executar uma instrução SQL em lote em uma matriz de dados

Você pode executar uma instrução SQL em lote em uma matriz de dados usando o comando `aws rds-data batch-execute-statement` da CLI. Você pode usar esse comando para executar uma importação em massa ou uma operação de atualização.

Você pode executar a instrução SQL em uma transação especificando o identificador da transação com a opção `--transaction-id`. Você pode iniciar uma transação usando o comando `aws rds-data begin-transaction` da CLI. Você pode encerrar e confirmar uma transação usando o comando `aws rds-data commit-transaction` da CLI.

> **Important**
>
> Se você não especificar a opção `--transaction-id`, as alterações resultantes da chamadas serão confirmadas automaticamente.

Além das opções comuns, especifique as opções a seguir:

- `--sql` (obrigatório): uma instrução SQL para ser executada no cluster de banco de dados.

  > **Tip**
  >
  > Para obter instruções compatíveis com o MySQL, não inclua ponto e vírgula no final do parâmetro `--sql`. Um ponto e vírgula à direita pode causar um erro de sintaxe.

- `--transaction-id` (opcional): o identificador de uma transação que foi iniciada usando o comando `begin-transaction` da CLI. Especifique o ID da transação que você deseja incluir na instrução SQL.
- `--parameter-set` (opcional): os conjuntos de parâmetros para a operação em lote.
- `--database` (opcional): o nome do banco de dados.

O cluster de banco de dados retorna uma resposta para a chamada.

> **Note**
>
> Não há um limite posterior fixo para o número de conjuntos de parâmetros. No entanto, o tamanho máximo da solicitação HTTP enviada via API de dados é 4 MiB. Se a solicitação exceder esse limite, a API de dados exibirá um erro e não processará a solicitação. Este limite de 4 MiB inclui o tamanho dos cabeçalhos HTTP e a notação JSON na solicitação. Assim, o número de conjuntos de parâmetros que você pode incluir depende de uma combinação de fatores, como o tamanho da instrução SQL e o tamanho de cada conjunto de parâmetros.
>
> O limite de tamanho da resposta é de 1 MiB. Se a chamada retornar mais que 1 MiB de dados de resposta, ela será encerrada.
>
> No Aurora Serverless v1, o número máximo de solicitações por segundo é mil. Para todos os outros bancos de dados compatíveis, não há limite.

Por exemplo, o comando da CLI a seguir executa uma instrução SQL em lote em uma matriz de dados com um conjunto de parâmetros.

Para Linux, macOS ou Unix:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" \
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

Para Windows:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
```

```
--sql "insert into mytable values (:id, :val)" ^
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

> **Note**
>
> Não inclua quebras de linha na opção `--parameter-sets`.

Confirmar uma transação SQL

Com o uso do comando `aws rds-data commit-transaction` da CLI, você pode encerrar uma transação SQL que iniciou com `aws rds-data begin-transaction` e confirmar as alterações.

Além das opções comuns, especifique a opção a seguir:

- `--transaction-id` (obrigatório): o identificador de uma transação que foi iniciada usando o comando `begin-transaction` da CLI. Especifique o ID da transação que você deseja encerrar e confirmar.

Por exemplo, o comando da CLI a seguir encerra uma transação SQL e confirma as alterações.

Para Linux, macOS ou Unix:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Este é um exemplo da resposta.

```
{
    "transactionStatus": "Transaction Committed"
}
```

Reverter uma transação SQL

Com o uso do comando `aws rds-data rollback-transaction` da CLI, você pode reverter uma transação SQL que iniciou com `aws rds-data begin-transaction`. Reverter uma transação cancela as alterações feitas nela.

> **Important**
>
> Se o ID da transação expirou, a transação foi revertida automaticamente. Nesse caso, um comando `aws rds-data rollback-transaction` que especifica o ID da transação expirado retorna um erro.

Além das opções comuns, especifique a opção a seguir:

- `--transaction-id` (obrigatório): o identificador de uma transação que foi iniciada usando o comando `begin-transaction` da CLI. Especifique o ID da transação que você deseja reverter.

Por exemplo, o comando da AWS CLI a seguir reverte uma transação SQL.

Para Linux, macOS ou Unix:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Este é um exemplo da resposta.

```
{
    "transactionStatus": "Rollback Complete"
    }
```

Chamar a API de dados do Amazon RDS por meio de uma aplicação Python

É possível chamar a API de dados do Amazon RDS (API de dados) por meio de uma aplicação Python.

Os exemplos a seguir usam o AWSSDK for Python (Boto). Para obter mais informações sobre o Boto, consulte a Documentação do AWSSDK for Python (Boto 3).

Em cada exemplo, substitua o nome do recurso da Amazon (ARN) do cluster de banco de dados pelo ARN do cluster de banco de dados do Aurora. Além disso, substitua o ARN do segredo pelo ARN do segredo no Secrets Manager que concede acesso ao cluster de banco de dados.

Tópicos

- Executar uma consulta SQL
- Executar uma instrução SQL DML
- Executar uma transação SQL

Executar uma consulta SQL

Você pode executar uma instrução `SELECT` e obter os resultados com uma aplicação Python.

O exemplo a seguir executa uma consulta SQL.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

response1 = rdsData.execute_statement(
            resourceArn = cluster_arn,
            secretArn = secret_arn,
            database = 'mydb',
            sql = 'select * from employees limit 3')
```

```
print (response1['records'])
[
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'ROSALEZ'
        },
        {
            'stringValue': 'ALEJANDRO'
        },
        {
            'stringValue': '2016-02-15 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'DOE'
        },
        {
            'stringValue': 'JANE'
        },
        {
            'stringValue': '2014-05-09 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'STILES'
        },
        {
            'stringValue': 'JOHN'
        },
        {
            'stringValue': '2017-09-20 04:34:33.0'
        }
```

```
    ]
]
```

Executar uma instrução SQL DML

Você pode executar uma instrução de linguagem de manipulação de dados (DML) para inserir, atualizar ou excluir dados no banco de dados. Você também pode usar parâmetros em instruções DML.

> **Important**
>
> Se uma chamada não faz parte de uma transação por não incluir o parâmetro `transactionID`, alterações resultantes da chamada são confirmadas automaticamente.

O exemplo a seguir executa um instrução SQL de inserção e usa parâmetros.

```
import boto3

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

rdsData = boto3.client('rds-data')


param1 = {'name':'firstname', 'value':{'stringValue': 'JACKSON'}}
param2 = {'name':'lastname', 'value':{'stringValue': 'MATEO'}}
paramSet = [param1, param2]

response2 = rdsData.execute_statement(resourceArn=cluster_arn,
                                      secretArn=secret_arn,
                                      database='mydb',
                                      sql='insert into employees(first_name, last_name)
 VALUES(:firstname, :lastname)',
                                      parameters = paramSet)

print (response2["numberOfRecordsUpdated"])
```

Executar uma transação SQL

Você pode iniciar uma transação SQL, execute uma ou mais instruções SQL e confirme as alterações com uma aplicação Python.

> **Important**
>
> Uma transação expira se não há chamadas que usam o ID da transação em três minutos. Se uma transação expira antes de ser confirmada, ela é revertida automaticamente.
> Se você não especificar um ID de transação, as alterações resultantes da chamadas serão confirmadas automaticamente.

O exemplo a seguir executa uma transação SQL que insere uma linha em uma tabela.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

tr = rdsData.begin_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb')

response3 = rdsData.execute_statement(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb',
     sql = 'insert into employees(first_name, last_name) values('XIULAN', 'WANG')',
     transactionId = tr['transactionId'])

cr = rdsData.commit_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     transactionId = tr['transactionId'])

cr['transactionStatus']
'Transaction Committed'
```

```
response3['numberOfRecordsUpdated']
1
```

> **Note**
>
> Se você executar uma instrução de linguagem de definição de dados (DDL), recomendamos continuar a executar a instrução depois que a chamada expira. Quando uma instrução DDL é encerrada antes que ela termine de ser executada, podem ocorrer erros e possivelmente estruturas de dados corrompidos. Para continuar a executar uma instrução depois que uma chamada ultrapassa o intervalo de tempo limite de 45 segundos da API de dados do RDS, defina o parâmetro `continueAfterTimeout` como `true`.

Chamar a API de dados do Amazon RDS por meio de uma aplicação Java

É possível chamar a API de dados do Amazon RDS (API de dados) por meio de uma aplicação Java.

Os exemplos a seguir usam o AWS SDK for Java. Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Java.

Em cada exemplo, substitua o nome do recurso da Amazon (ARN) do cluster de banco de dados pelo ARN do cluster de banco de dados do Aurora. Além disso, substitua o ARN do segredo pelo ARN do segredo no Secrets Manager que concede acesso ao cluster de banco de dados.

Tópicos

- Executar uma consulta SQL
- Executar uma transação SQL
- Executar uma operação SQL em lote

Executar uma consulta SQL

Você pode executar uma instrução `SELECT` e obter os resultados com uma aplicação Java.

O exemplo a seguir executa uma consulta SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
```

```
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementResult;
import com.amazonaws.services.rdsdata.model.Field;

import java.util.List;

public class FetchResultsExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    ExecuteStatementRequest request = new ExecuteStatementRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb")
            .withSql("select * from mytable");

    ExecuteStatementResult result = rdsData.executeStatement(request);

    for (List<Field> fields: result.getRecords()) {
      String stringValue = fields.get(0).getStringValue();
      long numberValue = fields.get(1).getLongValue();

      System.out.println(String.format("Fetched row: string = %s, number = %d",
 stringValue, numberValue));
    }
  }
}
```

Executar uma transação SQL

Você pode iniciar uma transação SQL, execute uma ou mais instruções SQL e confirme as alterações com uma aplicação Java.

> **Important**
>
> Uma transação expira se não há chamadas que usam o ID da transação em três minutos. Se uma transação expira antes de ser confirmada, ela é revertida automaticamente.

Se você não especificar um ID de transação, as alterações resultantes da chamadas serão confirmadas automaticamente.

O exemplo a seguir executa uma transação SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BeginTransactionRequest;
import com.amazonaws.services.rdsdata.model.BeginTransactionResult;
import com.amazonaws.services.rdsdata.model.CommitTransactionRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;

public class TransactionExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BeginTransactionRequest beginTransactionRequest = new BeginTransactionRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb");
    BeginTransactionResult beginTransactionResult =
 rdsData.beginTransaction(beginTransactionRequest);
    String transactionId = beginTransactionResult.getTransactionId();

    ExecuteStatementRequest executeStatementRequest = new ExecuteStatementRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table VALUES ('hello world!')");
    rdsData.executeStatement(executeStatementRequest);

    CommitTransactionRequest commitTransactionRequest = new CommitTransactionRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN);
```

```
    rdsData.commitTransaction(commitTransactionRequest);
  }
}
```

> **Note**
>
> Se você executar uma instrução de linguagem de definição de dados (DDL), recomendamos continuar a executar a instrução depois que a chamada expira. Quando uma instrução DDL é encerrada antes que ela termine de ser executada, podem ocorrer erros e possivelmente estruturas de dados corrompidos. Para continuar a executar uma instrução depois que uma chamada ultrapassa o intervalo de tempo limite de 45 segundos da API de dados do RDS, defina o parâmetro `continueAfterTimeout` como `true`.

Executar uma operação SQL em lote

Você pode executar operações de atualização e inserção em massa em uma matriz de dados com uma aplicação Java. Você pode executar uma instrução DML com matriz de conjuntos de parâmetros.

> **Important**
>
> Se você não especificar um ID de transação, as alterações resultantes da chamadas serão confirmadas automaticamente.

O exemplo a seguir executa uma operação de inserção em lote.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BatchExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.Field;
import com.amazonaws.services.rdsdata.model.SqlParameter;

import java.util.Arrays;

public class BatchExecuteExample {
```

```
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
      AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BatchExecuteStatementRequest request = new BatchExecuteStatementRequest()
            .withDatabase("test")
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table2 VALUES (:string, :number)")
            .withParameterSets(Arrays.asList(
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("Hello")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(1L))
                    ),
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("World")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(2L))
                    )
            ));

    rdsData.batchExecuteStatement(request);
  }
}
```

Controlar o comportamento de tempo limite da API de dados

Todas as chamadas para a API de dados são síncronas. Suponha que você realize uma operação da API de dados que execute uma instrução SQL, como `INSERT` ou `CREATE TABLE`. Se a chamada da API de dados for retornada com êxito, o processamento de SQL será concluído quando a chamada for retornada.

Por padrão, a API de dados cancelará uma operação e retornará um erro de tempo limite se o processamento da operação não for concluído em 45 segundos. Nesse caso, os dados não são inseridos, a tabela não é criada e assim por diante.

É possível usar a API de dados para realizar operações de longa duração que não podem ser concluídas em 45 segundos. Se você espera que uma operação, como uma operação `INSERT` em massa ou uma operação de DDL em uma tabela grande, leve mais de 45 segundos, especifique o parâmetro `continueAfterTimeout` para a operação `ExecuteStatement`. A aplicação ainda recebe o erro de tempo limite. No entanto, a operação continua em execução e não é cancelada. Para obter um exemplo, consulte Executar uma transação SQL.

Se o SDK da AWS da sua linguagem de programação tiver um período de tempo limite próprio para chamadas de API ou conexões de soquete HTTP, todos esses períodos de tempo limite deverão ser superiores a 45 segundos. Para alguns SDKs, o período de tempo limite é inferior a 45 segundos por padrão. Recomendamos definir qualquer período de tempo limite específico do SDK ou do cliente para pelo menos um minuto. Isso evita a possibilidade de a aplicação receber um erro de tempo limite e garante que a operação da API de dados seja concluída com êxito. Dessa forma, você pode ter certeza sobre se deve tentar novamente a operação ou não.

Por exemplo, suponha que o SDK retorne um erro de tempo limite para a aplicação, mas a operação da API de dados ainda seja concluída dentro do respectivo intervalo de tempo limite. Nesse caso, repetir a operação pode inserir dados duplicados ou produzir resultados incorretos. O SDK pode repetir a operação automaticamente, originando dados incorretos sem nenhuma ação da aplicação.

O intervalo de tempo limite é especialmente importante para o SDK para Java 2. Nesse SDK, o tempo limite da chamada de API e o tempo limite do soquete HTTP são de 30 segundos por padrão. Veja um exemplo de como configurar o tempo limite para um valor mais alto:

```
public RdsDataClient createRdsDataClient() {
    return RdsDataClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .httpClientBuilder(createHttpClientBuilder())
        .credentialsProvider(defaultCredentialsProvider()) // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
```

```
    return ApacheHttpClient.builder() // Change this to your desired HttpClient
        .socketTimeout(Duration.ofSeconds(60));
}
```

Veja um exemplo equivalente usando o cliente de dados assíncrono:

```
public static RdsDataAsyncClient createRdsDataAsyncClient() {
    return RdsDataAsyncClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .credentialsProvider(defaultCredentialsProvider())  // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return NettyNioAsyncHttpClient.builder() // Change this to your desired
 AsyncHttpClient
        .readTimeout(Duration.ofSeconds(60));
}
```

Usar a biblioteca do cliente Java para a API de dados do RDS

É possível baixar e usar uma biblioteca do cliente Java para a API de dados do RDS (API de dados). Essa biblioteca do cliente Java fornece uma forma alternativa de usar a API de dados. Usando essa biblioteca, é possível associar as classes no lado do cliente a solicitações e respostas da API de dados. Esse suporte ao mapeamento pode facilitar a integração com alguns tipos específicos de Java, como `Date`, `Time` e `BigDecimal`.

Baixar a biblioteca cliente Java para API de dados

A biblioteca de cliente Java da API de dados é de código aberto no GitHub no seguinte local:

https://github.com/awslabs/rds-data-api-client-library-java

É possível criar a biblioteca manualmente a partir de arquivos de origem, mas a melhor prática é consumir a biblioteca usando o gerenciamento de dependência do Apache Maven. Adicione a dependência a seguir ao arquivo POM Maven.

Para a versão 2.x, que é compatível com o AWS SDK 2.x, use o seguinte:

```
<dependency>
   <groupId>software.amazon.rdsdata</groupId>
   <artifactId>rds-data-api-client-library-java</artifactId>
   <version>2.0.0</version>
</dependency>
```

Para a versão 1.x, que é compatível com o AWS SDK 1.x, use o seguinte:

```
<dependency>
    <groupId>software.amazon.rdsdata</groupId>
    <artifactId>rds-data-api-client-library-java</artifactId>
    <version>1.0.8</version>
</dependency>
```

Exemplos de biblioteca cliente Java

É possível encontrar a seguir alguns exemplos comuns de uso da biblioteca de cliente Java da API de dados. Estes exemplos pressupõem que você tenha uma tabela `accounts` com duas colunas: `accountId` e `name`. Você também tem o seguinte objeto de transferência de dados (DTO).

```
public class Account {
    int accountId;
    String name;
    // getters and setters omitted
}
```

A biblioteca de cliente permite passar DTOs como parâmetros de entrada. O exemplo a seguir mostra como DTOs do cliente são mapeados para conjuntos de parâmetros de entrada.

```
var account1 = new Account(1, "John");
var account2 = new Account(2, "Mary");
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParamSets(account1, account2)
```

```
        .execute();
```

Em alguns casos, é mais fácil trabalhar com valores simples como parâmetros de entrada. Para isso, siga a seguinte sintaxe.

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
        .withParameter("accountId", 3)
        .withParameter("name", "Zhang")
        .execute();
```

Veja a seguir outro exemplo que funciona com valores simples como parâmetros de entrada.

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(?, ?)", 4, "Carlos")
        .execute();
```

A biblioteca de cliente fornece mapeamento automático para DTOs quando o resultado é retornado. Os exemplos a seguir mostram como o resultado é mapeado para os DTOs.

```
List<Account> result = client.forSql("SELECT * FROM accounts")
         .execute()
         .mapToList(Account.class);

Account result = client.forSql("SELECT * FROM accounts WHERE account_id = 1")
         .execute()
         .mapToSingle(Account.class);
```

Em vários casos, o conjunto de resultados do banco de dados inclui apenas um valor. Para simplificar a recuperação de tais resultados, a biblioteca de cliente oferece a API a seguir:

```
int numberOfAccounts = client.forSql("SELECT COUNT(*) FROM accounts")
         .execute()
         .singleValue(Integer.class);
```

> **Note**
>
> A função `mapToList` converte um conjunto de resultados do SQL em uma lista de objetos definida pelo usuário. Não é possível usar a instrução

`.withFormatRecordsAs(RecordsFormatType.JSON)` em uma chamada de `ExecuteStatement` para a biblioteca do cliente Java, porque ela atende ao mesmo propósito. Para obter mais informações, consulte Processar resultados de consulta da API de dados do Amazon RDS no formato JSON.

Processar resultados de consulta da API de dados do Amazon RDS no formato JSON

Ao chamar a operação `ExecuteStatement`, você pode optar por ter os resultados da consulta retornados como uma string no formato JSON. Dessa forma, é possível usar os recursos de análise JSON da linguagem de programação para interpretar e reformatar o conjunto de resultados. Isso pode ajudar a evitar a necessidade de escrever código adicional para percorrer o conjunto de resultados e interpretar cada valor de coluna.

Para solicitar o conjunto de resultados no formato JSON, transmita o parâmetro `formatRecordsAs` opcional com um valor de `JSON`. O conjunto de resultados formatado em JSON é retornado no campo `formattedRecords` da estrutura `ExecuteStatementResponse`.

A ação `BatchExecuteStatement` não retorna um conjunto de resultados. Dessa forma, a opção JSON não se aplica a essa ação.

Para personalizar as chaves na estrutura de hash JSON, defina aliases de coluna no conjunto de resultados. Você pode fazer isso usando a cláusula `AS` na lista de colunas da consulta SQL.

É possível usar o recurso JSON para tornar o conjunto de resultados mais fácil de ler e mapear o respectivo conteúdo para frameworks específicos de linguagem. Como o volume do conjunto de resultados com codificação ASCII é maior do que a representação padrão, talvez você escolha a representação padrão para consultas que retornem números de linhas ou valores de coluna grandes que consomem mais memória do que o disponível para a aplicação.

Tópicos

- Recuperar resultados de consulta no formato JSON
- Mapeamento de tipo de dados
- Solução de problemas
- Exemplos

Recuperar resultados de consulta no formato JSON

Para receber o conjunto de resultados como uma string JSON, inclua `.withFormatRecordsAs(RecordsFormatType.JSON)` na chamada `ExecuteStatement`. O valor de retorno volta como uma string JSON no campo `formattedRecords`. Nesse caso, `columnMetadata` é `null`. Os rótulos de coluna são as chaves do objeto que representa cada linha. Esses nomes de coluna são repetidos para cada linha do conjunto de resultados. Os valores de coluna são strings entre aspas, valores numéricos ou valores especiais que representam `true`, `false` ou `null`. Os metadados de coluna, como restrições de comprimento e o tipo preciso para números e strings, não são preservados na resposta JSON.

Se você omitir a chamada ao `.withFormatRecordsAs()` ou especificar um parâmetro de `NONE`, o conjunto de resultados será retornado em formato binário usando os campos `Records` e `columnMetadata`.

Mapeamento de tipo de dados

Os valores SQL no conjunto de resultados são mapeados para um conjunto menor de tipos JSON. Os valores são representados em JSON como strings, números e algumas constantes especiais, como `true`, `false` e `null`. Você pode converter esses valores em variáveis em sua aplicação, usando digitação forte ou fraca, conforme apropriado para sua linguagem de programação.

Tipo de dados JDBC	Tipo de dados do JSON
`INTEGER`, `TINYINT`, `SMALLINT`, `BIGINT`	Número por padrão. String se a opção `LongReturnType` for definida como `STRING`.
`FLOAT`, `REAL`, `DOUBLE`	Número
`DECIMAL`	String por padrão. Número se a opção `DecimalReturnType` for definida como `DOUBLE_OR_LONG` .
`STRING`	String
`BOOLEAN`, `BIT`	Booleano
`BLOB`, `BINARY`, `VARBINARY` , `LONGVARBINARY`	String na codificação base64.

Tipo de dados JDBC	Tipo de dados do JSON
`CLOB`	String
`ARRAY`	Array
`NULL`	`null`
Outros tipos (incluindo tipos relacionados a data e hora)	String

Solução de problemas

A resposta JSON é limitada a 10 megabytes. Se a resposta for maior que esse limite, seu programa receberá o erro `BadRequestException`. Nesse caso, você pode resolvê-lo com uma das seguintes técnicas:

- Reduza o número de linhas no conjunto de resultados. Para fazer isso, adicione uma cláusula `LIMIT`. Você pode dividir um grande conjunto de resultados em vários menores enviando várias consultas com as cláusulas `LIMIT` e `OFFSET`.

  Se o conjunto de resultados incluir linhas filtradas pela lógica da aplicação, será possível remover essas linhas do conjunto de resultados adicionando mais condições à cláusula `WHERE`.
- Reduza o número de colunas no conjunto de resultados. Para fazer isso, remova itens da lista de seleção da consulta.
- Reduza os rótulos da coluna usando aliases de coluna na consulta. Cada nome de coluna é repetido na string JSON para cada linha no conjunto de resultados. Dessa forma, um resultado de consulta com nomes de coluna longos e muitas linhas pode exceder o limite de tamanho. Especificamente, use aliases de coluna para expressões complicadas a fim de evitar que a expressão inteira seja repetida na string JSON.
- Embora com o SQL você possa usar aliases de coluna para produzir um conjunto de resultados com mais de uma coluna com o mesmo nome, nomes de chave duplicados não são permitidos no JSON. A API de dados do RDS gerará um erro se você solicitar o conjunto de resultados no formato JSON e mais de uma coluna tiver o mesmo nome. Dessa forma, todos os rótulos das colunas deverão ter nomes exclusivos.

Exemplos

Os exemplos de Java a seguir mostram como chamar o `ExecuteStatement` com a resposta como uma string formatada em JSON e como interpretar o conjunto de resultados. Substitua os valores apropriados para os parâmetros *`databaseName`*, *`secretStoreArn`* e *`clusterArn`*.

O exemplo Java a seguir demonstra uma consulta que retorna um valor numérico decimal no conjunto de resultados. As chamadas ao `assertThat` verificam se os campos da resposta têm as propriedades esperadas com base nas regras para conjuntos de resultados JSON.

Este exemplo funciona com o esquema e os dados de amostra a seguir:

```
create table test_simplified_json (a float);
insert into test_simplified_json values(10.0);
```

```
public void JSON_result_set_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request);
}
```

O valor do campo `formattedRecords` do programa anterior é:

```
[{"a":10.0}]
```

Os campos `Records` e `ColumnMetadata` na resposta são nulos devido à presença do conjunto de resultados JSON.

O exemplo de Java a seguir demonstra uma consulta que retorna um valor numérico decimal no conjunto de resultados. No exemplo, o `getFormattedRecords` é chamado para retornar somente a string formatada em JSON e ignorar os outros campos de resposta que estão em branco ou são nulos. O exemplo desserializa o resultado em uma estrutura que representa uma lista de registros. Cada registro tem campos cujos nomes correspondem aos aliases de coluna do conjunto de resultados. Essa técnica simplifica o código que analisa o conjunto de resultados. Sua aplicação

não precisa percorrer as linhas e colunas do conjunto de resultados e converter cada valor no tipo apropriado.

Este exemplo funciona com o esquema e os dados de amostra a seguir:

```
create table test_simplified_json (a int);
insert into test_simplified_json values(17);
```

```
public void JSON_deserialization_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request)
      .getFormattedRecords();

/* Turn the result set into a Java object, a list of records.
   Each record has a field 'a' corresponding to the column
   labelled 'a' in the result set. */
    private static class Record { public int a; }
    var recordsList = new ObjectMapper().readValue(
        response, new TypeReference<List<Record>>() {
        });
}
```

O valor do campo `formattedRecords` do programa anterior é:

```
[{"a":17}]
```

Para recuperar a coluna a da linha de resultado 0, a aplicação fará referência a `recordsList.get(0).a`.

Em contrapartida, o exemplo de Java a seguir mostra o tipo de código necessário para criar uma estrutura de dados que contenha o conjunto de resultados quando o formato JSON não é usado. Nesse caso, cada linha do conjunto de resultados contém campos com informações sobre um único usuário. Para criar uma estrutura de dados a fim de representar o conjunto de resultados, é necessário examinar as linhas. Para cada linha, o código recupera o valor de cada campo, realiza

uma conversão de tipo apropriada e atribui o resultado ao campo correspondente no objeto que representa a linha. Depois, o código adiciona o objeto que representa cada usuário à estrutura de dados que representa todo o conjunto de resultados. Se a consulta tiver sido alterada para reordenar, adicionar ou remover campos no conjunto de resultados, o código da aplicação também precisará mudar.

```
/* Verbose result-parsing code that doesn't use the JSON result set format */
for (var row: response.getRecords()) {
    var user = User.builder()
      .userId(row.get(0).getLongValue())
      .firstName(row.get(1).getStringValue())
      .lastName(row.get(2).getStringValue())
      .dob(Instant.parse(row.get(3).getStringValue()))
      .build();
    result.add(user);
  }
```

Os valores de exemplo a seguir mostram os valores do campo `formattedRecords` para conjuntos de resultados com diferentes números e aliases de coluna, bem como tipos de dados de coluna.

Se o conjunto de resultados incluir várias linhas, cada uma será representada como um objeto que é um elemento de matriz. Cada coluna no conjunto de resultados se torna uma chave no objeto. Esses nomes de coluna são repetidos para cada linha do conjunto de resultados. Dessa forma, para conjuntos de resultados que compreendem muitas linhas e colunas, talvez seja necessário definir aliases de coluna curtos para não exceder o limite de comprimento da resposta completa.

Este exemplo funciona com o esquema e os dados de amostra a seguir:

```
create table sample_names (id int, name varchar(128));
insert into sample_names values (0, "Jane"), (1, "Mohan"), (2, "Maria"), (3, "Bruce"),
 (4, "Jasmine");
```

```
[{"id":0,"name":"Jane"},{"id":1,"name":"Mohan"},
{"id":2,"name":"Maria"},{"id":3,"name":"Bruce"},{"id":4,"name":"Jasmine"}]
```

Se uma coluna no conjunto de resultados for definida como uma expressão, o texto da expressão se tornará a chave JSON. Por isso, em geral é conveniente definir um alias de coluna descritivo para cada expressão na lista de seleção da consulta. Por exemplo, a consulta a seguir inclui expressões como chamadas de função e operações aritméticas em sua lista de seleção.

```
select count(*), max(id), 4+7 from sample_names;
```

Essas expressões são transmitidas para o conjunto de resultados JSON como chaves.

```
[{"count(*)":5,"max(id)":4,"4+7":11}]
```

Adicionar colunas AS com rótulos descritivos torna as chaves mais simples de interpretar no conjunto de resultados JSON.

```
select count(*) as rows, max(id) as largest_id, 4+7 as addition_result from
 sample_names;
```

Com a consulta SQL revisada, os rótulos de coluna definidos pelas cláusulas AS são usados como nomes de chave.

```
[{"rows":5,"largest_id":4,"addition_result":11}]
```

O valor para cada par de chave-valor na string JSON pode ser uma string entre aspas. A string pode conter caracteres unicode. Se a string contiver sequências de escape ou os caracteres " ou \, esses caracteres serão precedidos por caracteres de escape de barra invertida. Os exemplos de strings JSON a seguir demonstram essas possibilidades. Por exemplo, o resultado `string_with_escape_sequences` contém os caracteres especiais backspace, nova linha, retorno de carro, guia, feed de formulário e \.

```
[{"quoted_string":"hello"}]
[{"unicode_string":"####"}]
[{"string_with_escape_sequences":"\b \n \r \t \f \\ '"}]
```

O valor para cada par de chave-valor na string JSON pode ser uma string entre aspas. O número pode ser um valor inteiro, um valor de ponto flutuante, um valor negativo ou um valor representado como notação exponencial. Os exemplos de strings JSON a seguir demonstram essas possibilidades.

```
[{"integer_value":17}]
[{"float_value":10.0}]
[{"negative_value":-9223372036854775808,"positive_value":9223372036854775807}]
[{"very_small_floating_point_value":4.9E-324,"very_large_floating_point_value":1.79769313486231
```

Valores boolianos e nulos são representados com as palavras-chave especiais `true`, `false` e `null` sem aspas. Os exemplos de strings JSON a seguir demonstram essas possibilidades.

```
[{"boolean_value_1":true,"boolean_value_2":false}]
[{"unknown_value":null}]
```

Se você selecionar um valor de um tipo BLOB, o resultado será representado na string JSON como um valor codificado em base64. Para converter o valor de volta à representação original, você pode usar a função de decodificação apropriada na linguagem da aplicação. Por exemplo, em Java, chame a função `Base64.getDecoder().decode()`. O exemplo de saída a seguir mostra o resultado da seleção de um valor BLOB de `hello world` e do retorno do conjunto de resultados como uma string JSON.

```
[{"blob_column":"aGVsbG8gd29ybGQ="}]
```

O exemplo de Python a seguir mostra como acessar os valores do resultado de uma chamada para a função `execute_statement` do Python. O conjunto de resultados é um valor de string no campo `response['formattedRecords']`. O código transforma a string JSON em uma estrutura de dados chamando a função `json.loads`. Depois, cada linha do conjunto de resultados é um elemento de lista dentro da estrutura de dados e, em cada linha, é possível fazer referência a cada campo do conjunto de resultados pelo nome.

```
import json

result = json.loads(response['formattedRecords'])
print (result[0]["id"])
```

O exemplo de Javascript a seguir mostra como acessar os valores do resultado de uma chamada para a função `executeStatement` do Javascript. O conjunto de resultados é um valor de string no campo `response.formattedRecords`. O código transforma a string JSON em uma estrutura de dados chamando a função `JSON.parse`. Depois, cada linha do conjunto de resultados é um elemento de matriz dentro da estrutura de dados e, em cada linha, é possível fazer referência a cada campo do conjunto de resultados pelo nome.

```
<script>
    const result = JSON.parse(response.formattedRecords);
    document.getElementById("display").innerHTML = result[0].id;
</script>
```

Solução de problemas com a API de dados do Amazon RDS

Use as seções a seguir, intituladas com mensagens de erro comuns, para ajudar na solução de problemas com a API de dados do Amazon RDS (API de dados).

Tópicos

- Transação <transaction_ID> não encontrada
- O pacote para consulta é muito grande
- A resposta do banco de dados excedeu o limite de tamanho
- HttpEndpoint não está habilitado para o cluster <cluster_ID>
- DatabaseErrorException: a transação ainda está executando uma consulta
- Exceção de resultado não aceito
- Declarações múltiplas não são aceitas
- O parâmetro schema não é aceito
- Problemas de conectividade IPv6

Transação <transaction_ID> não encontrada

Nesse caso, o ID da transação especificado em uma chamada de API de dados não foi encontrado. A causa desse problema é anexada à mensagem de erro e é uma das seguintes:

- A transação pode ter expirado.

  Verifique se a chamada da transação é executada três minutos depois da última.

  Também é possível que o ID da transação especificada não tenha sido criado por uma chamada de BeginTransaction. Verifique se a chamada tem um ID de transação válido.
- Uma chamada anterior provocou o encerramento da transação.

  A transação já foi encerrada pela chamada `CommitTransaction` ou `RollbackTransaction`.
- A transação foi interrompida devido a um erro de uma chamada anterior.

  Verifique se suas chamadas anteriores lançaram exceções.

Para obter informações sobre como executar transações, consulte Chamar a API de dados do Amazon RDS.

O pacote para consulta é muito grande

Nesse caso, o conjunto de resultados obtido para uma linha era muito grande. O limite de tamanho da API de dados é de 64 KB por linha no conjunto de resultados obtido pelo banco de dados.

Para resolver esse problema, verifique se cada linha em um conjunto de resultados tem até 64 KB.

A resposta do banco de dados excedeu o limite de tamanho

Nesse caso, o tamanho do conjunto de resultados obtido pelo banco de dados era muito grande. O limite de tamanho da API de dados é de 1 MiB no conjunto de resultados obtido pelo banco de dados.

Para resolver esse problema, garanta que as chamadas para a API de dados exibam até 1 MiB de dados. Se você precisar retornar mais de 1 MiB, poderá usar várias chamadas `ExecuteStatement` com a cláusula `LIMIT` na sua consulta.

Para obter mais informações sobre a cláusula `LIMIT`, consulte SELECT Syntax na documentação do MySQL.

HttpEndpoint não está habilitado para o cluster <cluster_ID>

Verifique as possíveis causas a seguir para esse problema:

- O cluster de banco de dados do Aurora não é compatível com a API de dados. Para ter informações sobre os tipos de clusters de banco de dados compatíveis com a API de dados do RDS, consulte the section called “Disponibilidade de região e versão para a API de dados do Amazon RDS”.
- A API de dados não está habilitada para o cluster de banco de dados do Aurora. Para usar a API de dados com um cluster de banco de dados do Aurora, a API de dados deve estar habilitada para o cluster de banco de dados. Para ter informações sobre como habilitar a API de dados, consulte Habilitar a API de dados do Amazon RDS.
- O cluster de banco de dados foi renomeado depois que a API de dados foi habilitada para ele. Nesse caso, desative a API de dados desse cluster e, depois, habilite-a novamente.
- O ARN especificado não corresponde com precisão ao ARN do cluster. Verifique se o ARN retornado de outra fonte ou criado pela lógica do programa corresponde exatamente ao ARN do cluster. Por exemplo, verifique se o ARN usado tem a letra correta para todos os caracteres alfabéticos.

DatabaseErrorException: a transação ainda está executando uma consulta

Se a aplicação enviar uma solicitação com um ID de transação e essa transação estiver processando outra solicitação no momento, a API de dados retornará esse erro à aplicação imediatamente. Essa condição poderá surgir se a aplicação fizer solicitações assíncronas, usando um mecanismo como "promessas" em JavaScript.

Para resolver esse problema, aguarde até que a solicitação anterior seja concluída e repita a solicitação. Você pode continuar tentando novamente até que o erro não ocorra mais ou que a aplicação receba algum tipo diferente de erro.

Essa condição pode ocorrer com a API de dados do Aurora Serverless v2 e instâncias provisionadas. Na API de dados do Aurora Serverless v1, as solicitações subsequentes para o mesmo ID de transação aguardam automaticamente a conclusão da solicitação anterior. No entanto, esse comportamento mais antigo poderia encontrar tempos limite devido ao fato de a solicitação anterior demorar muito. Se você tiver uma aplicação antiga da API de dados que faz solicitações simultâneas, modifique sua lógica de tratamento de exceções para considerar esse novo tipo de erro.

Exceção de resultado não aceito

A API de dados não comporta todos os tipos de dados. Esse erro ocorre quando você executa uma consulta que exibe um tipo de dados não aceito.

Para contornar esse problema, converta o tipo de dados não aceito em `TEXT`. Por exemplo:

```
SELECT custom_type::TEXT FROM my_table;
-- OR
SELECT CAST(custom_type AS TEXT) FROM my_table;
```

Declarações múltiplas não são aceitas

Na API de dados para o Aurora Sem Servidor v2 e em clusters provisionados, não é possível usar declarações múltiplas. A tentativa de executar várias declarações em uma única chamada de API gera esse erro.

Para executar várias declarações, use chamadas de API `ExecuteStatement` separadas ou use a API `BatchExecuteStatement` para processamento em lote.

O parâmetro schema não é aceito

O Aurora Sem Servidor v1 ignora silenciosamente o parâmetro schema. No entanto, o Aurora Sem Servidor v2 e os clusters provisionados rejeitam explicitamente as chamadas de API que incluem o parâmetro schema.

Para resolver esse problema, remova o parâmetro schema de todas as chamadas à API de dados ao usar o Aurora Sem Servidor v2 ou clusters provisionados.

Problemas de conectividade IPv6

Se você tiver problemas ao se conectar à API de dados usando endpoints IPv6, confira as seguintes possíveis causas:

- A rede não aceita IPv6: verifique se sua infraestrutura de rede comporta IPv6 e se o roteamento IPv6 está configurado corretamente.
- Problemas de resolução DNS: verifique se seu resolvedor de DNS consegue resolver registros AAAA para os endpoints de pilha dupla (por exemplo, `rds-data.us-east-1.api.aws`).
- Configuração dos grupos de segurança: atualize as regras do grupo de segurança para permitir o tráfego IPv6 na porta 443 (HTTPS). Adicione regras para blocos CIDR IPv6 (por exemplo, `::/0` para todos os endereços IPv6).
- Configuração da ACL de rede: certifique-se de que as ACLs de rede permitam tráfego IPv6 nas portas necessárias.
- Compatibilidade da biblioteca de cliente: verifique se suas bibliotecas de cliente HTTP e AWS SDKs aceitam a conectividade IPv6 e de pilha dupla.
- Configuração do endpoint da VPC: se estiver usando o PrivateLink, garanta que seu endpoint da VPC esteja configurado para aceitar IPv6 e que as sub-redes associadas tenham blocos CIDR IPv6 atribuídos.

Para solucionar problemas de conectividade IPv6:

1. Teste a conectividade usando os endpoints somente IPv4 (`.amazonaws.com`) para verificar se o problema é específico do IPv6.
2. Use ferramentas de diagnóstico de rede para verificar a conectividade IPv6 com os endpoints de pilha dupla.
3. Confira se há erros de autenticação ou autorização nos logs do CloudTrail ao usar endpoints IPv6.

4. Verifique se sua aplicação está configurada corretamente para usar os novos URLs de endpoint de pilha dupla.

Registrar em log chamadas da API de dados do Amazon RDS com o AWS CloudTrail

A API de dados do RDS (API de dados) é integrada ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS na API de dados. O CloudTrail captura todas as chamadas de API para a API de dados como eventos, inclusive as chamadas do console do Amazon RDS e de chamadas do código para operações da API de dados. Se você criar uma trilha, será possível ativar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3, incluindo eventos da API Data. Ao usar os dados coletados pelo CloudTrail, é possível determinar várias informações. Essas informações incluem a solicitação feita à API Data, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita, além de detalhes adicionais.

Para saber mais sobre o CloudTrail, consulte o Manual do usuário do AWS CloudTrail.

Trabalhar com informações da API Data no CloudTrail

O CloudTrail é habilitado em sua conta da AWS quando ela é criada. Quando uma atividade compatível (eventos de gerenciamento) ocorre na API de dados, ela é registrada em um evento do CloudTrail além de outros eventos de serviços da AWS no Histórico de eventos. É possível visualizar, pesquisar e baixar eventos de gerenciamento recentes na conta da AWS. Para obter mais informações, consulte Trabalhar com histórico de eventos do CloudTrail no Guia do usuário do AWS CloudTrail.

Para ter um registro contínuo de eventos na sua conta da AWS, incluindo os eventos da API Data, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando uma trilha é criada no console, a mesma é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões da AWS na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, você pode configurar outros produtos da AWS para analisar mais profundamente e agir sobre os dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte os seguintes tópicos no Manual do usuário do AWS CloudTrail:

- Visão geral da criação de uma trilha

- Serviços e integrações compatíveis com o CloudTrail
- Configuração notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e Receber arquivos de log do CloudTrail de várias contas

Todas as operações da API de dados são registradas em log pelo CloudTrail e documentadas na Referência de API do serviço de dados do Amazon RDS. Por exemplo, as chamadas para as operações `BatchExecuteStatement`, `BeginTransaction`, `CommitTransaction` e `ExecuteStatement` geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário da raiz ou do .
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para saber mais, consulte Elemento userIdentity do CloudTrail.

Incluir e excluir eventos da API de dados de uma trilha do AWS CloudTrail

A maioria dos usuários da API de dados depende dos eventos em uma trilha do AWS CloudTrail para fornecer um registro das operações da API de dados. Os dados do evento não revelam o nome do banco de dados, o nome do esquema ou as instruções SQL nas solicitações à API de dados. No entanto, saber qual usuário fez um tipo de chamada em um cluster de banco de dados específico em determinado momento pode ajudar a detectar padrões de acesso anômalos.

Incluir eventos da API de dados em uma trilha do AWS CloudTrail

Para o Aurora PostgreSQL Sem Servidor v2 e bancos de dados provisionados, as seguintes operações da API de dados são registradas em log como eventos de dados no AWS CloudTrail como eventos de dados. Eventos de dados são operações de API de plano de dados de alto volume que o CloudTrail não registra por padrão. Há cobranças adicionais para eventos de dados. Para obter mais informações sobre preços do CloudTrail, consulte Preços do AWS CloudTrail.

- BatchExecuteStatement

- BeginTransaction
- CommitTransaction
- ExecuteStatement
- RollbackTransaction

É possível usar o console do CloudTrail, AWS CLI, ou as operações da API do CloudTrail para registrar essas operações da API de dados. No console do CloudTrail, selecione API de dados do RDS: cluster de banco de dados para o tipo de evento Dados. Para ter mais informações, consulte Logging data events with theConsole de gerenciamento da AWS, no Manual do usuário do AWS CloudTrail.

Usando a AWS CLI, execute o comando `aws cloudtrail put-event-selectors` para registrar essas operações da API de dados para a trilha. Para registrar todos os eventos da API de dados em clusters de banco de dados, especifique `AWS::RDS::DBCluster` para o tipo de recurso. O exemplo a seguir registra todos os eventos da API de dados em clusters de banco de dados. Para ter mais informações, consulte Logging data events with theAWS Command Line Interface, no Manual do usuário do AWS CloudTrail.

```
aws cloudtrail put-event-selectors --trail-name trail_name --advanced-event-selectors \
'{
   "Name": "RDS Data API Selector",
   "FieldSelectors": [
      {
         "Field": "eventCategory",
         "Equals": [
            "Data"
         ]
      },
      {
         "Field": "resources.type",
         "Equals": [
            "AWS::RDS::DBCluster"
         ]
      }
   ]
}'
```

É possível configurar seletores de eventos avançados para filtrar adicionalmente os campos `readOnly`, `eventName`, e `resources.ARN`. Para ter mais informações sobre esses campos, consulte AdvancedFieldSelector.

Excluir eventos da API de dados de uma trilha do AWS CloudTrail (somente o Aurora Serverless v1)

Para o Aurora Serverless v1, os eventos da API de dados são eventos de gerenciamento. Por padrão, todos os eventos da API de dados são incluídos em uma trilha do AWS CloudTrail. No entanto, como a API de dados pode gerar um grande número de eventos, é possível excluir esses eventos da trilha do CloudTrail. A configuração Excluir eventos da API de dados do Amazon RDS exclui todos os eventos da API de dados da trilha. Não é possível excluir eventos específicos da API de dados.

Para excluir eventos da API de da dados de uma trilha, faça o seguinte:

- No console do CloudTrail, escolha a opção Exclude Amazon RDS Data API events (Excluir eventos da API de dados do Amazon RDS) quando você criar uma trilha ou atualizar uma trilha.
- Na API do CloudTrail, use a operação PutEventSelectors. Se estiver usando seletores de eventos avançados, você poderá excluir eventos da API de dados definindo o campo `eventSource` diferente de `rdsdata.amazonaws.com`. Se estiver usando seletores de eventos básicos, você poderá excluir eventos da API de dados definindo o valor do atributo `ExcludeManagementEventSources` como `rdsdata.amazonaws.com`. Consulte mais informações em Logging data events with the AWS Command Line Interface, no Guia do usuário do AWS CloudTrail.

> **Warning**
>
> Excluir eventos da API de dados de um log do CloudTrail pode ocultar ações da API de dados. Tenha cuidado ao conceder aos principais a permissão `cloudtrail:PutEventSelectors` que é necessária para executar essa operação.

É possível desativar essa exclusão a qualquer momento alterando a configuração do console ou os seletores de eventos de uma trilha. Então, a trilha começará a gravar os eventos da API de dados . No entanto, não pode recuperar eventos da API de dados que ocorreram enquanto a exclusão estava em vigor.

Ao excluir eventos da API de dados usando o console ou a API, a operação `PutEventSelectors` da API do CloudTrail também é registrada nos logs do CloudTrail. Se os eventos da API de dados não aparecerem nos logs do CloudTrail, procure um evento `PutEventSelectors` com o atributo `ExcludeManagementEventSources` definido como `rdsdata.amazonaws.com`.

Para obter mais informações, consulte Logging management events for trails (Registro de eventos de dados para trilhas) no Manual do usuário do AWS CloudTrail.

Noções básicas sobre as entradas do arquivo de log da API Data

Uma trilha é uma configuração que permite a entrega de eventos como registros de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública, portanto, não são exibidos em uma ordem específica.

Aurora PostgreSQL Sem Servidor v2 e provisionado

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a operação `ExecuteStatement` do Aurora PostgreSQL Sem Servidor v2 e dos bancos de dados provisionados. Para esses bancos de dados, todos os eventos da API de dados são eventos de dados em que a origem do evento é rdsdataapi.amazonaws.com e o tipo de evento é Rds Data Service.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdataapi.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
```

```
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "Rds Data Service",
    "recipientAccountId": "123456789012"
}
```

Aurora Serverless v1

O exemplo a seguir mostra como o exemplo anterior da entrada de log do CloudTrail aparece para o Aurora Serverless v1. Para o Aurora Serverless v1, todos os eventos são eventos de gerenciamento em que a origem do evento é rdsdata.amazonaws.com e o tipo de evento é AwsApiCall.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdata.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
```

```
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Monitorar consultas da API de dados do RDS com o Insights de Performance

Se o cluster do Aurora estiver executando instâncias do Aurora Serverless v2 ou provisionadas, você poderá usar o Insights de Performance com a API de dados do RDS.

Consulte mais informações sobre como usar o Insights de Performance com o Aurora em Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

Como as consultas da API de dados do RDS são representadas no Insights de Performance

Com a API de dados, o cluster do Aurora processa consultas com base nas chamadas da API de dados que você envia da aplicação. A API de dados também executa algumas instruções SQL como parte de seu próprio funcionamento interno, como cancelar consultas que excedam o tempo limite. Os dois tipos de operações SQL são mostrados em estatísticas e gráficos do Insights de Performance.

- Para consultas da API de dados que você envia para um cluster do Aurora, o campo Host no painel do PI é marcado como API de dados do RDS. Para o Aurora PostgreSQL, o campo application_name tem o valor `rds-data-api`. Procure esses rótulos ao analisar a carga do banco de dados usando Principais hosts ou Principais aplicações como uma dimensão.

- Todas as consultas internas que a API de dados executa para gerenciar aspectos do banco de dados, como o grupo de conexões e o tempo limite de consulta, são anotadas com o prefixo API de dados do RDS. Exemplo: `/* RDS Data API */ select * from my_table;` procura esses prefixos ao analisar a carga do banco de dados por Principal SQL como uma dimensão. As instruções são anotadas com um comentário de SQL da `/* RDS Data API */`.

Como usar o editor de consultas do Aurora

O editor de consultas do Aurora permite que você execute instruções SQL no cluster de banco de dados do Aurora por meio do Console de gerenciamento da AWS. É possível executar consultas SQL, instruções de manipulação de dados (DML) e instruções de definição de dados (DDL). O uso da interface do console permite que você realize a manutenção do banco de dados, gere relatórios e conduza experimentos de SQL. Você pode evitar definir a configuração de rede para se conectar ao cluster de banco de dados por um sistema cliente separado, como uma instância do EC2 ou um notebook.

O editor de consultas exige um cluster de banco de dados do Aurora com a API de dados do RDS (API de dados) habilitada. Para ter informações sobre clusters de banco de dados que são compatíveis com a API de dados e como habilitá-la, consulte Usar a API de dados do Amazon RDS. O SQL que você pode executar está sujeito às limitações da API de dados. Para obter mais informações, consulte the section called “Limitações”.

Disponibilidade do editor de consultas

O editor de consultas está disponível para clusters de banco de dados do Aurora usando versões dos mecanismos do Aurora MySQL e do Aurora PostgreSQL compatíveis com a API de dados e as Regiões da AWS onde a API de dados está disponível. Para obter mais informações, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a API de dados do RDS.

Autorizar acesso ao editor de consultas

Um usuário deve estar autorizado a executar consultas no editor. Você pode autorizar um usuário a executar consultas no editor de consultas adicionando a política `AmazonRDSDataFullAccess`, uma política predefinida do AWS Identity and Access Management (IAM), a esse usuário.

> **Note**
>
> Para criar o usuário do IAM, use o mesmo nome do usuário e senha utilizados para o usuário do banco de dados, como o nome de usuário administrativo e a senha. Para obter mais informações, consulte Criar um usuário do IAM em sua Conta da AWS no Guia do usuário do AWS Identity and Access Management.

Também crie uma política do IAM que conceda acesso ao editor de consultas. Depois de criar a política, adicione-a a cada usuário que exija acesso ao editor de consultas.

A política a seguir oferece as permissões obrigatórias mínimas para um usuário acessar o editor de consultas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "QueryEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutResourcePolicy",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "QueryEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "tag:GetResources",
                "secretsmanager:CreateSecret",
                "secretsmanager:ListSecrets",
                "dbqms:CreateFavoriteQuery",
                "dbqms:DescribeFavoriteQueries",
                "dbqms:UpdateFavoriteQuery",
                "dbqms:DeleteFavoriteQueries",
                "dbqms:GetQueryString",
                "dbqms:CreateQueryHistory",
                "dbqms:UpdateQueryHistory",
                "dbqms:DeleteQueryHistory",
                "dbqms:DescribeQueryHistory",
                "rds-data:BatchExecuteStatement",
```

```
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

Para obter mais informações sobre como criar uma política do IAM, consulte Criar políticas do IAM no Manual do usuário do AWS Identity and Access Management.

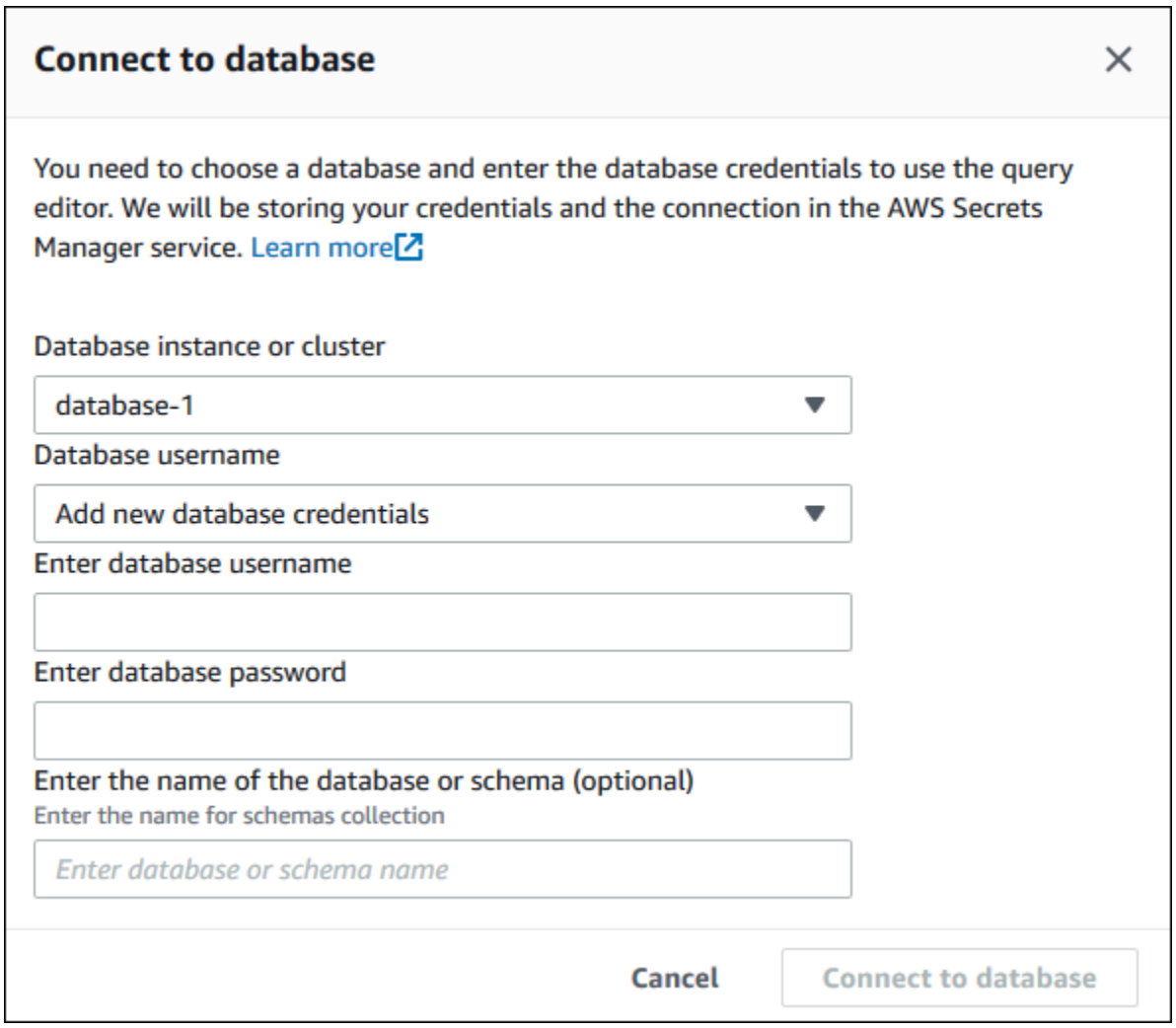
Para obter informações sobre como adicionar uma política do IAM a um usuário, consulte Adicionar e remover permissões de identidade do IAM no Manual do usuário do AWS Identity and Access Management.

Executar consultas no editor de consultas

Execute declarações SQL em um cluster de banco de dados do Aurora no editor de consultas. O SQL que você pode executar está sujeito às limitações da API de dados. Para obter mais informações, consulte the section called “Limitações”.

Para executar uma consulta no editor

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do Console de gerenciamento da AWS, selecione a Região da AWS na qual você criou os clusters de banco de dados do Aurora que deseja consultar.
3. No painel de navegação, escolha Bancos de dados.
4. Selecione o cluster de banco de dados do Aurora em que você deseja executar consultas SQL.
5. Em Actions (Ações), escolha Query (Consulta). Caso ainda não tenha se conectado ao banco de dados, a página Connect to database (Conectar ao banco de dados) é aberta.



6. Insira as seguintes informações:

   a. Em Cluster ou instância de banco de dados, selecione o cluster de banco de dados do Aurora no qual você deseja executar consultas SQL.

   b. Em Database username (Nome de usuário do banco de dados), escolha o nome do usuário do banco de dados para conexão ou escolha Add new database credentials (Adicionar novas credenciais do banco de dados). Se você selecionar Add new database credentials (Adicionar novas credenciais do banco de dados), insira o nome do usuário para as novas credenciais do banco de dados em Enter database username (Inserir nome de usuário do banco de dados).

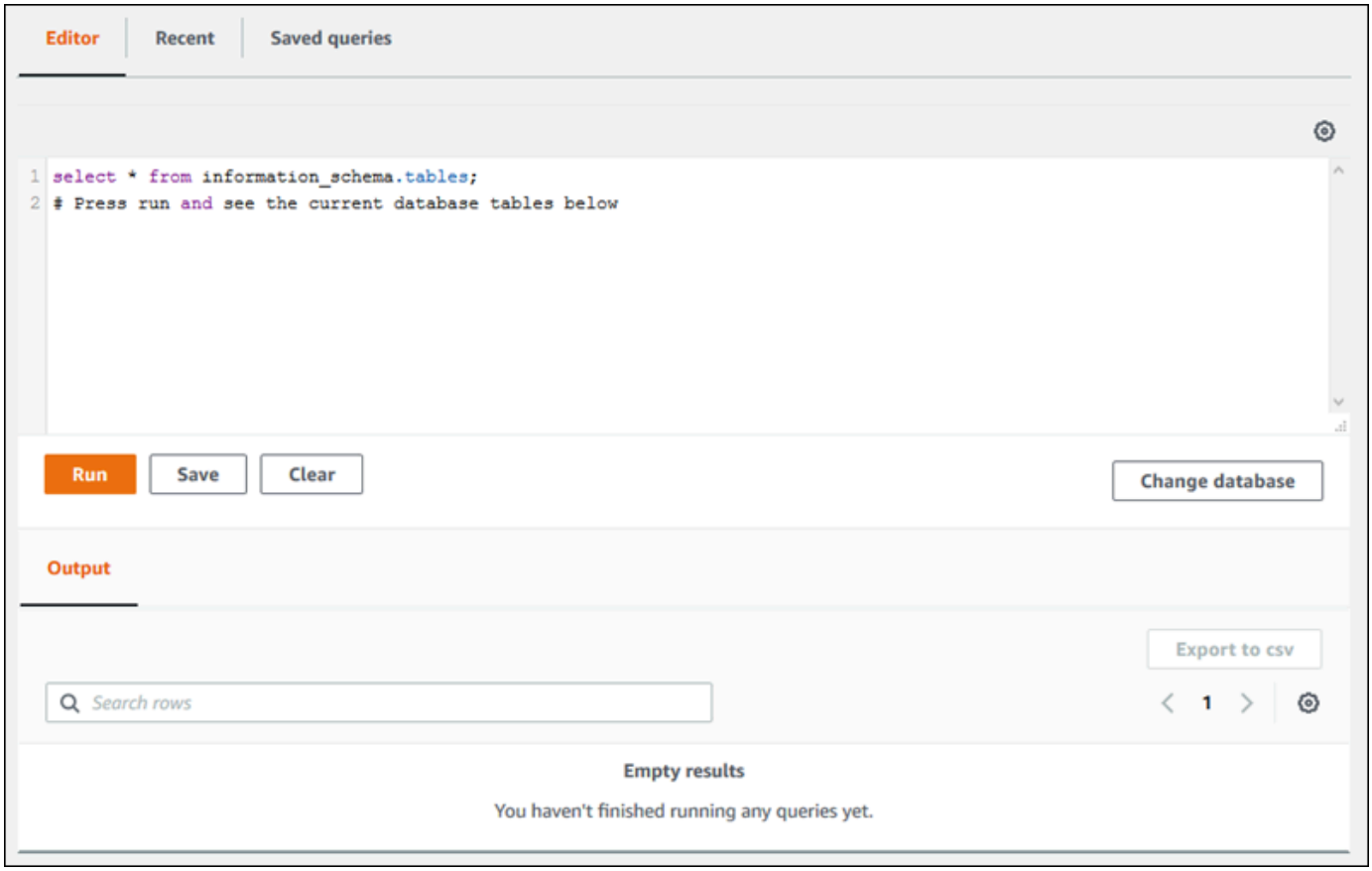
   c. Em Enter database password (Inserir senha do banco de dados), insira a senha do usuário do banco de dados escolhido.

   d. Na última caixa, digite o nome do esquema ou banco de dados que você deseja usar no cluster de banco de dados Aurora.

e. Escolha Connect to database (Conectar ao banco de dados).

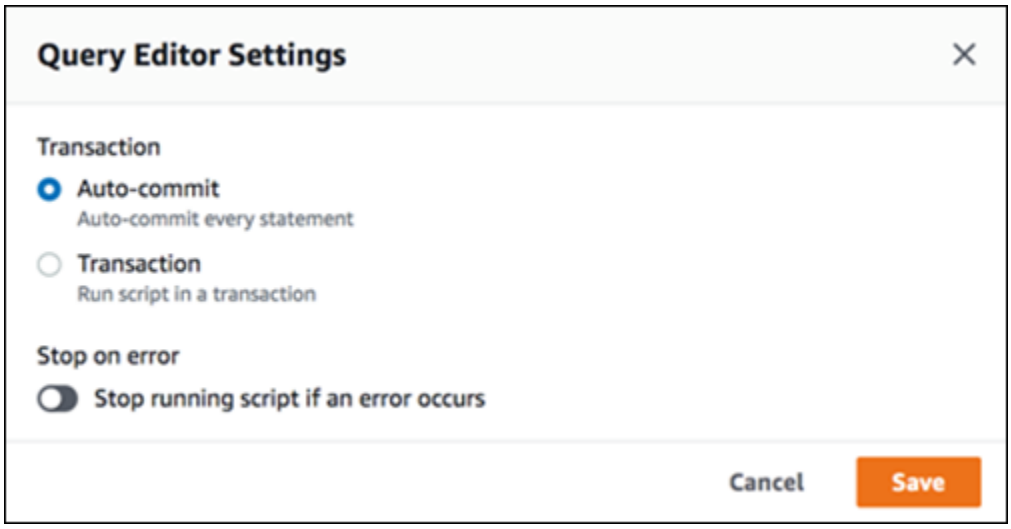
> **Note**
>
> Caso a conexão seja bem-sucedida, as informações de conexão e autenticação são armazenadas no AWS Secrets Manager. Não é necessário digitar novamente as informações da conexão.

7. No editor de consultas, insira a consulta SQL que você deseja executar no banco de dados.



Cada instrução SQL pode ser confirmada automaticamente ou você pode executar instruções SQL em um script como parte de uma transação. Para controlar esse comportamento, escolha o ícone de engrenagem acima da janela de consulta.

A janela Query Editor Settings (Configurações do editor de consultas) é exibida.



Se você selecionar Auto-commit (Confirmar automaticamente), cada instrução SQL é confirmada automaticamente. Se você selecionar Transação, poderá executar um grupo de instruções em um script. As instruções são confirmadas automaticamente no final do script, a menos que explicitamente confirmadas ou revertidas antes disso. Além disso, você poderá optar por interromper a execução de um script se ocorrer um erro habilitando a opção Stop on error (Interromper em caso de erro).

> **Note**
>
> Em um grupo de instruções, as instruções da linguagem de definição de dados (DDL) podem fazer com que as instruções da linguagem de manipulação de dados (DML) sejam confirmadas. Você também pode incluir instruções `COMMIT` e `ROLLBACK` em um grupo de instruções em um script.

Após fazer suas escolhas na janela Query Editor Settings (Configurações do editor de consultas), escolha Save (Salvar).

8. Escolha Run (Executar) ou pressione Ctrl+Enter e o editor de consultas exibirá os resultados de sua consulta.

   Depois de executar a consulta, salve-a em Saved queries (Consultas salvas) selecionando Save (Salvar).

   Exporte os resultados da consulta para o formato de planilha escolhendo Export to csv (Exportar para csv).

Você pode encontrar, editar e executar consultas anteriores novamente. Para isso, escolha a guia Recent (Recente) ou a guia Saved queries (Consultas salvas), escolha o texto da consulta e escolha Run (Executar).

Para alterar o banco de dados, escolha Change database (Alterar banco de dados).

Referência da API DBQMS (Database Query Metadata Service)

O Database Query Metadata Service (`dbqms`) é um serviço somente interno. Ele fornece suas consultas recentes e salvas para o editor de consultas no Console de gerenciamento da AWS para vários produtos da AWS, inclusive o Amazon RDS .

Ações compatíveis do DBQMS

- CreateFavoriteQuery
- CreateQueryHistory
- CreateTab
- DeleteFavoriteQueries

- DeleteQueryHistory
- DeleteTab
- DescribeFavoriteQueries
- DescribeQueryHistory
- DescribeTabs
- GetQueryString
- UpdateFavoriteQuery
- UpdateQueryHistory
- UpdateTab

CreateFavoriteQuery

Salva uma nova consulta como favorita. Cada usuário do pode criar até 1.000 consultas salvas. Este limite está sujeito a alterações no futuro.

CreateQueryHistory

Salve uma nova entrada de histórico de consultas.

CreateTab

Salve uma nova guia de consulta. Cada usuário do pode criar até 10 guias de consulta.

DeleteFavoriteQueries

Exclua uma ou mais consultas salvas.

DeleteQueryHistory

Exclua entradas do histórico de consultas.

DeleteTab

Exclua entradas da guia de consulta.

DescribeFavoriteQueries

Liste as consultas salvas criadas por um usuário em determinada conta.

DescribeQueryHistory

Liste entradas do histórico de consultas.

DescribeTabs

Liste guias de consulta criadas por um usuário em determinada conta.

GetQueryString

Recupere o texto completo da consulta a partir de um ID de consulta.

UpdateFavoriteQuery

Atualize a string de consulta, a descrição, o nome ou a data de validade.

UpdateQueryHistory

Atualize o status do histórico de consultas.

UpdateTab

Atualize o nome da guia de consulta e a string de consulta.

Usar o machine learning do Amazon Aurora

Ao usar o machine learning do Amazon Aurora, é possível integrar o cluster de banco de dados do Aurora a um dos serviços de machine learning da AWS a seguir, dependendo das necessidades. Eles oferecem compatibilidade com casos de uso de machine learning específicos.

Amazon Bedrock

O Amazon Bedrock é um serviço totalmente gerenciado que disponibiliza os principais modelos de base de empresas de IA por meio de uma API, junto com ferramentas para desenvolvedores para ajudar a criar e escalar aplicações de IA generativa. Com o Amazon Bedrock, você paga para executar inferências em qualquer um dos modelos de base de terceiros. O preço é baseado no volume de tokens de entrada e tokens de saída e no fato de você ter comprado o throughput provisionado para o modelo. Para obter mais informações, consulte O que é a Amazon Bedrock? no Guia do Usuário Amazon Bedrock.

Amazon Comprehend

O Amazon Comprehend é um serviço gerenciado de processamento de linguagem natural (PLN) utilizado para extrair insights de documentos. Com o Amazon Comprehend, você pode deduzir sentimentos com base no conteúdo dos documentos, analisando entidades, frases-chave, linguagem e outros recursos. Para saber mais, consulte What is Amazon Comprehend? (O que é o Amazon Comprehend?) no Amazon Comprehend Developer Guide (Guia do desenvolvedor do Amazon Comprehend).

SageMaker AI

Amazon SageMaker AI é um serviço de machine learning totalmente gerenciado. Cientistas de dados e desenvolvedores usam o Amazon SageMaker AI para criar, treinar e testar modelos de machine learning para uma série de tarefas de inferência, como detecção de fraudes e recomendação de produtos. Quando um modelo de machine learning está pronto para uso na produção, ele pode ser implantado no ambiente hospedado do Amazon SageMaker AI. Para obter informações, consulte O que é Amazon SageMaker AI? no Guia do desenvolvedor do Amazon SageMaker AI.

Usar o Amazon Comprehend com seu cluster de banco de dados Aurora exige menos configuração preliminar do que usar o SageMaker AI. Se você não conhece o machine learning da AWS e o machine learning do Aurora, recomendamos que comece examinando o Amazon Comprehend.

Tópicos

- Usar o machine learning do Amazon Aurora com o Aurora MySQL
- Usar machine learning do Amazon Aurora com o Aurora PostgreSQL

Usar o machine learning do Amazon Aurora com o Aurora MySQL

Ao usar o machine learning do Amazon Aurora com o cluster de banco de dados do Aurora MySQL, é possível usar o Amazon Bedrock, o Amazon Comprehend, ou o Amazon SageMaker AI, dependendo das suas necessidades. Eles oferecem compatibilidade com casos de uso de machine learning diferentes.

Sumário

- Requisitos para usar o machine learning do Aurora com o Aurora MySQL
- Disponibilidade de regiões e versões
- Recursos compatíveis e limitações do machine learning do Aurora com o MySQL
- Configurar o cluster de banco de dados do Aurora MySQL para usar machine learning do Aurora
  - Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Bedrock
  - Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Comprehend
  - Configuração do cluster de banco de dados do Aurora MySQL para usar o SageMaker AI
    - Configuração do cluster de banco de dados do Aurora MySQL para usar o Amazon S3 para SageMaker AI (opcional)
  - Conceder aos usuários de banco de dados acesso ao machine learning do Aurora
    - Conceder acesso às funções do Amazon Bedrock
    - Conceder acesso às funções do Amazon Comprehend
    - Concessão de acesso às funções do SageMaker AI
- Usar o Amazon Bedrock com o cluster de banco de dados do Aurora MySQL
- Usar o Amazon Comprehend com seu cluster de banco de dados do Aurora MySQL
- Uso do SageMaker AI com o cluster de banco de dados do Aurora MySQL
  - Requisito de conjunto de caracteres para funções do SageMaker AI que retornam strings
  - Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado)
- Considerações de performance para usar o machine learning do Aurora com o Aurora MySQL
  - Modelo e prompt

  - Cache de consultas
  - Otimização em lotes para chamadas de função do machine learning do Aurora
- Monitorar o machine learning do Aurora

Requisitos para usar o machine learning do Aurora com o Aurora MySQL

Os serviços do AWS Machine Learning são serviços gerenciados que são configurados e executados em seus próprios ambientes de produção. O machine learning do Aurora é compatível com a integração ao Amazon Bedrock, ao Amazon Comprehend e ao SageMaker AI. Antes de tentar configurar seu cluster de banco de dados Aurora MySQL para usar o machine learning do Aurora, entenda os requisitos e pré-requisitos a seguir.

- Os serviços de machine learning devem ser executados na mesma Região da AWS do cluster de banco de dados do Aurora MySQL. Não é possível usar os serviços de machine learning de um cluster de banco de dados do Aurora MySQL em uma região diferente.
- Se o cluster de banco de dados do Aurora MySQL estiver em uma nuvem pública virtual (VPC) do serviço Amazon Bedrock, Amazon Comprehend ou SageMaker AI, o grupo de segurança da VPC precisará permitir conexões de saída com o serviço de machine learning de destino do Aurora. Para obter mais informações, consulte Controlar o tráfego para seus recursos da AWS usando grupos de segurança no Guia do usuário da Amazon VPC.
- Você pode fazer upgrade de um cluster do Aurora que executa uma versão inferior do Aurora MySQL para uma versão superior para usar o machine learning do Aurora com esse cluster. Para obter mais informações, consulte Atualizações do mecanismo de banco de dados Amazon Aurora MySQL.
- O cluster de banco de dados do Aurora MySQL deve usar um grupo de parâmetros de cluster de banco de dados personalizado. Ao final do processo de configuração de cada serviço de machine learning do Aurora que você deseja usar, adicione o nome do recurso da Amazon (ARN) do perfil do IAM associado que foi criado para o serviço. Recomendamos criar um grupo de parâmetros de cluster de banco de dados personalizado para seu Aurora MySQL com antecedência e configurar seu cluster de banco de dados Aurora MySQL para usá-lo de forma que esteja pronto para ser modificado no final do processo de configuração.
- Para o SageMaker AI:
  - Os componentes de machine learning que você deseja usar para inferências devem estar configurados e prontos para uso. Durante o processo de configuração do cluster de banco de dados do Aurora MySQL, é necessário ter o ARN do endpoint do SageMaker AI disponível.

Os cientistas de dados de sua equipe provavelmente estão mais aptos a trabalhar com o SageMaker AI para preparar os modelos e realizar outras tarefas desse tipo. Para começar a usar o Amazon SageMaker AI, consulte Comece a usar o Amazon SageMaker AI. Para obter mais informações sobre inferências e endpoints, consulte Real-time Inference.

- Para usar o SageMaker AI com seus próprios dados de treinamento, é necessário configurar um bucket do Amazon S3 como parte da configuração do Aurora MySQL para o machine learning do Aurora. Para fazer isso, siga o mesmo processo geral de configuração da integração com o SageMaker AI. Para obter um resumo desse processo de configuração opcional, consulte Configuração do cluster de banco de dados do Aurora MySQL para usar o Amazon S3 para SageMaker AI (opcional).

- Para bancos de dados globais do Aurora, configure os serviços de machine learning do Aurora que deseja usar em todas as Regiões da AWS que compõem seu banco de dados global do Aurora. Por exemplo, se quiser usar o machine learning do Aurora com o SageMaker AI para o banco de dados global do Aurora, faça o seguinte para cada cluster de banco de dados do Aurora MySQL em cada Região da AWS:
  - Configure os serviços do Amazon SageMaker AI com os mesmos modelos de treinamento e endpoints do SageMaker AI. Eles também devem usar os mesmos nomes.
  - Crie os perfis do IAM conforme detalhado em Configurar o cluster de banco de dados do Aurora MySQL para usar machine learning do Aurora.
  - Adicione o ARN do perfil do IAM ao grupo de parâmetros do cluster de banco de dados personalizado para cada cluster de banco de dados Aurora MySQL em cada Região da AWS.

  Essas tarefas exigem que o machine learning do Aurora esteja disponível para sua versão do Aurora MySQL em todas as Regiões da AWS que compõem seu banco de dados global do Aurora.

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS.

- Para obter informações sobre a disponibilidade de versões e regiões do Amazon Comprehend e do Amazon SageMaker AI com o Aurora MySQL, consulte Machine learning do Aurora com o MySQL.
- O Amazon Bedrock é compatível somente com o Aurora MySQL versão 3.06 e posterior.

Consulte informações sobre a disponibilidade de regiões do Amazon Bedrock em Model support by Região da AWS no Guia do usuário do Amazon Bedrock.

Recursos compatíveis e limitações do machine learning do Aurora com o MySQL

Ao usar o Aurora MySQL com o machine learning do Aurora, as seguintes limitações se aplicam:

- A extensão de machine learning do Aurora não é compatível com interfaces de vetor.
- As integrações de machine learning do Aurora não são compatíveis quando usadas em um gatilho.
- As funções de machine learning do Aurora não são compatíveis com a replicação de log binário (binlog).
  - A configuração `--binlog-format=STATEMENT` lança uma exceção para chamadas a funções do machine learning do Aurora.
  - As funções do machine learning do Aurora não determinísticas e funções armazenadas não determinísticas não são compatíveis com o formato de binlog.

  Consulte mais informações em Binary Logging Formats na documentação do MySQL.
- Não há suporte para funções armazenadas que chamam tabelas com colunas sempre geradas. Isso se aplica a qualquer função armazenada do Aurora MySQL. Para saber mais sobre esse tipo de coluna, consulte CREATE TABLE and Generated Columns (CREATE TABLE e colunas geradas) na documentação do MySQL.
- As funções do Amazon Bedrock não são compatíveis com `RETURNS JSON`. É possível usar `CONVERT` ou `CAST` para converter de `TEXT` para `JSON`, se necessário.
- O Amazon Bedrock não é compatível com solicitações em lote.
- O Aurora MySQL é compatível com qualquer endpoint do SageMaker AI que leia e grave no formado de valores separados por vírgulas (CSV) por um `ContentType` de `text/csv`. Esse formato é aceito pelos seguintes algoritmos integrados do SageMaker AI:
  - Aprendizagem linear
  - Random Cut Forest
  - XGBoost

  Para saber mais sobre esses algoritmos, consulte Selecionar um algoritmo no Guia do desenvolvedor do Amazon SageMaker AI.

Configurar o cluster de banco de dados do Aurora MySQL para usar machine learning do Aurora

Nos tópicos a seguir, você pode encontrar procedimentos de configuração separados para cada um desses serviços de machine learning do Aurora.

Tópicos

- Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Bedrock
- Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Comprehend
- Configuração do cluster de banco de dados do Aurora MySQL para usar o SageMaker AI
  - Configuração do cluster de banco de dados do Aurora MySQL para usar o Amazon S3 para SageMaker AI (opcional)
- Conceder aos usuários de banco de dados acesso ao machine learning do Aurora
  - Conceder acesso às funções do Amazon Bedrock
  - Conceder acesso às funções do Amazon Comprehend
  - Concessão de acesso às funções do SageMaker AI

Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Bedrock

O machine learning do Aurora Machine depende de políticas e perfis do AWS Identity and Access Management (IAM) para permitir que o cluster de banco de dados do Aurora MySQL acesse e use os serviços do Amazon Bedrock. Os procedimentos a seguir criam uma política e um perfil de permissão do IAM para que o cluster de banco de dados possa se integrar ao Amazon Bedrock.

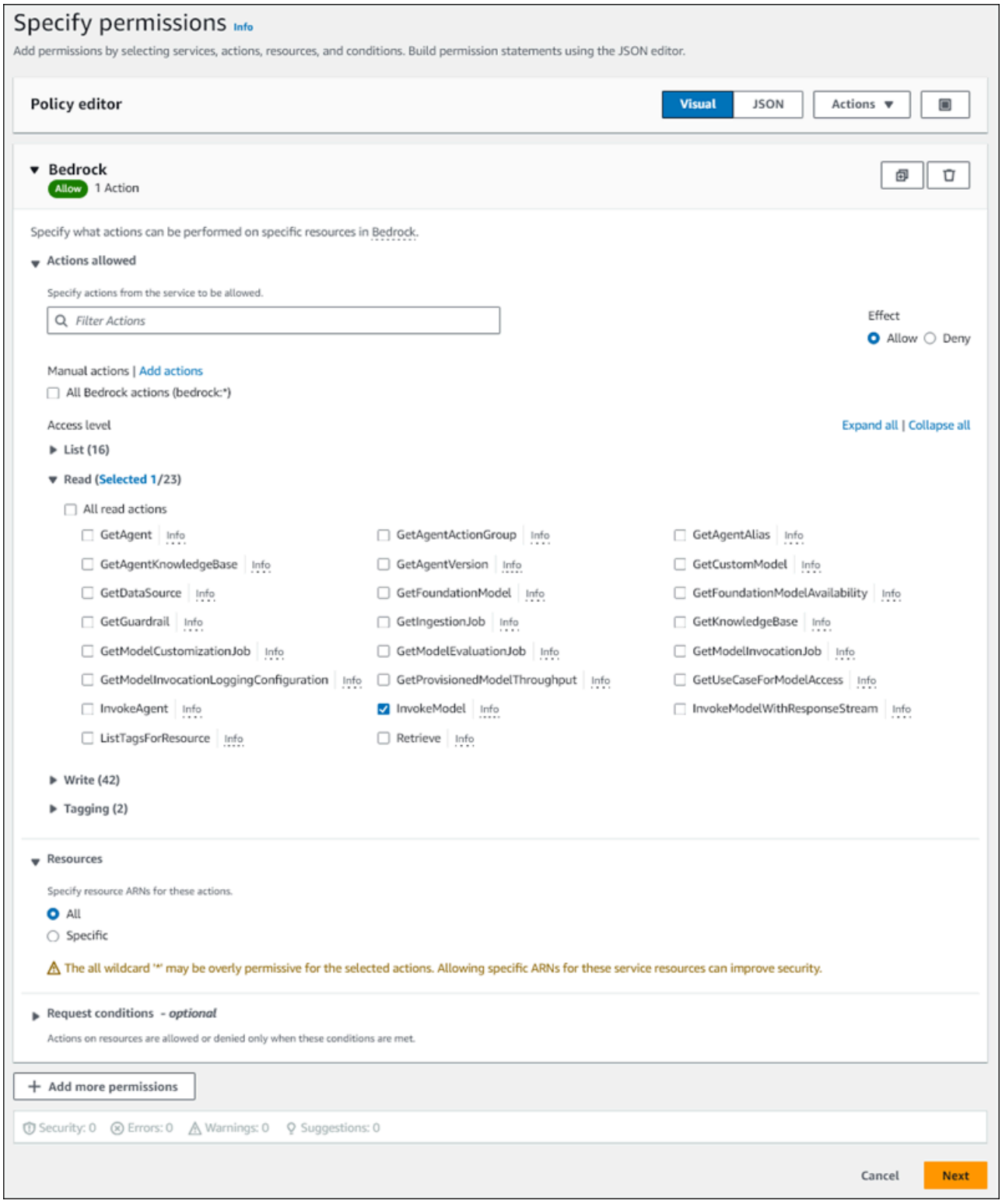
Para criar a política do IAM

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Selecione Políticas no painel de navegação.
3. Escolha Create a policy (Criar uma política).
4. Na página Especificar permissões, em Selecionar um serviço, escolha Bedrock.

   As permissões do Amazon Bedrock são exibidas.
5. Expanda Leitura e selecione InvokeModel.

6. Em Recursos, selecione Tudo.

   A página Especificar permissões deve ser semelhante à figura a seguir.



7. Escolha Próximo.

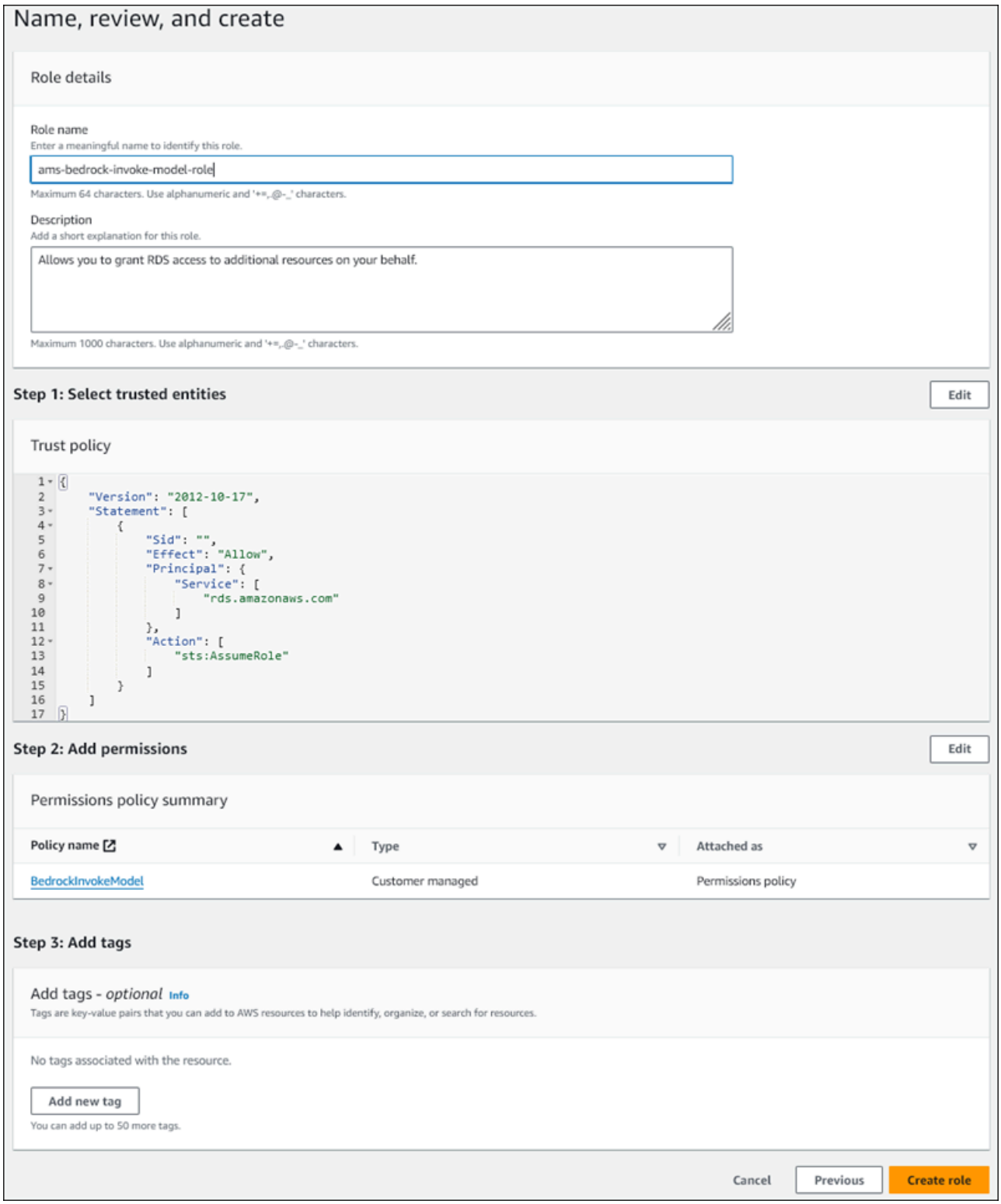
8. Na página Revisar e criar, insira um nome para a política, por exemplo, **`BedrockInvokeModel`**.
9. Analise a política e escolha Criar política.

Depois, crie o perfil do IAM que usa a política de permissão do Amazon Bedrock.

Para criar perfil do IAM

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Selecione Roles (Funções) no painel de navegação.
3. Selecione Criar perfil.
4. Na página Selecionar entidade confiável, para Caso de uso, selecione RDS.
5. Selecione RDS: adicionar perfil ao banco de dados e escolha Próximo.
6. Na página Adicionar permissões, em Políticas de permissões, selecione a política do IAM que você criou e escolha Próximo.
7. Na página Nomear, revisar e criar, insira um nome para o perfil, por exemplo, **`ams-bedrock-invoke-model-role`**.

   O perfil deve se parecer com a figura a seguir.



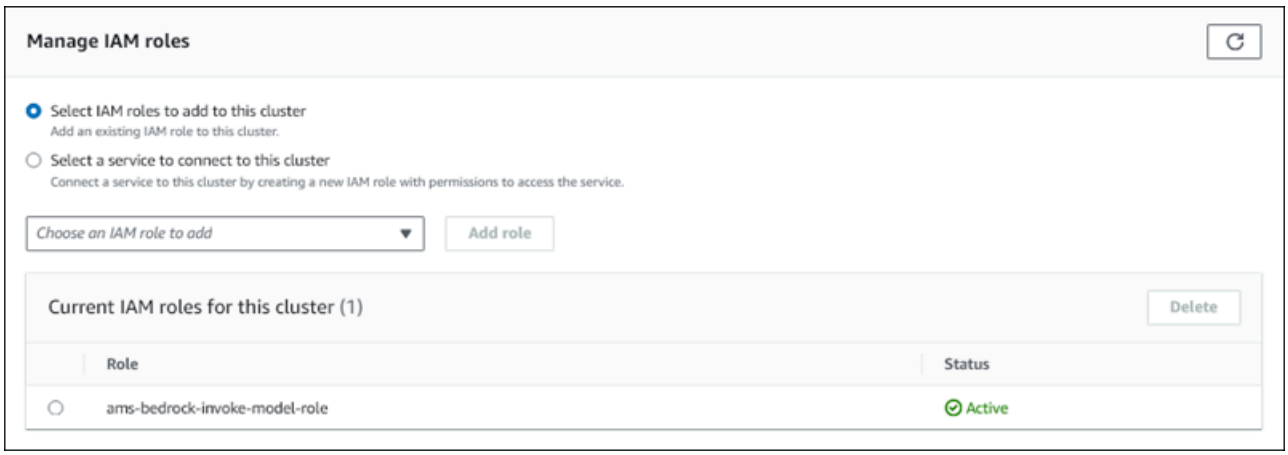
8. Revise o perfil e selecione Criar perfil.

Depois, associe o perfil do IAM do Amazon Bedrock ao cluster de banco de dados.

Como associar o perfil do IAM ao cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados) no painel de navegação.
3. Selecione o cluster de banco de dados do Aurora MySQL que você deseja conectar aos serviços do Amazon Bedrock.
4. Escolha a guia Connectivity & security (Conectividade e segurança).
5. Na seção Gerenciar perfis do IAM, escolha Selecionar os perfis do IAM a serem adicionadas a esse cluster.
6. Escolha o perfil do IAM que você criou e selecione Adicionar perfil.

   O perfil do IAM está associado ao cluster de banco de dados, primeiro com o status Pendente e, depois, Ativo. Quando o processo for concluído, você poderá encontrar o perfil na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster).



É necessário adicionar o ARN desse perfil do IAM ao parâmetro `aws_default_bedrock_role` do grupo de parâmetros do cluster de banco de dados personalizado associado ao cluster de banco de dados do Aurora MySQL. Se seu cluster de banco de dados do Aurora MySQL não usa um grupo de parâmetros de cluster de banco de dados personalizado, será necessário criar um para ser usado com seu cluster de banco de dados do Aurora MySQL a fim de concluir a integração. Para obter mais informações, consulte Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

Como configurar o parâmetro de cluster de banco de dados

1. No console do Amazon RDS, abra a guia Configuration (Configuração) de seu cluster de banco de dados do Aurora MySQL.
2. Localize o grupo de parâmetros de cluster de banco de dados configurado para o cluster. Selecione o link para abrir o grupo de parâmetros de cluster de banco de dados personalizado e escolha Editar.
3. Encontre o parâmetro `aws_default_bedrock_role` em seu grupo de parâmetros de cluster de banco de dados personalizado.
4. No campo Valor, insira o ARN do perfil do IAM.
5. Selecione Save changes (Salvar alterações) para salvar a configuração.
6. Reinicialize a instância primária de seu cluster de banco de dados do Aurora MySQL para que essa configuração de parâmetro entre em vigor.

A integração do IAM com o Amazon Bedrock está concluída. Continue configurando o cluster de banco de dados do Aurora MySQL para trabalhar com o Amazon Bedrock de acordo com Conceder aos usuários de banco de dados acesso ao machine learning do Aurora.

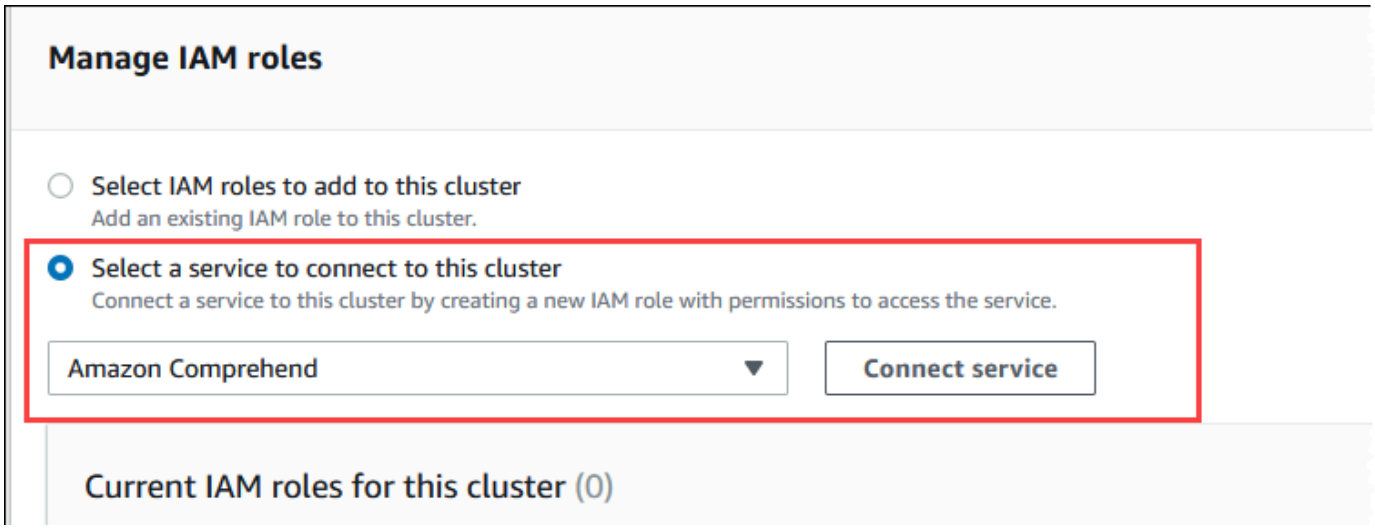
Configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Comprehend

O Aurora Machine Learning depende de políticas e perfis do AWS Identity and Access Management para permitir que seu cluster de banco de dados do Aurora MySQL acesse e use os serviços do Amazon Comprehend. O procedimento a seguir cria automaticamente um perfil e uma política do IAM para seu cluster para que ele possa usar o Amazon Comprehend.

Como configurar o cluster de banco de dados do Aurora MySQL para usar o Amazon Comprehend

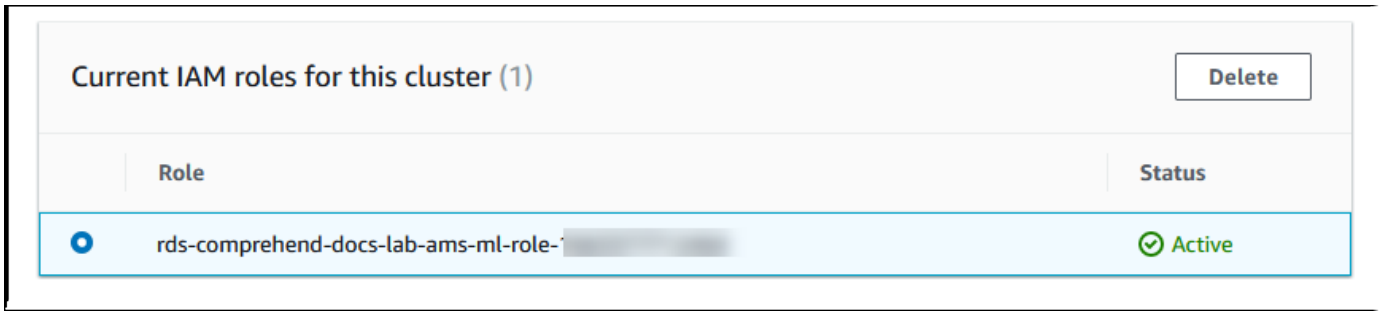
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Escolha Databases (Bancos de dados) no painel de navegação.
3. Selecione o cluster de banco de dados do Aurora MySQL que você deseja conectar aos serviços do Amazon Comprehend.
4. Escolha a guia Connectivity & security (Conectividade e segurança).
5. Na seção Gerenciar perfis do IAM, selecione Selecionar um serviço para se conectar a esse cluster.

6. Selecione Amazon Comprehend no menu e escolha Conectar serviço.



7. A caixa de diálogo Connect cluster to Amazon Comprehend (Conectar cluster ao Amazon Comprehend) não exige nenhuma informação adicional. No entanto, você pode ver uma mensagem notificando que a integração entre o Aurora e o Amazon Comprehend está atualmente em visualização prévia. Não deixe de ler a mensagem antes de continuar. Selecione Cancelar se preferir não continuar.
8. Selecione Connect service (Conectar serviço) para concluir o processo de integração.

   O Aurora cria o perfil do IAM. Ele também cria a política que permite que o cluster de banco de dados do Aurora MySQL use os serviços do Amazon Comprehend e anexa a política ao perfil. Quando o processo for concluído, você poderá encontrar o perfil na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster), conforme mostrado na imagem a seguir.



   É necessário adicionar o ARN desse perfil do IAM ao parâmetro `aws_default_comprehend_role` do grupo de parâmetros do cluster de banco de dados personalizado associado ao cluster de banco de dados do Aurora MySQL. Se seu cluster de banco de dados do Aurora MySQL não usa um grupo de parâmetros de cluster de banco de dados personalizado, será necessário criar um para ser usado com seu cluster de banco de dados do Aurora MySQL a fim de concluir a integração. Para obter mais informações, consulte

Grupos de parâmetros do cluster de banco de dados para clusters de banco de dados do Amazon Aurora.

Depois de criar seu grupo de parâmetros de cluster de banco de dados personalizado e associá-lo ao seu cluster de banco de dados do Aurora MySQL, você poderá continuar seguindo estas etapas.

Se o cluster usa um grupo de parâmetros de cluster de banco de dados personalizado, faça o seguinte.

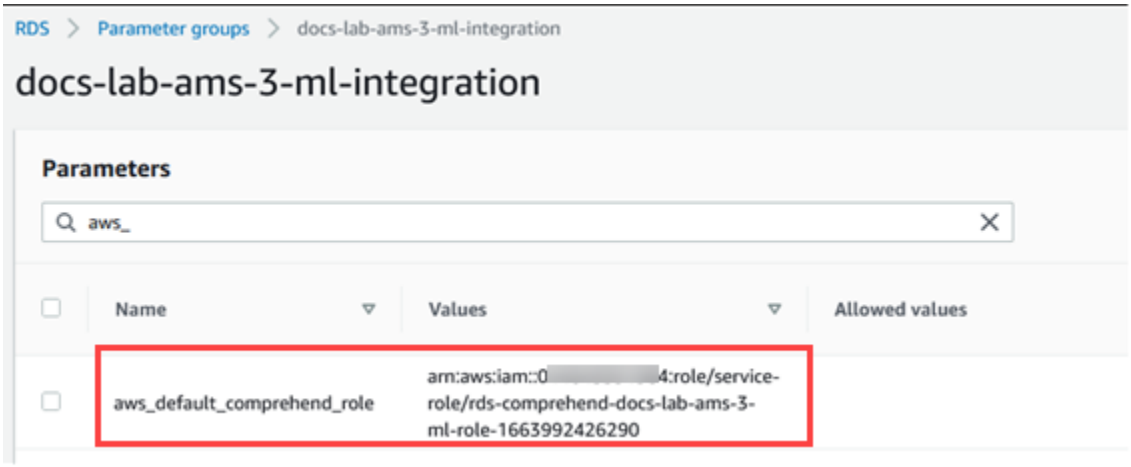
a. No console do Amazon RDS, abra a guia Configuration (Configuração) de seu cluster de banco de dados do Aurora MySQL.

b. Localize o grupo de parâmetros de cluster de banco de dados configurado para o cluster. Selecione o link para abrir o grupo de parâmetros de cluster de banco de dados personalizado e escolha Editar.

c. Encontre o parâmetro `aws_default_comprehend_role` em seu grupo de parâmetros de cluster de banco de dados personalizado.

d. No campo Valor, insira o ARN do perfil do IAM.

e. Selecione Save changes (Salvar alterações) para salvar a configuração. Na imagem a seguir, é fornecido um exemplo.

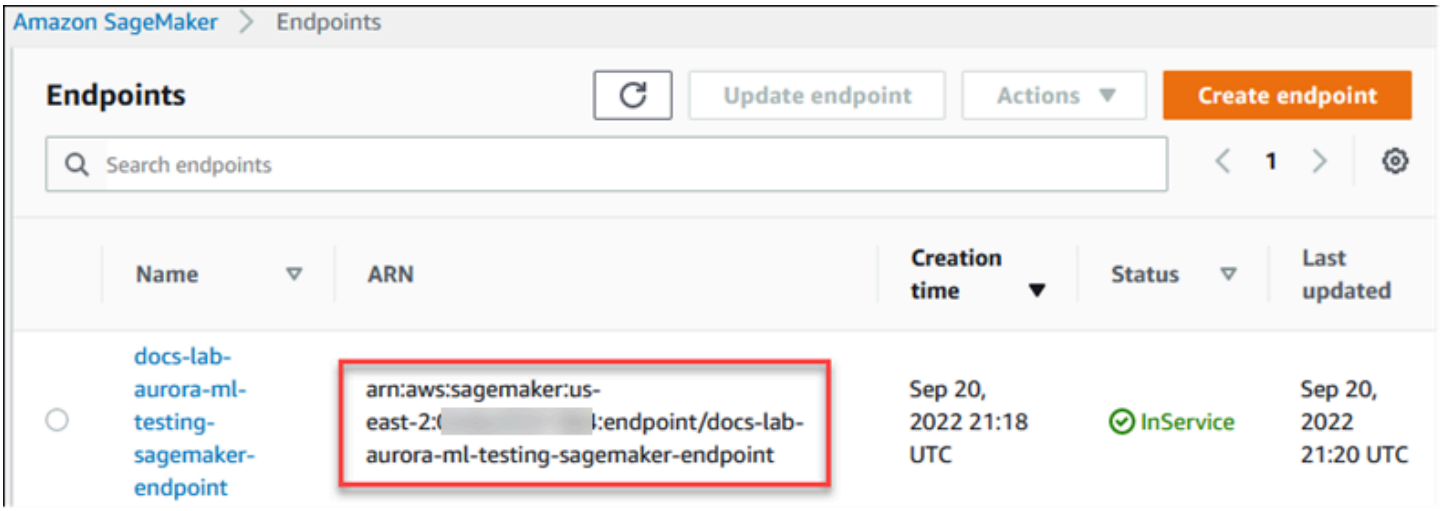


Reinicialize a instância primária de seu cluster de banco de dados do Aurora MySQL para que essa configuração de parâmetro entre em vigor.

A integração do IAM com o Amazon Comprehend está concluída. Continue configurando seu cluster de banco de dados do Aurora MySQL para trabalhar com o Amazon Comprehend concedendo acesso aos usuários apropriados do banco de dados.

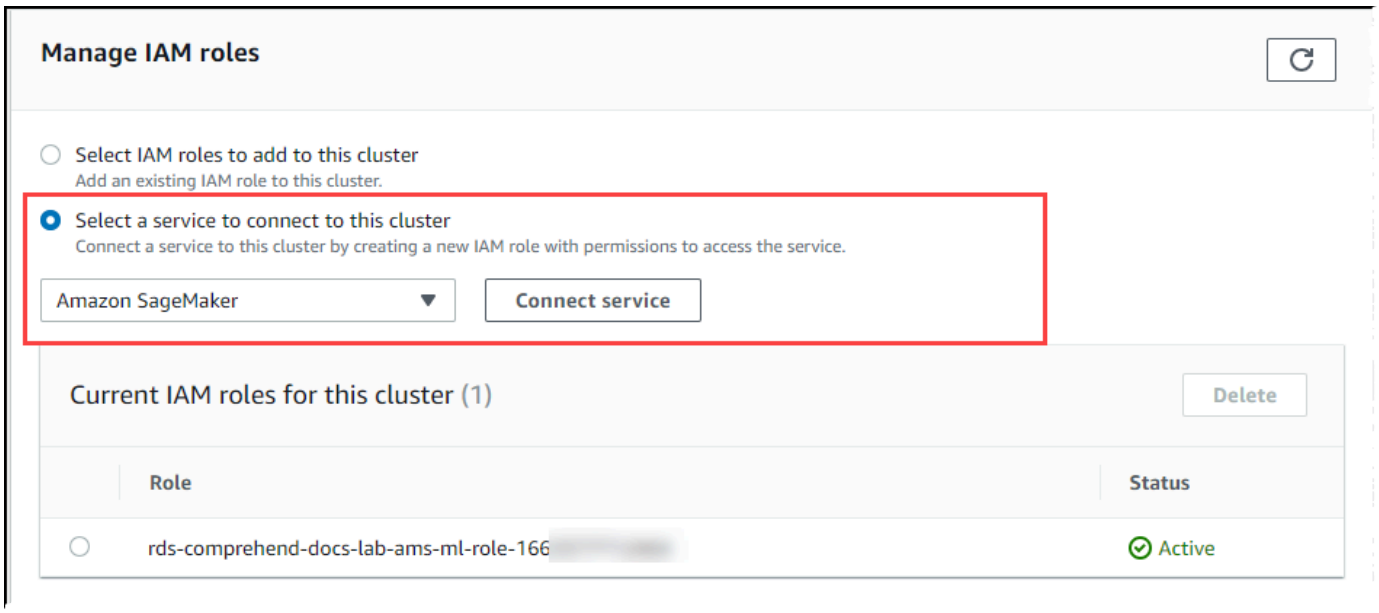
Configuração do cluster de banco de dados do Aurora MySQL para usar o SageMaker AI

O procedimento a seguir cria automaticamente o perfil e uma política do IAM para seu cluster de banco de dados do Aurora MySQL para que ele possa usar o SageMaker AI. Antes de tentar seguir esse procedimento, é necessário que o endpoint do SageMaker AI esteja disponível para poder inseri-lo quando necessário. Normalmente, os cientistas de dados de sua equipe trabalham para produzir um endpoint que você possa usá-lo em seu cluster de banco de dados do Aurora MySQL. Você pode encontrar esses endpoints no console do SageMaker AI. No painel de navegação, abra o menu Inference (Inferência) e selecione Endpoints. Na imagem a seguir, é fornecido um exemplo.

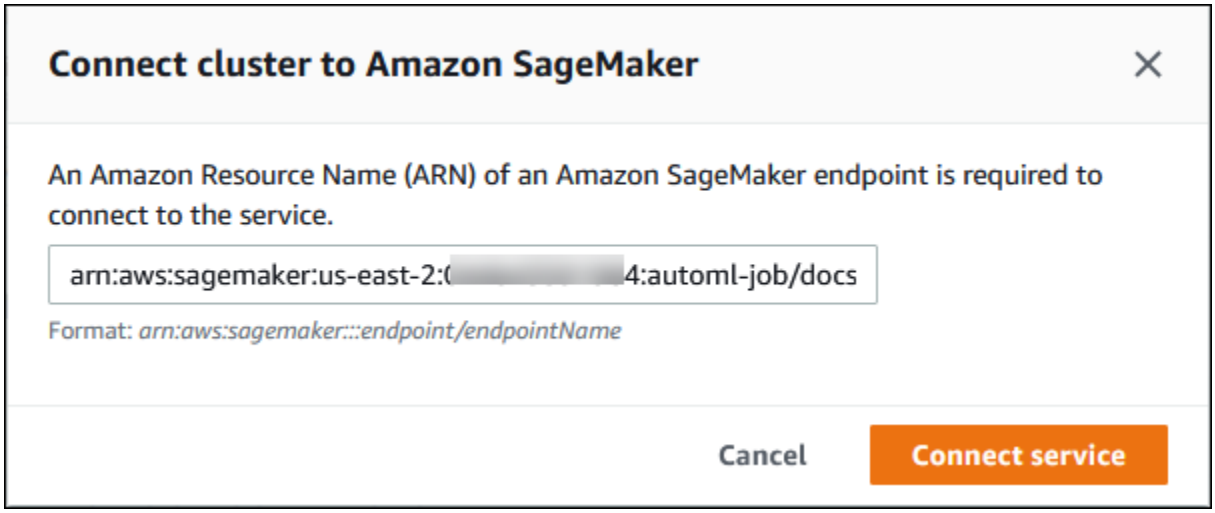


Como configurar o cluster de banco de dados do Aurora MySQL para usar o SageMaker AI

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Selecione Bancos de dados no menu de navegação do Amazon RDS e depois o cluster de banco de dados do Aurora MySQL que deseja conectar aos serviços do SageMaker AI.
3. Escolha a guia Connectivity & security (Conectividade e segurança).
4. Role a página para a seção Manage IAM roles (Gerenciar perfis do IAM) e depois Select a service to connect to this cluster (Selecionar um serviço para se conectar a esse cluster). Selecione SageMaker AI no seletor.



5. Escolha Connect service (Conectar serviço).
6. Na caixa de diálogo Conectar cluster ao SageMaker AI, insira o ARN do endpoint do SageMaker AI.



7. O Aurora cria o perfil do IAM. Ele também cria a política que permite que o cluster de banco de dados do Aurora MySQL use os serviços do SageMaker AI e anexa a política ao perfil. Quando o processo for concluído, você poderá encontrar o perfil na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster).
8. Abra o console do IAM em https://console.aws.amazon.com/iam/.
9. Selecione Roles (Perfis) na seção Access management (Gerenciamento de acesso) do menu de navegação do AWS Identity and Access Management.
10. Encontre o perfil dentre os listados. Seu nome usa o padrão a seguir.

```
rds-sagemaker-your-cluster-name-role-auto-generated-digits
```

11. Abra a página de resumo da função e localize o ARN. Anote o ARN ou copie-o usando o widget de cópia.
12. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
13. Selecione o cluster de banco de dados do Aurora MySQL e depois a guia Configuration (Configuração).
14. Localize o grupo de parâmetros de cluster de banco de dados e depois clique no link para abrir o grupo de parâmetros de cluster de banco de dados personalizado. Encontre o parâmetro `aws_default_sagemaker_role` e insira o ARN do perfil do IAM no campo Value (Valor) e salve a configuração.
15. Reinicialize a instância primária de seu cluster de banco de dados do Aurora MySQL para que essa configuração de parâmetro entre em vigor.

Agora a configuração do IAM está concluída. Continue configurando seu cluster de banco de dados do Aurora MySQL para trabalhar com o SageMaker AI concedendo acesso aos usuários apropriados do banco de dados.

Se você quiser usar seus modelos do SageMaker AI para treinamento em vez de usar componentes pré-criados do SageMaker AI, também precisará adicionar o bucket do Amazon S3 ao seu cluster de banco de dados do Aurora MySQL, conforme descrito no Configuração do cluster de banco de dados do Aurora MySQL para usar o Amazon S3 para SageMaker AI (opcional) a seguir.

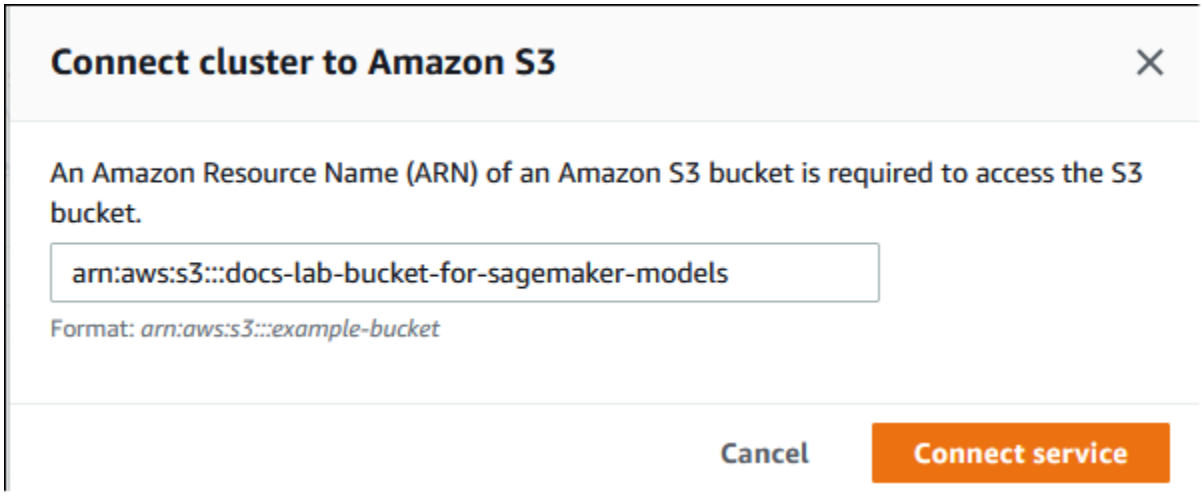
Configuração do cluster de banco de dados do Aurora MySQL para usar o Amazon S3 para SageMaker AI (opcional)

Para usar o SageMaker AI com seus próprios modelos em vez de usar componentes pré-criados fornecidos pelo SageMaker AI, é necessário configurar um bucket do Amazon S3 para o cluster de banco de dados do Aurora MySQL usar. Para obter mais informações sobre como criar um bucket do Amazon S3, consulte Criação de um bucket, no Guia do usuário do Amazon Simple Storage Service.

Como configurar o cluster de banco de dados do Aurora MySQL para usar um bucket do Amazon S3 para SageMaker AI

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

2. Selecione Bancos de dados no menu de navegação do Amazon RDS e depois o cluster de banco de dados do Aurora MySQL que deseja conectar aos serviços do SageMaker AI.
3. Escolha a guia Connectivity & security (Conectividade e segurança).
4. Role a página para a seção Manage IAM roles (Gerenciar perfis do IAM) e, depois, Select a service to connect to this cluster (Selecionar um serviço para se conectar a esse cluster). Selecione Amazon S3 no seletor.
5. Escolha Connect service (Conectar serviço).
6. Na caixa de diálogo Conectar cluster ao Amazon S3, insira o ARN do bucket do Amazon S3, como mostrado na imagem a seguir.



7. Selecione Connect service (Conectar serviço) para concluir o processo.

Para obter mais informações sobre o uso de buckets do Amazon S3 com o SageMaker AI, consulte Especificar um bucket do Amazon S3 para fazer upload de conjuntos de dados de treinamento e armazenar dados de saída no Guia do desenvolvedor do Amazon SageMaker AI. Para saber mais sobre como trabalhar com o SageMaker AI, consulte Começar a usar instâncias de notebook do Amazon SageMaker AI no Guia do desenvolvedor do Amazon SageMaker AI.

Conceder aos usuários de banco de dados acesso ao machine learning do Aurora

Os usuários do banco de dados devem receber permissão para invocar as funções de machine learning do Aurora. A forma como você concede a permissão depende da versão do MySQL que você usa para seu cluster de banco de dados do Aurora MySQL, conforme descrito a seguir. A forma como você faz isso depende da versão do MySQL que seu cluster de banco de dados do Aurora MySQL usa.

- Para o Aurora MySQL versão 3 (compatível com o MySQL 8.0), os usuários do banco de dados devem receber o perfil de banco de dados apropriado. Consulte mais informações em Using Roles no MySQL 8.0 Reference Manual.
- Para o Aurora MySQL versão 2 (compatível com o MySQL 5.7), os usuários do banco de dados recebem privilégios. Consulte mais informações em Access Control and Account Management no MySQL 5.7 Reference Manual.

A tabela a seguir mostra os perfis e privilégios que os usuários do banco de dados precisam para trabalhar com funções de machine learning.

Aurora MySQL versão 3 (perfil)	Aurora MySQL versão 2 (privilégio)
AWS_BEDROCK_ACCESS	–
AWS_COMPREHEND_ACCESS	INVOCAR COMPREHEND
AWS_SAGEMAKER_ACCESS	INVOCAR SAGEMAKER

Conceder acesso às funções do Amazon Bedrock

Para conceder aos usuários do banco de dados acesso às funções do Amazon Bedrock, use a seguinte instrução SQL:

```
GRANT AWS_BEDROCK_ACCESS TO user@domain-or-ip-address;
```

Os usuários do banco de dados também precisam receber permissões EXECUTE para as funções criadas para trabalhar com o Amazon Bedrock:

```
GRANT EXECUTE ON FUNCTION database_name.function_name TO user@domain-or-ip-address;
```

Por fim, os usuários do banco de dados devem ter seus perfis definidos como AWS_BEDROCK_ACCESS:

```
SET ROLE AWS_BEDROCK_ACCESS;
```

As funções do Amazon Bedrock já estão disponíveis para uso.

Conceder acesso às funções do Amazon Comprehend

Para conceder aos usuários do banco de dados acesso às funções do Amazon Comprehend, use a declaração apropriada para sua versão do Aurora MySQL.

- Aurora MySQL versão 3 (compatível com o MySQL 8.0)

```
GRANT AWS_COMPREHEND_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versão 2 (compatível com o MySQL 5.7)

```
GRANT INVOKE COMPREHEND ON *.* TO user@domain-or-ip-address;
```

As funções do Amazon Comprehend agora estão disponíveis para uso. Para obter exemplos de uso, consulte Usar o Amazon Comprehend com seu cluster de banco de dados do Aurora MySQL.

Concessão de acesso às funções do SageMaker AI

Para conceder aos usuários do banco de dados acesso às funções do SageMaker AI, use a declaração apropriada para sua versão do Aurora MySQL.

- Aurora MySQL versão 3 (compatível com o MySQL 8.0)

```
GRANT AWS_SAGEMAKER_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versão 2 (compatível com o MySQL 5.7)

```
GRANT INVOKE SAGEMAKER ON *.* TO user@domain-or-ip-address;
```

Os usuários do banco de dados também precisam receber permissões `EXECUTE` para as funções criadas para trabalhar com o SageMaker AI. Suponha que você tenha criado duas funções, `db1.anomoly_score` e `db2.company_forecasts`, para invocar os serviços de seu endpoint do SageMaker AI. Você deve conceder privilégios de execução conforme mostrado no exemplo a seguir.

```
GRANT EXECUTE ON FUNCTION db1.anomaly_score TO user1@domain-or-ip-address1;
GRANT EXECUTE ON FUNCTION db2.company_forecasts TO user2@domain-or-ip-address2;
```

As funções do SageMaker AI agora estão disponíveis para uso. Para obter exemplos de uso, consulte Uso do SageMaker AI com o cluster de banco de dados do Aurora MySQL.

Usar o Amazon Bedrock com o cluster de banco de dados do Aurora MySQL

Para usar o Amazon Bedrock, você cria uma função definida pelo usuário (UDF) no banco de dados do Aurora MySQL que invoca um modelo. Consulte mais informações em Supported models in Amazon Bedrock no Guia do usuário do Amazon Bedrock.

Uma UDF usa a seguinte sintaxe:

```
CREATE FUNCTION function_name (argument type)
        [DEFINER = user]
    RETURNS mysql_data_type
    [SQL SECURITY {DEFINER | INVOKER}]
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'model_id'
    [CONTENT_TYPE 'content_type']
    [ACCEPT 'content_type']
    [TIMEOUT_MS timeout_in_milliseconds];
```

- As funções do Amazon Bedrock não são compatíveis com `RETURNS JSON`. É possível usar `CONVERT` ou `CAST` para converter de `TEXT` para `JSON`, se necessário.
- Se você não especificar `CONTENT_TYPE` ou `ACCEPT`, o padrão será `application/json`.
- Se você não especificar `TIMEOUT_MS`, o valor de `aurora_ml_inference_timeout` será usado.

Por exemplo, a UDF a seguir invoca o modelo Amazon Titan Text Express:

```
CREATE FUNCTION invoke_titan (request_body TEXT)
    RETURNS TEXT
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'amazon.titan-text-express-v1'
    CONTENT_TYPE 'application/json'
    ACCEPT 'application/json';
```

Para permitir que um usuário do banco de dados use essa função, use o seguinte comando do SQL:

```
GRANT EXECUTE ON FUNCTION database_name.invoke_titan TO user@domain-or-ip-address;
```

Depois, o usuário pode chamar `invoke_titan` como qualquer outra função, conforme mostrado no exemplo a seguir. Formate o corpo da solicitação de acordo com os modelos de texto do Amazon Titan.

```
CREATE TABLE prompts (request varchar(1024));
INSERT INTO prompts VALUES (
'{
    "inputText": "Generate synthetic data for daily product sales in various categories
 - include row number, product name, category, date of sale and price. Produce output
 in JSON format. Count records and ensure there are no more than 5.",
    "textGenerationConfig": {
        "maxTokenCount": 1024,
        "stopSequences": [],
        "temperature":0,
        "topP":1
    }
}');

SELECT invoke_titan(request) FROM prompts;

{"inputTextTokenCount":44,"results":[{"tokenCount":296,"outputText":"
```tabular-data-json
{
    "rows": [
        {
            "Row Number": "1",
            "Product Name": "T-Shirt",
            "Category": "Clothing",
            "Date of Sale": "2024-01-01",
            "Price": "$20"
        },
        {
            "Row Number": "2",
            "Product Name": "Jeans",
            "Category": "Clothing",
            "Date of Sale": "2024-01-02",
            "Price": "$30"
        },
        {
            "Row Number": "3",
            "Product Name": "Hat",
            "Category": "Accessories",
            "Date of Sale": "2024-01-03",
```
```

```
            "Price": "$15"
        },
        {
            "Row Number": "4",
            "Product Name": "Watch",
            "Category": "Accessories",
            "Date of Sale": "2024-01-04",
            "Price": "$40"
        },
        {
            "Row Number": "5",
            "Product Name": "Phone Case",
            "Category": "Accessories",
            "Date of Sale": "2024-01-05",
            "Price": "$25"
        }
    ]
}
```","completionReason":"FINISH"}]}

Para outros modelos que você usa, formate o corpo da solicitação de forma adequada para eles. Consulte mais informações em Inference parameters for foundation models no Guia do usuário do Amazon Bedrock.

Usar o Amazon Comprehend com seu cluster de banco de dados do Aurora MySQL

Para o Aurora MySQL, o machine learning do Aurora fornece as duas funções integradas a seguir para trabalhar com o Amazon Comprehend e seus dados de texto. Forneça o texto para analisar (`input_data`) e especifique o idioma (`language_code`).

aws_comprehend_detect_sentiment

Essa função identifica uma postura emocional positiva, negativa, neutra ou mista no texto. A documentação de referência dessa função é a seguinte.

```
aws_comprehend_detect_sentiment(
  input_text,
  language_code
  [,max_batch_size]
)
```

Para saber mais, consulte Sentiment (Sentimento) no Amazon Comprehend Developer Guide (Guia do desenvolvedor do Amazon Comprehend).

aws_comprehend_detect_sentiment_confidence

Essa função mede o nível de confiança do sentimento detectado em determinado texto. Ele retorna um valor (tipo, `double`) que indica a confiança do sentimento atribuído pela função aws_comprehend_detect_sentiment ao texto. A confiança é uma métrica estatística entre 0 e 1. Quanto maior for o nível de confiança, mais peso você poderá atribuir ao resultado. Veja um resumo da documentação da função.

```
aws_comprehend_detect_sentiment_confidence(
  input_text,
  language_code
  [,max_batch_size]
)
```

Nas duas funções (aws_comprehend_detect_sentiment_confidence, aws_comprehend_detect_sentiment), o `max_batch_size` usa um valor padrão de 25 se nenhum for especificado. O tamanho do lote deve ser sempre maior que 0. Você pode usar `max_batch_size` para ajustar a performance das chamadas de função do Amazon Comprehend. Um lote grande troca performance mais alta por maior uso de memória no cluster de banco de dados do Aurora MySQL. Para obter mais informações, consulte Considerações de performance para usar o machine learning do Aurora com o Aurora MySQL.

Para obter mais informações sobre parâmetros e tipos de retorno para as funções de detecção de sentimentos no Amazon Comprehend, consulte DetectSentiment

Example Exemplo: uma consulta simples utilizando as funções do Amazon Comprehend

Veja um exemplo de uma consulta simples que invoca essas duas funções para ver o nível de satisfação de clientes com sua equipe de atendimento. Suponha que você tenha uma tabela de banco de dados (`support`) que armazene o feedback do cliente após cada solicitação de ajuda. Esta consulta de exemplo aplica as duas funções incorporadas ao texto na coluna `feedback` da tabela e gera os resultados. Os valores de confiança retornados pela função são duplos entre 0.0 e 1.0. Para obter uma saída mais legível, esta consulta arredonda os resultados para 6 pontos decimais. Para facilitar as comparações, ela também classifica primeiro os resultados em ordem decrescente, começando pelo resultado com o maior grau de confiança.

```
SELECT feedback AS 'Customer feedback',
    aws_comprehend_detect_sentiment(feedback, 'en') AS Sentiment,
    ROUND(aws_comprehend_detect_sentiment_confidence(feedback, 'en'), 6)
    AS Confidence FROM support
    ORDER BY Confidence DESC;
+-----------------------------------------------------------+-----------+------------+
| Customer feedback                                         | Sentiment | Confidence |
+-----------------------------------------------------------+-----------+------------+
| Thank you for the excellent customer support!             | POSITIVE  |   0.999771 |
| The latest version of this product stinks!                | NEGATIVE  |   0.999184 |
| Your support team is just awesome! I am blown away.       | POSITIVE  |   0.997774 |
| Your product is too complex, but your support is great.   | MIXED     |   0.957958 |
| Your support tech helped me in fifteen minutes.           | POSITIVE  |   0.949491 |
| My problem was never resolved!                            | NEGATIVE  |   0.920644 |
| When will the new version of this product be released?    | NEUTRAL   |   0.902706 |
| I cannot stand that chatbot.                              | NEGATIVE  |   0.895219 |
| Your support tech talked down to me.                      | NEGATIVE  |   0.868598 |
| It took me way too long to get a real person.             | NEGATIVE  |   0.481805 |
+-----------------------------------------------------------+-----------+------------+
 10 rows in set (0.1898 sec)
```

Example Exemplo: determinar o sentimento médio para um texto acima de um nível de confiança específico

Uma típica consulta do Amazon Comprehend procura linhas em que o sentimento é um valor especificado, com um nível de confiança superior a um número especificado. Por exemplo, a consulta a seguir mostra como você pode determinar o sentimento médio de documentos em seu banco de dados. A consulta considera somente documentos em que a confiança da avaliação é superior a 80%.

```
SELECT AVG(CASE aws_comprehend_detect_sentiment(productTable.document, 'en')
  WHEN 'POSITIVE' THEN 1.0
  WHEN 'NEGATIVE' THEN -1.0
  ELSE 0.0 END) AS avg_sentiment, COUNT(*) AS total
FROM productTable
WHERE productTable.productCode = 1302 AND
    aws_comprehend_detect_sentiment_confidence(productTable.document, 'en') >= 0.80;
```

Uso do SageMaker AI com o cluster de banco de dados do Aurora MySQL

Para usar a funcionalidade do SageMaker AI de seu cluster de banco de dados do Aurora MySQL, você precisa criar funções armazenadas que incorporem suas chamadas ao endpoint do SageMaker AI e seus recursos de inferência. Isso é feito usando `CREATE FUNCTION` do MySQL geralmente da mesma forma realizada para outras tarefas de processamento em seu cluster de banco de dados do Aurora MySQL.

Para usar modelos implantados no SageMaker AI para inferência, crie funções definidas pelo usuário usando as instruções da linguagem de definição de dados (DDL) do MySQL para funções armazenadas. Cada função armazenada representa o endpoint do SageMaker AI que hospeda o modelo. Ao definir tal função, especifique os parâmetros de entrada para o modelo, o endpoint do SageMaker AI específico a ser invocado e o tipo de retorno. A função retorna a inferência computada pelo endpoint do SageMaker AI após a aplicação do modelo com os parâmetros de entrada.

Todas as funções armazenadas de machine learning do Aurora retornam tipos numéricos ou `VARCHAR`. Você pode usar qualquer tipo numérico, exceto `BIT`. Outros tipos, como `JSON`, `BLOB`, `TEXT` e `DATE`, não são permitidos.

O exemplo a seguir mostra a sintaxe `CREATE FUNCTION` para trabalhar com o SageMaker AI.

```
CREATE FUNCTION function_name (
     arg1 type1,
     arg2 type2, ...)
     [DEFINER = user]
  RETURNS mysql_type
  [SQL SECURITY { DEFINER | INVOKER } ]
  ALIAS AWS_SAGEMAKER_INVOKE_ENDPOINT
  ENDPOINT NAME 'endpoint_name'
  [MAX_BATCH_SIZE max_batch_size];
```

Essa é uma extensão da declaração DDL `CREATE FUNCTION` regular. Na instrução `CREATE FUNCTION` que define a função do SageMaker AI, não especifique um corpo de função. Em vez disso, especifique a palavra-chave `ALIAS` no lugar geralmente usado para o corpo da função. No momento, o machine learning do Aurora apenas oferece suporte a `aws_sagemaker_invoke_endpoint` para essa sintaxe estendida. Você deve especificar o parâmetro `endpoint_name`. Um endpoint do SageMaker AI pode ter características diferentes para cada modelo.

> **Note**
>
> Para obter mais informações sobre `CREATE FUNCTION`, consulte CREATE PROCEDURE and CREATE FUNCTION Statements (Declarações CREATE PROCEDURE e CREATE FUNCTION) no MySQL 8.0 Reference Manual (Manual de referência do MySQL 8.0).

O parâmetro `max_batch_size` é opcional. Por padrão, o tamanho máximo do lote é 10 mil. Você pode usar esse parâmetro em sua função para restringir o número máximo de entradas processadas em uma solicitação em lote para o SageMaker AI. O parâmetro `max_batch_size` pode ajudar a evitar um erro causado por entradas muito grandes, ou para fazer o SageMaker AI retornar uma resposta com mais rapidez. O parâmetro afeta o tamanho de um buffer interno usado para processamento de solicitações do SageMaker AI. Especificar um valor muito grande para `max_batch_size` pode causar sobrecarga substancial de memória em sua instância de banco de dados.

Recomendamos manter a configuração `MANIFEST` em seu valor padrão de `OFF`. Embora seja possível usar a opção `MANIFEST ON`, alguns recursos do SageMaker AI não podem usar diretamente o CSV exportado com essa opção. O formato do manifesto não é compatível com o formato de manifesto esperado do SageMaker AI.

Crie uma função armazenada separada para cada um de seus modelos do SageMaker AI. Esse mapeamento de funções para modelos é necessário pois um endpoint está associado a um modelo específico e cada modelo aceita parâmetros diferentes. Uso de tipos de SQL para as entradas do modelo e o tipo de saída do modelo ajuda a evitar erros de conversão de tipos ao movimentar dados entre serviços da AWS. É possível controlar quem pode aplicar o modelo. Você também pode controlar as características do tempo de execução especificando um parâmetro que representa o tamanho máximo do lote.

No momento, as funções de machine learning do Aurora têm a propriedade `NOT DETERMINISTIC`. Se você não especificar essa propriedade explicitamente, o Aurora definirá `NOT DETERMINISTIC` automaticamente. Esse requisito se deve ao fato de ser possível alterar o modelo do SageMaker AI sem nenhuma notificação ao banco de dados. Se isso acontecer, as chamadas para uma função de machine learning do Aurora talvez retornem resultados diferentes para a mesma entrada em uma única transação.

Não é possível usar as características `CONTAINS SQL`, `NO SQL`, `READS SQL DATA` e `MODIFIES SQL DATA` na instrução `CREATE FUNCTION`.

Veja a seguir um exemplo de uso de invocação de um endpoint do SageMaker AI para detectar anomalias. Há um endpoint `random-cut-forest-model` do SageMaker AI. O modelo correspondente já foi treinado pelo algoritmo `random-cut-forest`. Para cada entrada, o modelo retorna uma pontuação de anomalia. Este exemplo mostra os pontos de dados cuja pontuação é maior do que 3 desvios padrão (aproximadamente o percentil 99,9) da pontuação média.

```
CREATE FUNCTION anomaly_score(value real) returns real
  alias aws_sagemaker_invoke_endpoint endpoint name 'random-cut-forest-model-demo';

set @score_cutoff = (select avg(anomaly_score(value)) + 3 * std(anomaly_score(value))
 from nyc_taxi);

select *, anomaly_detection(value) score from nyc_taxi
  where anomaly_detection(value) > @score_cutoff;
```

Requisito de conjunto de caracteres para funções do SageMaker AI que retornam strings

Recomenda-se especificar um conjunto de caracteres de `utf8mb4` como tipo de retorno para as funções do SageMaker AI que retornam valores de string. Se isso não for prático, use um comprimento de string grande o suficiente para que o tipo de retorno mantenha um valor representado no conjunto de caracteres `utf8mb4`. A exemplo a seguir mostra como declarar o conjunto de caracteres `utf8mb4` para a função.

```
CREATE FUNCTION my_ml_func(...) RETURNS VARCHAR(5) CHARSET utf8mb4 ALIAS ...
```

Atualmente, cada função do SageMaker AI que retorna uma string usa o conjunto de caracteres `utf8mb4` para o valor de retorno. O valor de retorno usa esse conjunto de caracteres mesmo que a sua função do SageMaker AI declare um conjunto de caracteres diferente para o tipo de retorno, de forma implícita ou explícita. Se a sua função de do SageMaker AI declarar um conjunto de caracteres diferente para o valor de retorno, os dados retornados poderão ser truncados silenciosamente se forem armazenados em uma coluna de tabela que não tenha o comprimento suficiente. Por exemplo, uma consulta com uma cláusula `DISTINCT` cria uma tabela temporária. Assim, o resultado da função do SageMaker AI poderá ser truncado devido à maneira como as strings são tratadas internamente durante uma consulta.

Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado)

Recomendamos que você comece a usar o machine learning do Aurora e o SageMaker AI usando alguns dos algoritmos fornecidos e que os cientistas de dados de sua equipe forneçam os endpoints do SageMaker AI que você pode usar com seu código SQL. A seguir, você encontrará informações mínimas sobre como usar seu próprio bucket do Amazon S3 com seus próprios modelos do SageMaker AI e o cluster de banco de dados do Aurora MySQL.

O machine learning consiste em duas etapas principais: treinamento e inferência. Para treinar modelos do SageMaker AI, exporte os dados para um bucket do Amazon S3. O bucket do Amazon S3 é usado por uma instância de bloco de anotações Jupyter do SageMaker AI para treinar o modelo antes de ser implantado. Você pode usar a instrução `SELECT INTO OUTFILE S3` para consultar dados de um cluster de banco de dados Aurora MySQL e salvá-los diretamente em arquivos de texto armazenados em um bucket do Amazon S3. Depois, a instância de bloco de anotações consumirá os dados do bucket do Amazon S3 para treinamento.

O machine learning do Aurora estende a sintaxe `SELECT INTO OUTFILE` existente no Aurora MySQL para exportar dados no formato CSV. O arquivo CSV gerado pode ser consumido diretamente por modelos que precisam desse formato para fins de treinamento.

```
SELECT * INTO OUTFILE S3 's3_uri' [FORMAT {CSV|TEXT} [HEADER]] FROM table_name;
```

A extensão oferece suporte ao formato CSV padrão.

- O formato `TEXT` é igual ao formato de exportação do MySQL existente. Esse é o formato padrão.
- O formato `CSV` é um formato recém-apresentado que segue a especificação em RFC-4180.
- Se você especificar a palavra-chave opcional `HEADER`, o arquivo de saída conterá uma linha de cabeçalho. Os rótulos na linha de cabeçalho correspondem aos nomes de coluna da instrução `SELECT`.
- Você ainda pode usar as palavras-chave `CSV` e `HEADER` como identificadores.

A sintaxe estendida e a gramática de `SELECT INTO` agora tem a seguinte estrutura:

```
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
[FORMAT {CSV|TEXT} [HEADER]]
```

```
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
```

Considerações de performance para usar o machine learning do Aurora com o Aurora MySQL

Os serviços do Amazon Bedrock, do Amazon Comprehend e do SageMaker AI fazem a maior parte do trabalho quando invocados por uma função de machine learning do Aurora. Isso significa que você pode escalar esses recursos conforme necessário, de forma independente. Para seu cluster de banco de dados do Aurora MySQL, você pode tornar suas chamadas de função o mais eficientes possível. A seguir, você encontrará algumas considerações de performance a serem observadas ao trabalhar com o machine learning do Aurora.

Modelo e prompt

O desempenho ao usar o Amazon Bedrock depende muito do modelo e do prompt que você usa. Selecione um modelo e um prompt que sejam ideais para seu caso de uso.

Cache de consultas

O cache de consulta do Aurora MySQL não funciona para funções do machine learning do Aurora. O Aurora MySQL não armazena resultados de consultas no cache de consultas para nenhuma declaração de SQL que chame funções do machine learning do Aurora.

Otimização em lotes para chamadas de função do machine learning do Aurora

O principal aspecto de performance do machine learning do Aurora que pode ser influenciado no seu cluster do Aurora é a configuração do modo em lotes para chamadas às funções armazenadas do machine learning do Aurora. As funções de machine learning geralmente exigem sobrecarga substancial, o que torna impraticável chamar um serviço externo separadamente para cada linha. O machine learning do Aurora é capaz de minimizar essa sobrecarga, combinando as chamadas para o serviço de machine learning do Aurora externo para diversas linhas em um único lote. O

machine learning do Aurora obtêm as respostas para todas as linhas de entrada e retorna essas respostas, uma linha por vez, à consulta durante a sua execução. Essa otimização melhora a taxa de transferência e a latência de suas consultas do Aurora, sem alterar os resultados.

Ao criar uma função armazenada do Aurora que está conectada a um endpoint do SageMaker AI, defina o parâmetro de tamanho do lote. Esse parâmetro influencia quantas linhas são transferidas ao SageMaker AI para cada chamada subjacente. Para consultas que processam grandes quantidades de linhas, a sobrecarga para realizar uma chamada do SageMaker AI separada para cada linha pode ser substancial. Quanto maior for o conjunto de dados processado pelo procedimento armazenado, maior poderá ser o tamanho do lote.

Se houver a possibilidade de aplicar a otimização do modo em lote a uma função do SageMaker AI, isso poderá ser determinado ao verificar o plano de consulta produzido pela instrução `EXPLAIN PLAN`. Nesse caso, a coluna `extra` no plano de execução inclui `Batched machine learning`. O exemplo a seguir mostra uma chamada para uma função do SageMaker AI que usa o modo em lote.

```
mysql> CREATE FUNCTION anomaly_score(val real) returns real alias
 aws_sagemaker_invoke_endpoint endpoint name 'my-rcf-model-20191126';
Query OK, 0 rows affected (0.01 sec)

mysql> explain select timestamp, value, anomaly_score(value) from nyc_taxi;
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows | filtered | Extra                   |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
|  1 | SIMPLE      | nyc_taxi | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |   48 |   100.00 | Batched machine learning |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
1 row in set, 1 warning (0.01 sec)
```

Ao chamar uma das funções integradas do Amazon Comprehend, você pode controlar o tamanho do lote especificando o parâmetro `max_batch_size` opcional. Esse parâmetro restringe o número máximo de valores `input_text` processados em cada lote. Ao enviar vários itens por vez, ele reduz o número de percursos de ida e volta entre o Aurora e o Amazon Comprehend. Limitar o tamanho do lote é útil em situações, como uma consulta com uma cláusula `LIMIT`. Usando um valor pequeno para `max_batch_size`, é possível evitar a invocação do Amazon Comprehend mais vezes do que a quantidade de textos de entrada.

A otimização em lotes para avaliar funções do machine learning do Aurora é aplicável nos seguintes casos:

- Chamadas de funções dentro da lista de seleção ou da cláusula `WHERE` de instruções `SELECT`
- Chamadas de funções na lista `VALUES` de instruções `INSERT` e `REPLACE`
- Funções do SageMaker AI nos valores `SET` em instruções `UPDATE`:

```
INSERT INTO MY_TABLE (col1, col2, col3) VALUES
  (ML_FUNC(1), ML_FUNC(2), ML_FUNC(3)),
  (ML_FUNC(4), ML_FUNC(5), ML_FUNC(6));
UPDATE MY_TABLE SET col1 = ML_FUNC(col2), SET col3 = ML_FUNC(col4) WHERE ...;
```

Monitorar o machine learning do Aurora

É possível monitorar as operações em lote de machine learning do Aurora consultando diversas variáveis globais, conforme mostrado no exemplo a seguir.

```
show status like 'Aurora_ml%';
```

É possível redefinir as variáveis de status usando uma declaração `FLUSH STATUS`. Assim, todas as quantidades representam totais, médias e assim por diante, desde a última vez em que a variável foi redefinida.

`Aurora_ml_logical_request_cnt`

O número de solicitações lógicas que a instância de banco de dados avaliou para serem enviadas aos serviços de machine learning do Aurora desde a última redefinição de status. Dependendo se o lote foi ou não usado, esse valor pode ser maior que `Aurora_ml_actual_request_cnt`.

`Aurora_ml_logical_response_cnt`

A contagem agregada de respostas que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados.

`Aurora_ml_actual_request_cnt`

A contagem agregada de solicitações que o Aurora MySQL faz nos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados.

`Aurora_ml_actual_response_cnt`

A contagem agregada de respostas que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados.

`Aurora_ml_cache_hit_cnt`

A contagem agregada de acertos do cache interno que o Aurora MySQL recebe dos serviços de machine learning do Aurora entre todas as consultas executadas por usuários da instância de banco de dados.

`Aurora_ml_retry_request_cnt`

O número de novas solicitações que a instância de banco de dados enviou aos serviços de machine learning do Aurora desde a última redefinição de status.

`Aurora_ml_single_request_cnt`

A contagem agregada de funções do machine learning do Aurora que são avaliadas pelo modo que não seja em lote entre todas as consultas executadas por usuários da instância de banco de dados.

Para obter informações sobre como monitorar a performance das operações do SageMaker AI chamadas a partir de funções do machine learning do Aurora, consulte o tópico sobre como Monitorar o Amazon SageMaker AI.

Usar machine learning do Amazon Aurora com o Aurora PostgreSQL

Ao usar o machine learning do Amazon Aurora com o cluster de banco de dados do Aurora PostgreSQL, é possível usar o Amazon Comprehend, o Amazon SageMaker AI ou o Amazon Bedrock, dependendo das necessidades. Esses serviços oferecem compatibilidade com casos de uso de machine learning específicos.

O machine learning do Aurora é compatível somente com determinadas Regiões da AWS e versões específicas do Aurora PostgreSQL. Antes de tentar configurar o machine learning do Aurora, confira a disponibilidade de sua versão do Aurora PostgreSQL e sua região. Para obter detalhes, consulte Machine learning do Aurora com o Aurora PostgreSQL.

Tópicos

- Requisitos para usar o machine learning do Aurora com o Aurora PostgreSQL
- Recursos compatíveis e limitações do machine learning do Aurora com o Aurora PostgreSQL
- Configurar o cluster de banco de dados do Aurora PostgreSQL para usar machine learning do Aurora
- Usar o Amazon Bedrock com o cluster de banco de dados do Aurora PostgreSQL
- Usar o Amazon Comprehend com o cluster de banco de dados do Aurora PostgreSQL
- Uso do SageMaker AI com o cluster de banco de dados do Aurora PostgreSQL
- Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado)
- Considerações sobre performance para usar o machine learning do Aurora com o Aurora PostgreSQL
- Monitorar o machine learning do Aurora

Requisitos para usar o machine learning do Aurora com o Aurora PostgreSQL

Os serviços do AWS Machine Learning são serviços gerenciados que são configurados e executados em seus próprios ambientes de produção. O machine learning do Aurora é compatível com a integração ao Amazon Comprehend, ao SageMaker AI e ao Amazon Bedrock. Antes de tentar configurar o cluster de banco de dados do Aurora PostgreSQL para usar o machine learning do Aurora, entenda os requisitos e pré-requisitos a seguir.

- Os serviços Amazon Comprehend, SageMaker AI e Amazon Bedrock devem estar em execução na mesma Região da AWS que o cluster de banco de dados do Aurora PostgreSQL. Não é possível usar os serviços Amazon Comprehend, SageMaker AI e Amazon Bedrock de um cluster de banco de dados do Aurora PostgreSQL em uma região diferente.
- Se o cluster de banco de dados do Aurora PostgreSQL estiver em uma nuvem pública virtual (VPC) baseada no serviço Amazon VPC diferente dos serviços Amazon Comprehend e SageMaker AI, o grupo de segurança da VPC precisará permitir conexões de saída com o serviço de machine learning do Aurora de destino. Para obter mais informações, consulte Permitir a comunicação de rede do Amazon Aurora com outros serviços da AWS.
- Para o SageMaker AI, os componentes de machine learning que você deseja usar para inferências devem estar configurados e prontos para uso. Durante o processo de configuração do cluster de

banco de dados do Aurora PostgreSQL, é necessário ter o nome do recurso da Amazon (ARN) do endpoint do SageMaker AI disponível. Os cientistas de dados de sua equipe provavelmente estão mais aptos a trabalhar com o SageMaker AI para preparar os modelos e realizar outras tarefas desse tipo. Para começar a usar o Amazon SageMaker AI, consulte Comece a usar o Amazon SageMaker AI. Para obter mais informações sobre inferências e endpoints, consulte Real-time Inference.

- Para o Amazon Bedrock, é necessário ter o ID dos modelos do Bedrock que você deseja usar para inferências disponíveis durante o processo de configuração do cluster de banco de dados do Aurora PostgreSQL. Os cientistas de dados da equipe provavelmente estão mais aptos a trabalhar com o Bedrock para decidir quais modelos usar, ajustá-los se necessário e realizar outras tarefas desse tipo. Para começar a usar o Amazon Bedrock, consulte How to setup Bedrock.
- Os usuários do Amazon Bedrock precisam solicitar acesso aos modelos antes que eles estejam disponíveis para uso. Se você quiser adicionar modelos adicionais para geração de texto, bate-papo e imagem, precisará solicitar acesso aos modelos no Amazon Bedrock. Para ter mais informações, consulte Model access.

Recursos compatíveis e limitações do machine learning do Aurora com o Aurora PostgreSQL

Atualmente, o machine learning do Aurora é compatível com qualquer endpoint do SageMaker AI que possa ler e gravar no formato de valores separados por vírgula (CSV), por meio de um valor de `ContentType` de `text/csv`. Os formatos incorporados do SageMaker AI que atualmente aceitam esse formato são os seguintes.

- Aprendizagem linear
- Random Cut Forest
- XGBoost

  Para saber mais sobre esses algoritmos, consulte Selecionar um algoritmo no Guia do desenvolvedor do Amazon SageMaker AI.

Ao usar o Amazon Bedrock com o machine learning do Aurora, existem as seguintes limitações:

- As funções definidas pelo usuário (UDFs) oferecem um modo nativo de interagir com o Amazon Bedrock. As UDFs não têm requisitos específicos de solicitação ou resposta, portanto, podem usar qualquer modelo.

- É possível usar UDFs para criar qualquer fluxo de trabalho desejado. Por exemplo, é possível combinar primitivas básicas, como `pg_cron`, executar uma consulta, buscar dados, gerar inferências e gravar em tabelas para atender a consultas diretamente.
- As UDFs não são compatíveis com chamadas em lote ou paralelas.
- A extensão de Machine Learning do Aurora não é compatível com interfaces de vetor. Como parte da extensão, uma função está disponível para gerar as incorporações da resposta do modelo no formato `float8[]` para armazenar essas incorporações no Aurora. Para ter mais informações sobre o uso de `float8[]`, consulte Usar o Amazon Bedrock com o cluster de banco de dados do Aurora PostgreSQL.

Configurar o cluster de banco de dados do Aurora PostgreSQL para usar machine learning do Aurora

Para que o machine learning do Aurora funcione com seu cluster de banco de dados do Aurora PostgreSQL, você precisa criar um perfil do AWS Identity and Access Management (IAM) para cada um dos serviços que você deseja usar. O perfil do IAM permite que seu cluster de banco de dados do Aurora PostgreSQL use o serviço de machine learning do Aurora em nome do cluster. Você também precisa instalar a extensão de machine learning do Aurora. Nos tópicos a seguir, você pode encontrar procedimentos de configuração para cada um desses serviços de machine learning do Aurora.

Tópicos

- Configurar o Aurora PostgreSQL para usar o Amazon Bedrock
- Configurar o Aurora PostgreSQL para usar o Amazon Comprehend
- Configuração do Aurora PostgreSQL para usar o Amazon SageMaker AI
  - Configuração do Aurora PostgreSQL para usar o Amazon S3 para SageMaker AI (avançado)
- Instalar a extensão de machine learning do Aurora

Configurar o Aurora PostgreSQL para usar o Amazon Bedrock

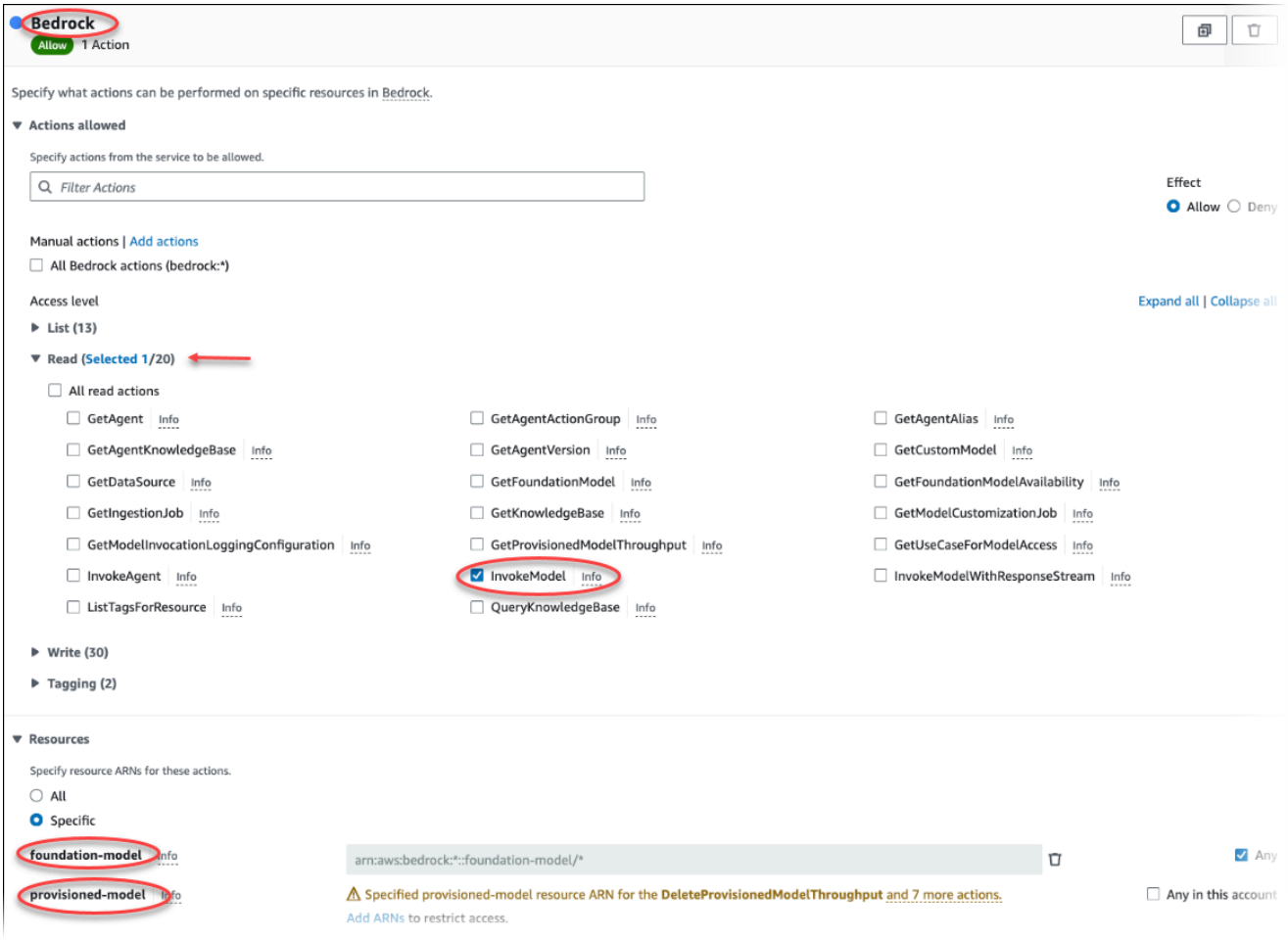
No procedimento a seguir, primeiro crie o perfil do IAM e a política que concede permissão ao Aurora PostgreSQL para usar o Amazon Bedrock em nome do cluster. Depois, anexe a política a um perfil do IAM que o cluster de banco de dados do Aurora PostgreSQL usa para trabalhar com o Amazon Bedrock. Para simplificar, esse procedimento usa o Console de gerenciamento da AWS para concluir todas as tarefas.

Como configurar o cluster de banco de dados do Aurora PostgreSQL para usar o Amazon Bedrock

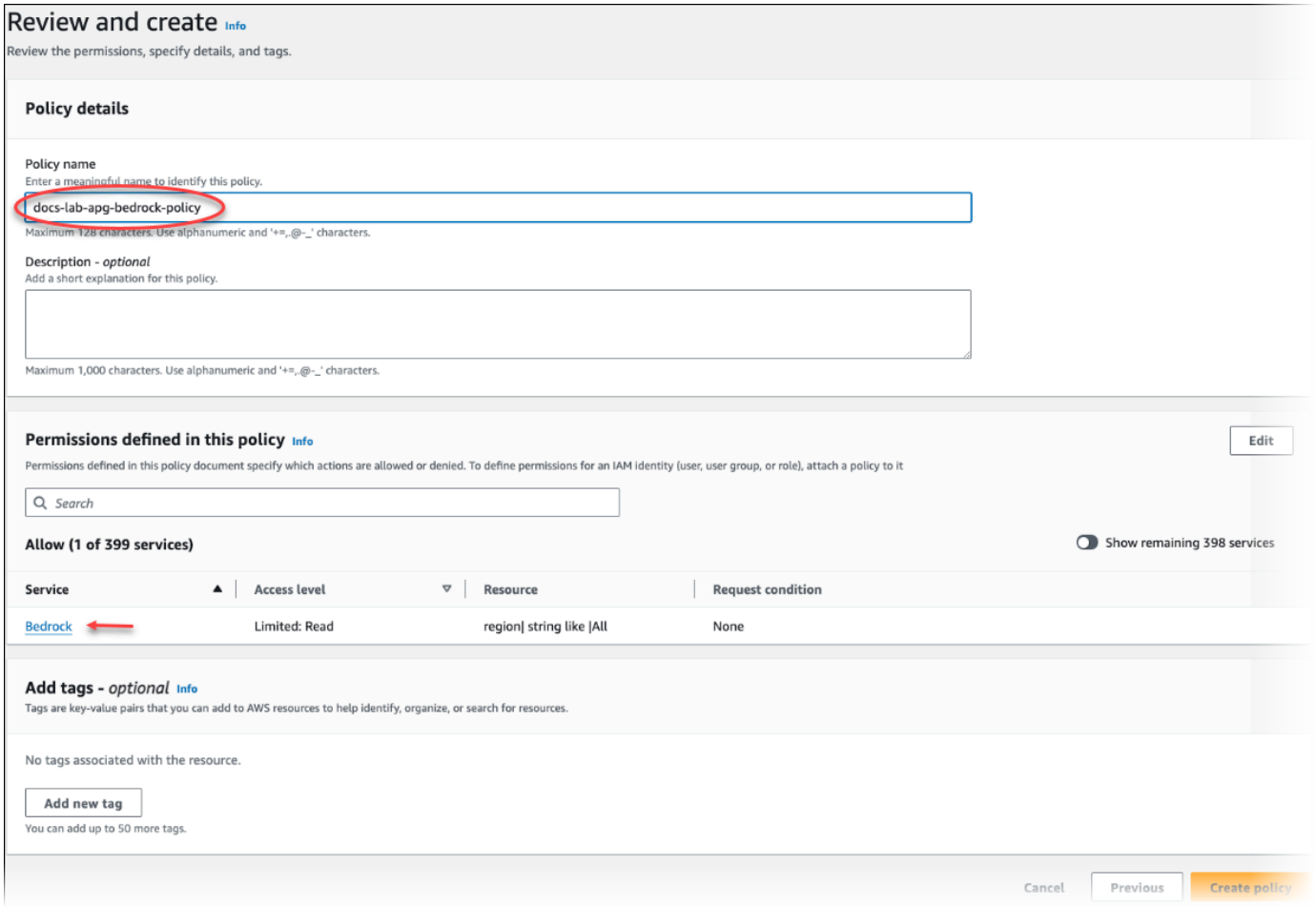
1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Abra o console do IAM em https://console.aws.amazon.com/iam/.
3. Selecione Policies (Políticas) [em Access management (Gerenciamento de acesso)] no menu do console do AWS Identity and Access Management (IAM).

   a. Escolha Criar política. Na página Editor visual, selecione Serviço e, depois, insira Bedrock no campo Selecionar um serviço. Expanda o nível de acesso de leitura. Selecione InvokeModel nas configurações de leitura do Amazon Bedrock.

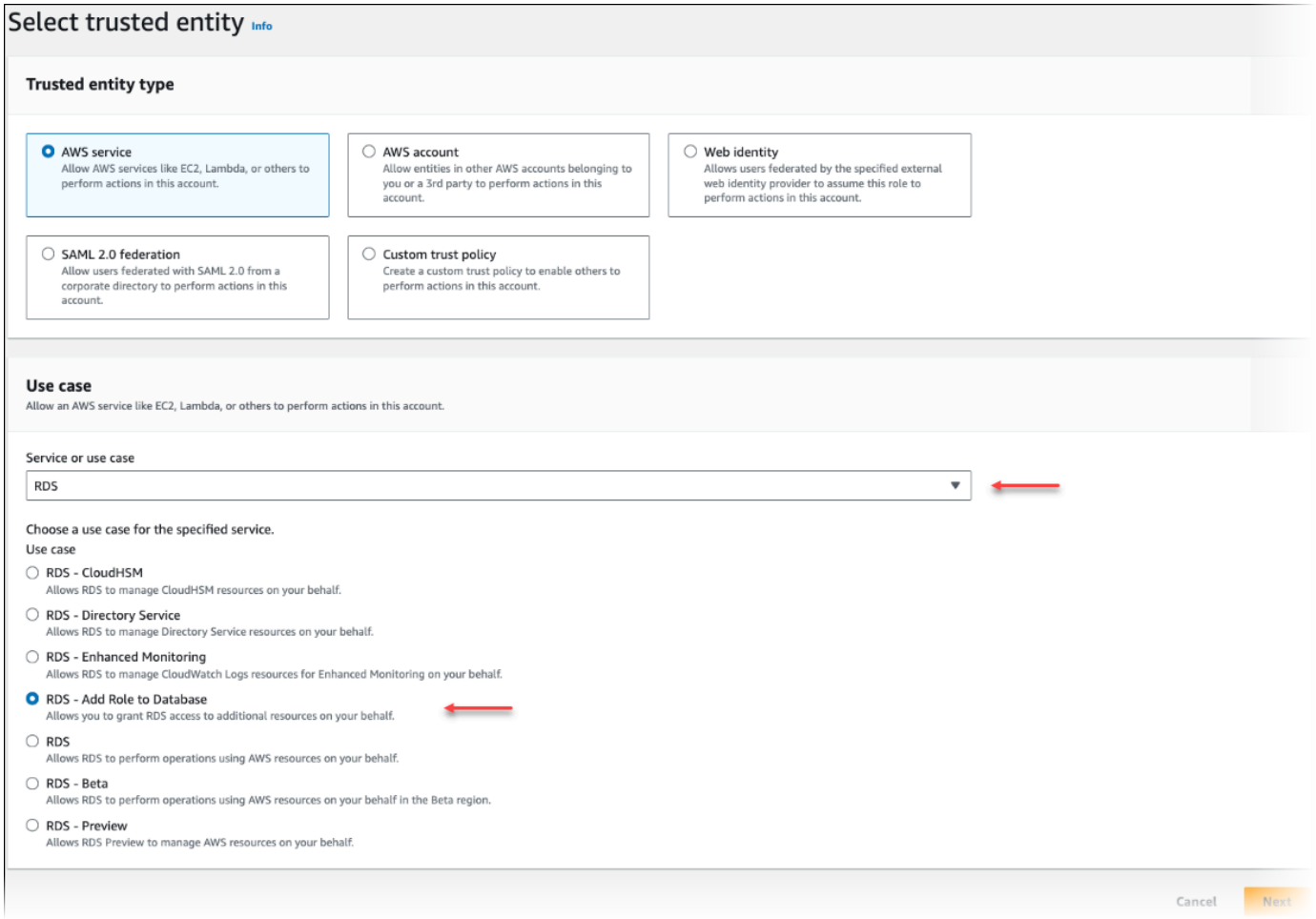
   b. Selecione o modelo de base/provisionado ao qual você deseja conceder acesso de leitura por meio da política.



4. Selecione Next: Tags (Próximo: tags) e defina todas as tags (isso é opcional). Escolha Próximo: revisar. Insira um nome para a política e uma descrição, conforme mostrado na imagem.

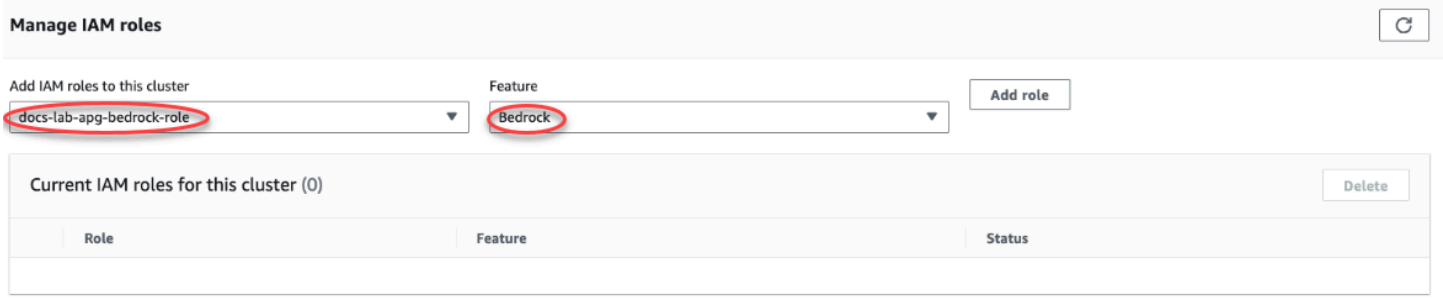


5. Escolha Create policy. O console exibe um alerta quando a política é salva. Você pode encontrá-la na lista de políticas.
6. Selecione Roles (Perfis) [em Access management (Gerenciamento de acesso)] no menu do console do IAM.
7. Selecione Criar perfil.
8. Na página Selecionar entidade confiável, escolha o bloco Serviço da AWS e, depois, selecione RDS para abrir o seletor.
9. Selecione RDS – Add Role to Database (RDS: adicionar função ao banco de dados)



10. Escolha Próximo. Na página Add permissions (Adicionar permissões), localize a política que você criou na etapa anterior e selecione-a entre as listadas. Escolha Próximo.
11. Next: Review (Próximo: revisar. Digite um nome para o perfil do IAM e uma descrição.
12. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
13. Navegue até a Região da AWS onde está seu cluster de banco de dados do Aurora PostgreSQL.
14. No painel de navegação, selecione Bancos de dados e, depois, selecione o cluster de banco de dados do Aurora PostgreSQL que deseja usar com o Bedrock.
15. Selecione a guia Connectivity & security (Conectividade e segurança) e role para baixo até a seção Manage IAM roles (Gerenciar perfis do IAM) da página. Em Add IAM roles to this cluster (Adicionar perfis do IAM a este seletor de cluster), selecione a função que você criou nas etapas anteriores. No seletor Recurso, selecione Bedrock e, depois, Adicionar perfil.

O perfil (com sua política) é associado ao cluster de banco de dados do Aurora PostgreSQL. Quando o processo for concluído, o perfil será exibido na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster), conforme mostrado a seguir.



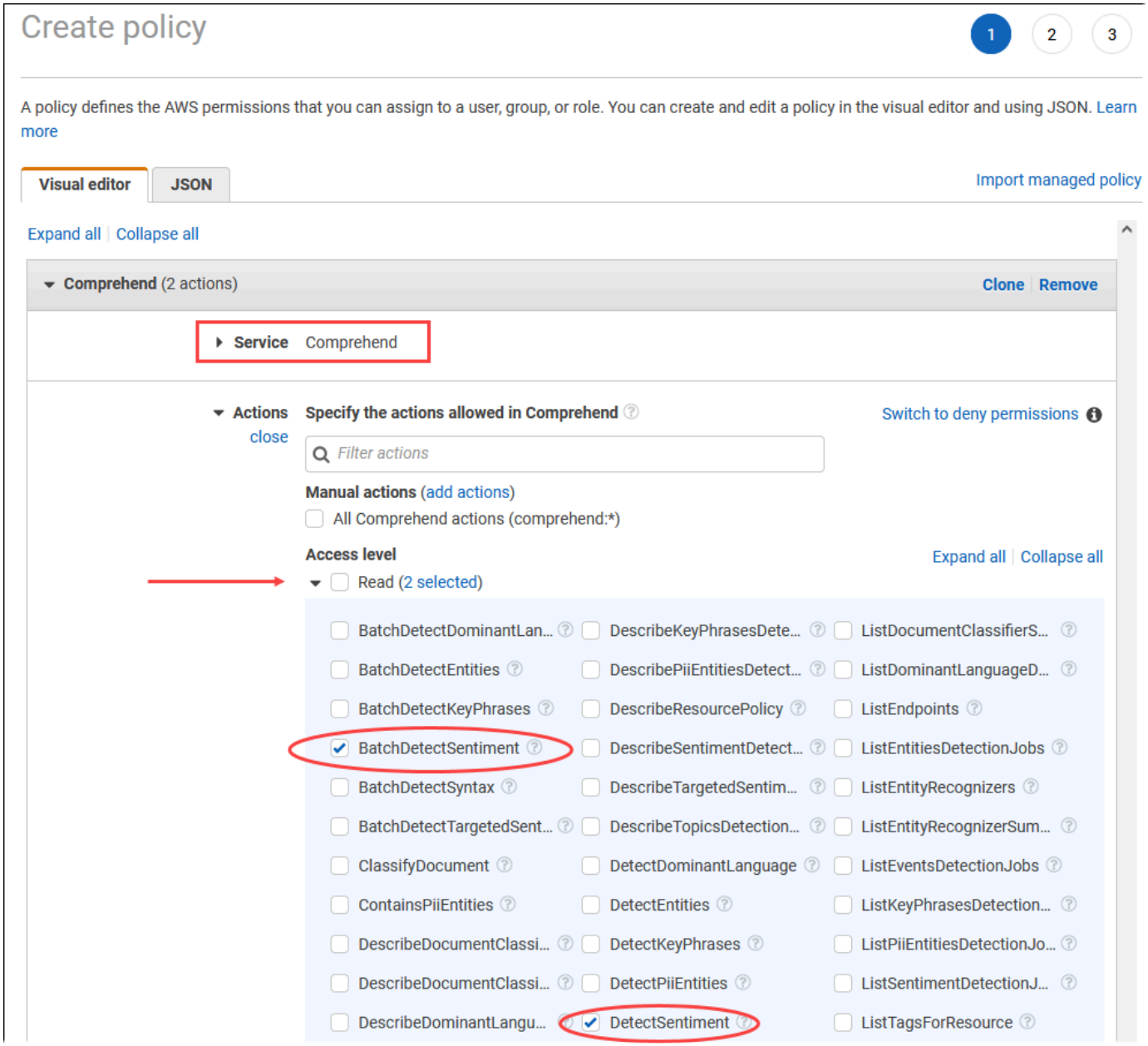
A configuração do IAM para o Amazon Bedrock está concluída. Continue configurando o Aurora PostgreSQL para trabalhar com o machine learning do Aurora instalando a extensão conforme detalhado em Instalar a extensão de machine learning do Aurora

Configurar o Aurora PostgreSQL para usar o Amazon Comprehend

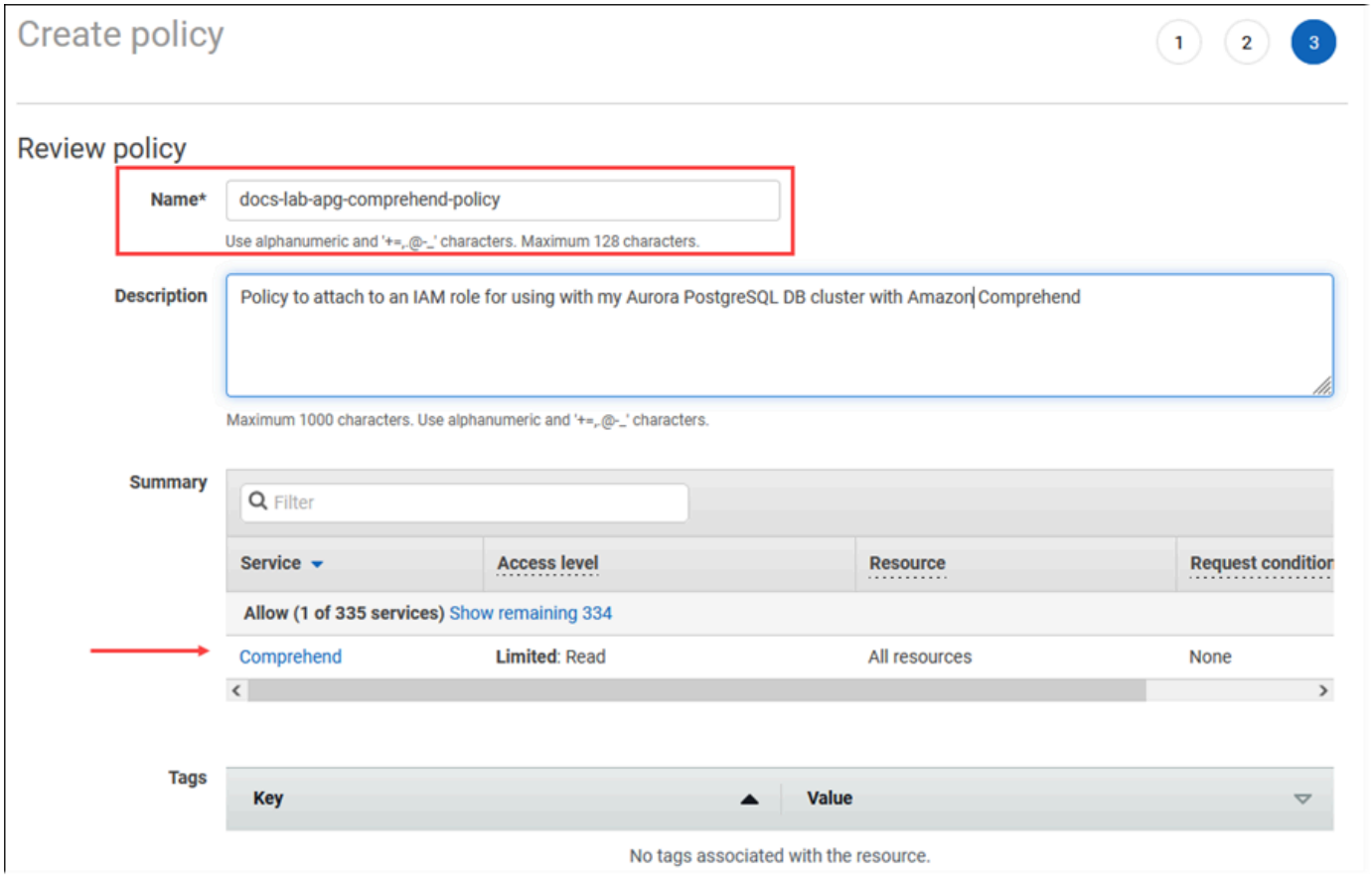
No procedimento a seguir, você primeiro cria o perfil do IAM e a política que concede permissão ao Aurora PostgreSQL para usar o Amazon Comprehend em nome do cluster. Depois, anexe a política a um perfil do IAM que seu cluster de banco de dados do Aurora PostgreSQL usa para trabalhar com o Amazon Comprehend. Para simplificar, esse procedimento usa o Console de gerenciamento da AWS para concluir todas as tarefas.

Como configurar o cluster de banco de dados do Aurora PostgreSQL para usar o Amazon Comprehend

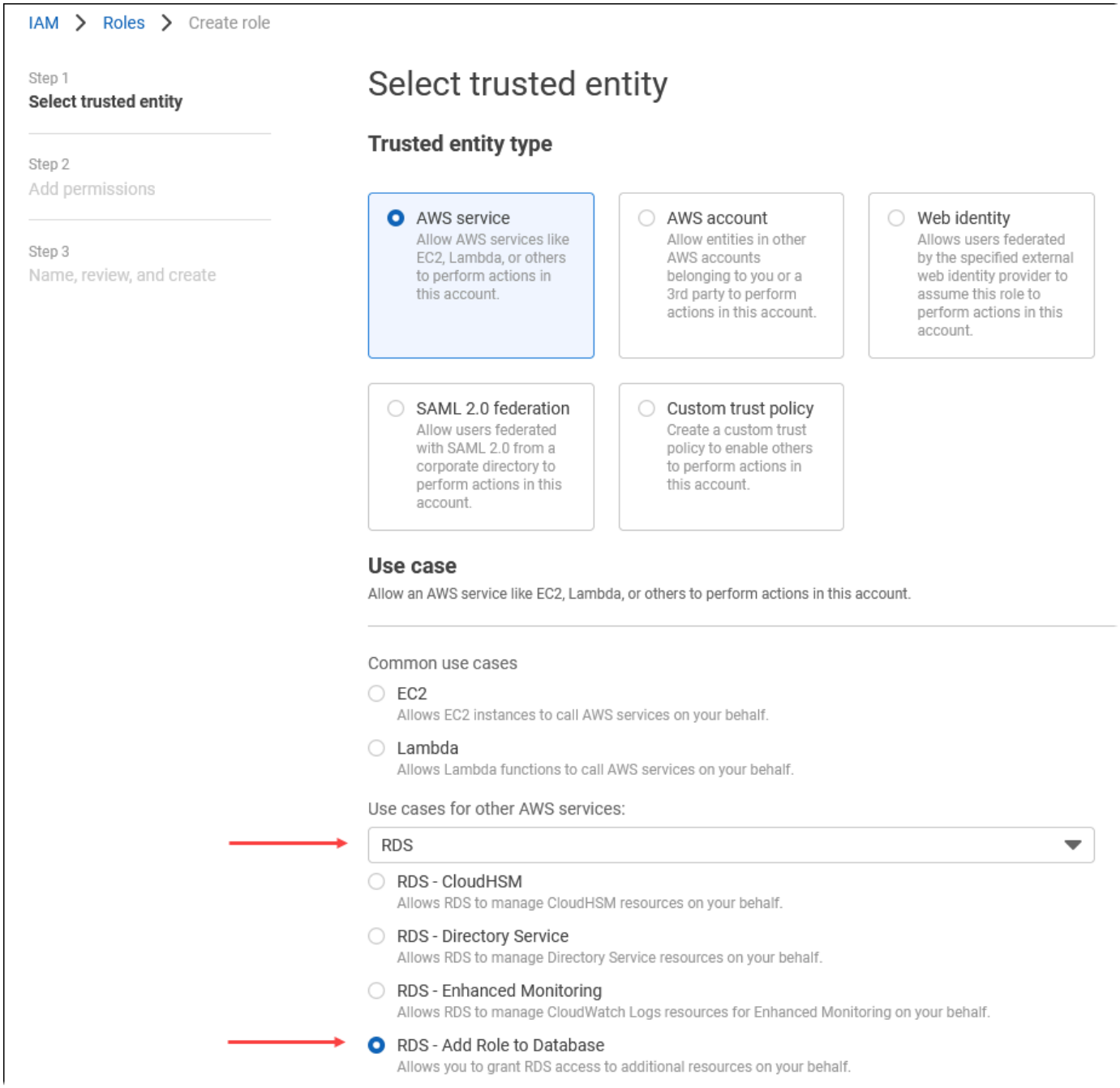
1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Abra o console do IAM em https://console.aws.amazon.com/iam/.
3. Selecione Policies (Políticas) [em Access management (Gerenciamento de acesso)] no menu do console do AWS Identity and Access Management (IAM).



4. Escolha Criar política. Na página Visual editor (Editor visual), selecione Service (Serviço) e depois insira Comprehend no campo Select a service (Selecionar um serviço). Expanda o nível de acesso de leitura. Selecione BatchDetectSentiment e DetectSentiment nas configurações de leitura do Amazon Comprehend.
5. Selecione Next: Tags (Próximo: tags) e defina todas as tags (isso é opcional). Escolha Próximo: revisar. Insira um nome para a política e uma descrição, conforme mostrado na imagem.



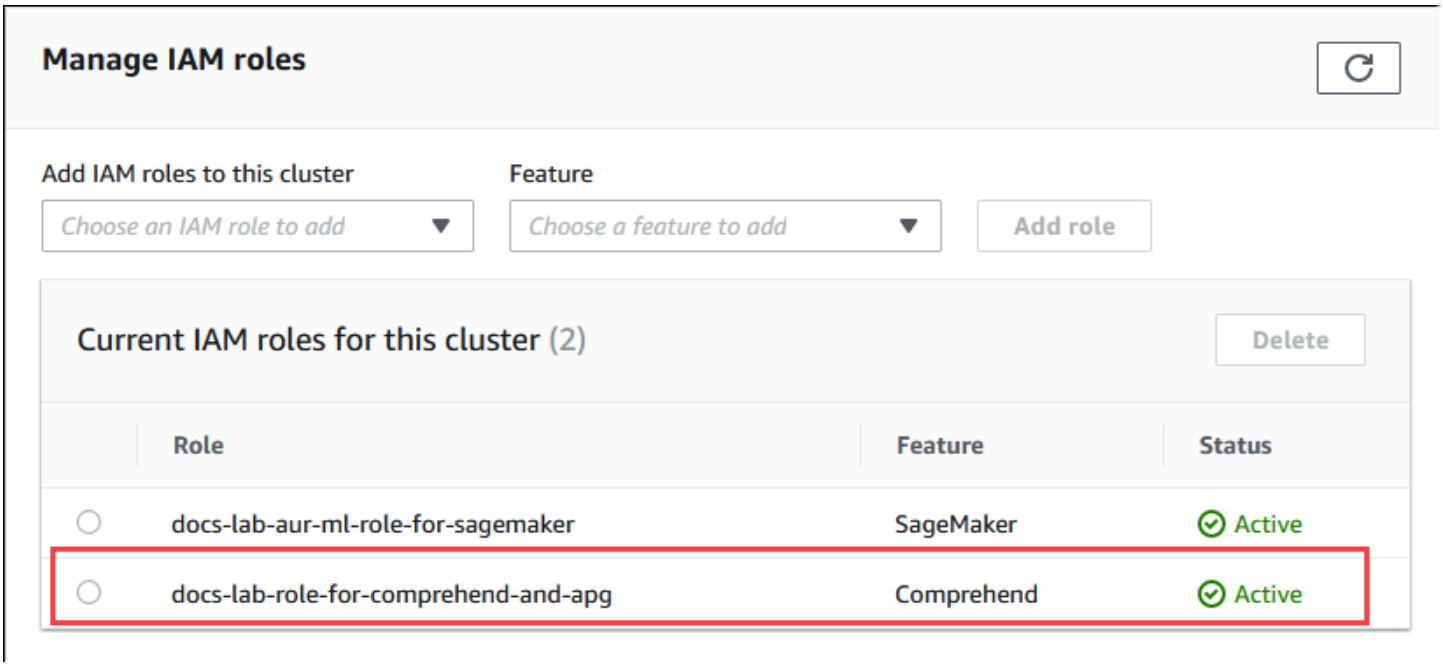
6. Escolha Criar política. O console exibe um alerta quando a política é salva. Você pode encontrá-la na lista de políticas.
7. Selecione Roles (Perfis) [em Access management (Gerenciamento de acesso)] no menu do console do IAM.
8. Selecione Criar perfil.
9. Na página Selecionar entidade confiável, escolha o bloco Serviço da AWS e, depois, selecione RDS para abrir o seletor.
10. Selecione RDS – Add Role to Database (RDS: adicionar função ao banco de dados)



11. Escolha Próximo. Na página Add permissions (Adicionar permissões), localize a política que você criou na etapa anterior e selecione-a entre as listadas. Escolha Next (Próximo).
12. Next: Review (Próximo: revisar. Digite um nome para o perfil do IAM e uma descrição.
13. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
14. Navegue até a Região da AWS onde está seu cluster de banco de dados do Aurora PostgreSQL.

15. No painel de navegação, selecione Databases (Bancos de dados) e selecione o cluster de banco de dados do Aurora PostgreSQL que deseja usar com o Amazon Comprehend.
16. Selecione a guia Connectivity & security (Conectividade e segurança) e role para baixo até a seção Manage IAM roles (Gerenciar perfis do IAM) da página. Em Add IAM roles to this cluster (Adicionar perfis do IAM a este seletor de cluster), selecione a função que você criou nas etapas anteriores. No seletor Recurso, escolha Comprehend e depois Adicionar perfil.

O perfil (com sua política) é associado ao cluster de banco de dados do Aurora PostgreSQL. Quando o processo for concluído, o perfil será exibido na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster), conforme mostrado a seguir.



A integração do IAM para o Amazon Comprehend está concluída. Continue configurando o Aurora PostgreSQL para trabalhar com o machine learning do Aurora instalando a extensão conforme detalhado em Instalar a extensão de machine learning do Aurora

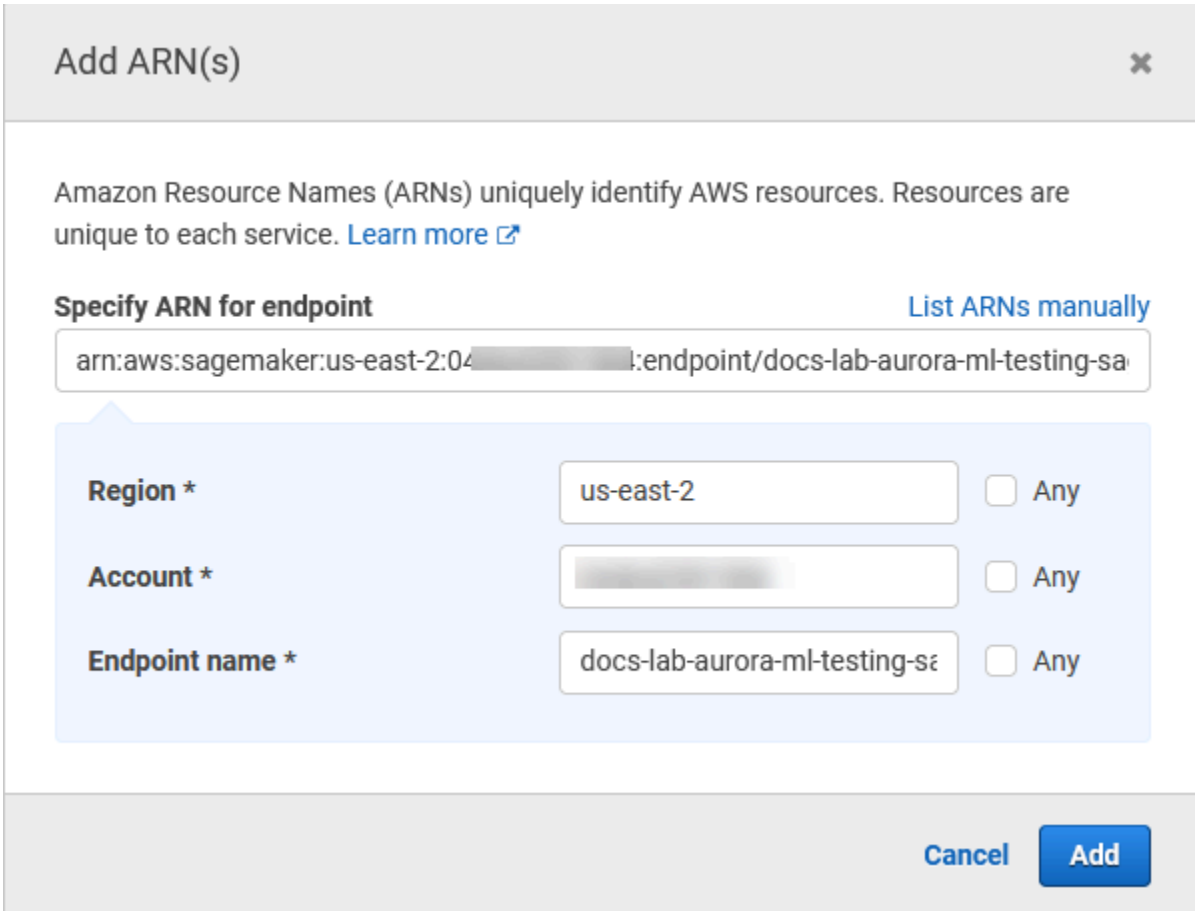
Configuração do Aurora PostgreSQL para usar o Amazon SageMaker AI

Antes de criar a política e o perfil do IAM para seu cluster de banco de dados do Aurora PostgreSQL, você precisa ter seu modelo do SageMaker AI configurado e seu endpoint disponíveis.

Como configurar o cluster de banco de dados do Aurora PostgreSQL para usar o SageMaker AI

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.

2. Selecione Policies (Políticas) [em Access management (Gerenciamento de acesso)] no menu do console do AWS Identity and Access Management (IAM) e depois selecione Create policy (Criar política). No editor visual, selecione SageMaker para o serviço. Em Actions (Ações), abra o seletor de leitura [em Access level (Nível de acesso)] e selecione InvokeEndpoint. Ao fazer isso, um ícone de aviso é exibido.
3. Abra o seletor Resources (Recursos) e clique no link Add ARN to restrict access (Adicionar ARN para restringir o acesso) em Specify endpoint resource ARN (Especificar ARN do recurso de endpoint) para a ação InvokeEndpoint.
4. Insira a Região da AWS dos recursos do SageMaker AI e o nome do endpoint. Sua conta AWS é pré-preenchida.



5. Selecione Add (Adicionar) para salvar. Selecione Next: Tags (Próximo: tags) e Next: Review (Próximo: revisar) para acessar a última página do processo de criação da política.
6. Insira um nome e uma descrição para essa política e depois selecione Create policy (Criar política). A política é criada e adicionada à lista Policies (Políticas). Você vê um alerta no console quando isso ocorre.
7. No console do IAM, selecione Roles (Funções).

8. Selecione Criar perfil.
9. Na página Selecionar entidade confiável, escolha o bloco Serviço da AWS e, depois, selecione RDS para abrir o seletor.
10. Selecione RDS – Add Role to Database (RDS: adicionar função ao banco de dados)
11. Escolha Próximo. Na página Add permissions (Adicionar permissões), localize a política que você criou na etapa anterior e selecione-a entre as listadas. Escolha Next (Próximo).
12. Next: Review (Próximo: revisar. Digite um nome para o perfil do IAM e uma descrição.
13. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
14. Navegue até a Região da AWS onde está seu cluster de banco de dados do Aurora PostgreSQL.
15. No painel de navegação, selecione Bancos de dados e o cluster de banco de dados do Aurora PostgreSQL que deseja usar com o SageMaker AI.
16. Selecione a guia Connectivity & security (Conectividade e segurança) e role para baixo até a seção Manage IAM roles (Gerenciar perfis do IAM) da página. Em Add IAM roles to this cluster (Adicionar perfis do IAM a este seletor de cluster), selecione a função que você criou nas etapas anteriores. No seletor Recurso, selecione SageMaker AI e depois Adicionar perfil.

O perfil (com sua política) é associado ao cluster de banco de dados do Aurora PostgreSQL. Quando o processo for concluído, o perfil será exibido na lista Current IAM roles for this cluster (Perfis atuais do IAM para esse cluster).

A configuração do IAM para o SageMaker AI foi concluída. Continue configurando o Aurora PostgreSQL para trabalhar com o machine learning do Aurora instalando a extensão conforme detalhado em Instalar a extensão de machine learning do Aurora

Configuração do Aurora PostgreSQL para usar o Amazon S3 para SageMaker AI (avançado)

Para usar o SageMaker AI com seus próprios modelos em vez de usar componentes pré-criados fornecidos pelo SageMaker AI, é necessário configurar um bucket do Amazon Simple Storage Service (Amazon S3) para o cluster de banco de dados do Aurora PostgreSQL usar. Esse é um tópico avançado e não está totalmente documentado neste Guia do usuário do Amazon Aurora. O processo geral é o mesmo para integrar o suporte para o SageMaker AI, da seguinte forma.

1. Crie a política e o perfil do IAM para o Amazon S3.

2. Adicione o perfil do IAM e a importação ou a exportação do Amazon S3 como um recurso na guia Connectivity & security (Conectividade e segurança) do cluster de banco de dados do Aurora PostgreSQL.
3. Adicione o ARN do perfil do IAM ao grupo de parâmetros do cluster de banco de dados personalizado para cada cluster de banco de dados do Aurora.

Para obter informações de uso básicas, consulte Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado).

Instalar a extensão de machine learning do Aurora

As extensões de machine learning do Aurora `aws_ml 1.0` fornecem duas funções que você pode usar para invocar os serviços Amazon Comprehend e SageMaker AI e `aws_ml 2.0` fornece duas funções adicionais que você pode usar para invocar os serviços do Amazon Bedrock. A instalação dessas extensões no cluster de banco de dados do Aurora PostgreSQL também cria um perfil administrativo para o recurso.

> **Note**
>
> O uso dessas funções depende da conclusão da configuração do IAM para o serviço de machine learning do Aurora (Amazon Comprehend, SageMaker AI, Amazon Bedrock), conforme detalhado em Configurar o cluster de banco de dados do Aurora PostgreSQL para usar machine learning do Aurora.

- aws_comprehend.detect_sentiment: você usa essa função para aplicar a análise de sentimentos ao texto armazenado no banco de dados do cluster de banco de dados do Aurora PostgreSQL.
- aws_sagemaker.invoke_endpoint: você usa essa função em seu código SQL para se comunicar com o endpoint do SageMaker AI a partir de seu cluster.
- aws_bedrock.invoke_model: use essa função no código SQL para se comunicar com os modelos do Bedrock por meio do cluster. A resposta dessa função será no formato de um TEXTO, portanto, se um modelo responder no formato de um corpo JSON, a saída dessa função será retransmitida no formato de string para o usuário final.
- aws_bedrock.invoke_model_get_embeddings: use essa função no código SQL para invocar modelos do Bedrock que exibem incorporações de saída em uma resposta JSON. Isso pode ser utilizado quando você deseja extrair as incorporações diretamente associadas à chave json para agilizar a resposta com qualquer fluxo de trabalho autogerenciado.

Como instalar a extensão de machine learning do Aurora no cluster de banco de dados do Aurora PostgreSQL

- Use o `psql` para se conectar à instância do gravador do cluster de banco de dados do Aurora PostgreSQL. Conecte-se ao banco de dados específico no qual a extensão `aws_ml` será instalada.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=labdb
```

```
labdb=> CREATE EXTENSION IF NOT EXISTS aws_ml CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
labdb=>
```

A instalação das extensões `aws_ml` também cria a função administrativa `aws_ml` e três esquemas, da forma a seguir.

- `aws_comprehend`: esquema para o serviço Amazon Comprehend e fonte da função `detect_sentiment` (`aws_comprehend.detect_sentiment`).
- `aws_sagemaker`: esquema para o serviço SageMaker AI e fonte da função `invoke_endpoint` (`aws_sagemaker.invoke_endpoint`).
- `aws_bedrock`: esquema para o serviço do Amazon Bedrock e origem das funções `invoke_model(aws_bedrock.invoke_model)` e `invoke_model_get_embeddings(aws_bedrock.invoke_model_get_embeddings)`.

O perfil `rds_superuser` recebe a função administrativa `aws_ml` e torna-se o `OWNER` desses três esquemas de machine learning do Aurora. Para permitir que outros usuários do banco de dados acessem as funções de machine learning do Aurora, o `rds_superuser` precisa conceder privilégios `EXECUTE` nas funções de machine learning do Aurora. Por padrão, os privilégios `EXECUTE` são revogados de `PUBLIC` nas funções nos dois esquemas de machine learning do Aurora.

Em uma configuração de banco de dados multilocatário, você pode impedir que os locatários acessem as funções de machine learning do Aurora usando `REVOKE USAGE` no esquema específico de machine learning do Aurora que você deseja proteger.

Usar o Amazon Bedrock com o cluster de banco de dados do Aurora PostgreSQL

Para o Aurora PostgreSQL, o machine learning do Aurora oferece a função do Amazon Bedrock a seguir para trabalhar com os dados de texto. Essa função está disponível somente após a instalação da extensão `aws_ml` 2.0 e a conclusão de todos os procedimentos de configuração. Para obter mais informações, consulte Configurar o cluster de banco de dados do Aurora PostgreSQL para usar machine learning do Aurora.

aws_bedrock.invoke_model

Essa função usa texto formatado em JSON como entrada, o processa para vários modelos hospedados no Amazon Bedrock e recupera a resposta de texto JSON do modelo. Essa resposta pode conter texto, imagem ou incorporações. Veja um resumo da documentação da função.

```
aws_bedrock.invoke_model(
   IN model_id      varchar,
   IN content_type  text,
   IN accept_type   text,
   IN model_input   text,
   OUT model_output varchar)
```

As entradas e as saídas dessa função são as seguintes.

- `model_id`: identificador do modelo.
- `content_type`: o tipo da solicitação ao modelo do Bedrock.
- `accept_type`: o tipo da resposta que se espera do modelo do Bedrock. Normalmente a aplicação/JSON para a maioria dos modelos.
- `model_input`: prompts; um conjunto específico de entradas para o modelo no formato especificado por content_type. Para ter mais informações sobre o formato/estrutura da solicitação que o modelo aceita, consulte Inference parameters for foundation models.
- `model_output`: a saída do modelo do Bedrock como texto.

O exemplo a seguir mostra como invocar um modelo do Anthropic Claude 2 para o Bedrock usando invoke_model.

Example Exemplo: uma consulta simples utilizando as funções do Amazon Bedrock

```
SELECT aws_bedrock.invoke_model (
    model_id    := 'anthropic.claude-v2',
    content_type:= 'application/json',
    accept_type := 'application/json',
    model_input := '{"prompt": "\n\nHuman: You are a helpful assistant that answers
 questions directly and only using the information provided in the context below.
\nDescribe the answer
    in detail.\n\nContext: %s \n\nQuestion: %s \n
\nAssistant:","max_tokens_to_sample":4096,"temperature":0.5,"top_k":250,"top_p":0.5,"stop_seque
[]}'
);
```

aws_bedrock.invoke_model_get_embeddings

A saída do modelo pode apontar para incorporações de vetores em alguns casos. Como a resposta varia de acordo com o modelo, pode ser utilizada outra função invoke_model_get_embeddings, que funciona exatamente como a invoke_model, mas gera as incorporações especificando a chave json apropriada.

```
aws_bedrock.invoke_model_get_embeddings(
   IN model_id      varchar,
   IN content_type  text,
   IN json_key      text,
   IN model_input   text,
   OUT model_output float8[])
```

As entradas e as saídas dessa função são as seguintes.

- `model_id`: identificador do modelo.
- `content_type`: o tipo da solicitação ao modelo do Bedrock. Aqui, o accept_type é definido como o valor padrão `application/json`.
- `model_input`: prompts; um conjunto específico de entradas para o modelo no formato especificado por content_type. Para ter mais informações sobre o formato/estrutura da solicitação que o modelo aceita, consulte Inference parameters for foundation models.
- `json_key`: referência ao campo do qual extrair a incorporação. Isso pode variar se o modelo de incorporação mudar.

- `model_output`: a saída do modelo do Bedrock como uma matriz de incorporações com decimais de 16 bits.

O exemplo a seguir mostra como gerar uma incorporação usando o modelo de incorporação de texto do Titan Embeddings G1 para a frase de visualizações de monitoramento de E/S do PostgreSQL.

Example Exemplo: uma consulta simples utilizando as funções do Amazon Bedrock

```
SELECT aws_bedrock.invoke_model_get_embeddings(
   model_id      := 'amazon.titan-embed-text-v1',
   content_type  := 'application/json',
   json_key      := 'embedding',
   model_input   := '{ "inputText": "PostgreSQL I/O monitoring views"}') AS embedding;
```

Usar o Amazon Comprehend com o cluster de banco de dados do Aurora PostgreSQL

Para o Aurora PostgreSQL, o machine learning do Aurora fornece a função do Amazon Comprehend a seguir para trabalhar com seus dados de texto. Essa função está disponível somente após a instalação da extensão `aws_ml` e a conclusão de todos os procedimentos de configuração. Para obter mais informações, consulte Configurar o cluster de banco de dados do Aurora PostgreSQL para usar machine learning do Aurora.

aws_comprehend.detect_sentiment

Essa função usa o texto como entrada e avalia se ele tem uma postura emocional positiva, negativa, neutra ou mista. Ela gera esse sentimento junto com um nível de confiança para sua avaliação. Veja um resumo da documentação da função.

```
aws_comprehend.detect_sentiment(
   IN input_text varchar,
   IN language_code varchar,
   IN max_rows_per_batch int,
   OUT sentiment varchar,
   OUT confidence real)
```

As entradas e as saídas dessa função são as seguintes.

- `input_text`: o texto para avaliar e atribuir o sentimento (negativo, positivo, neutro, misto).

- `language_code`: o idioma do `input_text` identificado usando o identificador ISO 639-1 de duas letras com subtag regional (conforme necessário) ou o código ISO 639-2 de três letras, conforme apropriado. Por exemplo, `en` é o código para inglês, `zh` é o código para chinês simplificado. Para obter mais informações, consulte Linguagens compatíveis no Guia do desenvolvedor do Amazon Comprehend.
- `max_rows_per_batch`: o número máximo de linhas por lote para processamento no modo em lote. Para obter mais informações, consulte Noções básicas sobre o modo em lote e as funções de machine learning do Aurora.
- `sentiment`: o sentimento do texto de entrada, identificado como POSITIVO, NEGATIVO, NEUTRO ou MISTO.
- `confidence`: o nível de confiança na precisão do `sentiment` especificado. Os valores variam de 0,0 a 1,0.

Veja exemplos de como usar essa função.

Example Exemplo: uma consulta simples utilizando as funções do Amazon Comprehend

Veja a seguir um exemplo de uma consulta simples que invoca essa função para avaliar a satisfação do cliente com sua equipe de atendimento. Suponha que você tenha uma tabela de banco de dados (`support`) que armazene o feedback do cliente após cada solicitação de ajuda. Essa consulta de exemplo aplica a função `aws_comprehend.detect_sentiment` ao texto na coluna `feedback` da tabela e gera o sentimento e o nível de confiança desse sentimento. Essa consulta também gera resultados em ordem decrescente.

```
SELECT feedback, s.sentiment,s.confidence
    FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
    ORDER BY s.confidence DESC;
 feedback                          | sentiment | confidence
 ------------------------------------------------------------+-----------+------------
 Thank you for the excellent customer support!              | POSITIVE  |   0.999771
 The latest version of this product stinks!                 | NEGATIVE  |   0.999184
 Your support team is just awesome! I am blown away.        | POSITIVE  |   0.997774
 Your product is too complex, but your support is great.    | MIXED     |   0.957958
 Your support tech helped me in fifteen minutes.            | POSITIVE  |   0.949491
 My problem was never resolved!                             | NEGATIVE  |   0.920644
 When will the new version of this product be released?     | NEUTRAL   |   0.902706
 I cannot stand that chatbot.                               | NEGATIVE  |   0.895219
 Your support tech talked down to me.                       | NEGATIVE  |   0.868598
 It took me way too long to get a real person.              | NEGATIVE  |   0.481805
```

```
 (10 rows)
```

Para evitar que você seja cobrado pela detecção de sentimento mais de uma vez por linha da tabela, você pode materializar os resultados. Faça isso nas linhas de interesse. Por exemplo, as anotações do médico estão sendo atualizadas para que somente as em francês (`fr`) usem a função de detecção de sentimento.

```
UPDATE clinician_notes
SET sentiment = (aws_comprehend.detect_sentiment (french_notes, 'fr')).sentiment,
    confidence = (aws_comprehend.detect_sentiment (french_notes, 'fr')).confidence
WHERE
    clinician_notes.french_notes IS NOT NULL AND
    LENGTH(TRIM(clinician_notes.french_notes)) > 0 AND
    clinician_notes.sentiment IS NULL;
```

Para obter mais informações sobre como otimizar suas chamadas de função, consulte Considerações sobre performance para usar o machine learning do Aurora com o Aurora PostgreSQL.

Uso do SageMaker AI com o cluster de banco de dados do Aurora PostgreSQL

Depois de configurar seu ambiente do SageMaker AI e integrá-lo ao Aurora PostgreSQL, conforme descrito em Configuração do Aurora PostgreSQL para usar o Amazon SageMaker AI, você pode invocar as operações utilizando a função `aws_sagemaker.invoke_endpoint`. A função `aws_sagemaker.invoke_endpoint` se conecta somente a um endpoint de modelo na mesma Região da AWS. Se sua instância de banco de dados tiver réplicas em várias Regiões da AWS, configure e implante todos os modelos do SageMaker AI em cada Região da AWS.

As chamadas para `aws_sagemaker.invoke_endpoint` são autenticadas utilizando o perfil do IAM que você configurou para associar o cluster de banco de dados do Aurora PostgreSQL ao serviço SageMaker AI e ao endpoint fornecido durante o processo de configuração. Os endpoints de modelos do SageMaker AI têm o escopo de uma conta individual e não são públicos. A URL `endpoint_name` não contém o ID da conta. O SageMaker AI determina o ID da conta no token de autenticação fornecido pelo perfil do IAM do SageMaker AI da instância do banco de dados.

aws_sagemaker.invoke_endpoint

Essa função usa o endpoint do SageMaker AI como entrada e o número de linhas que devem ser processadas como um lote. Ele também usa como entrada os vários parâmetros esperados pelo endpoint do modelo do SageMaker AI. A documentação de referência dessa função é a seguinte.

```
aws_sagemaker.invoke_endpoint(
  IN endpoint_name varchar,
  IN max_rows_per_batch int,
  VARIADIC model_input "any",
  OUT model_output varchar
  )
```

As entradas e as saídas dessa função são as seguintes.

- `endpoint_name`: um URL de um endpoint que é independente da Região da AWS.
- `max_rows_per_batch`: o número máximo de linhas por lote para processamento no modo em lote. Para obter mais informações, consulte Noções básicas sobre o modo em lote e as funções de machine learning do Aurora.
- `model_input`: um ou mais parâmetros de entrada para o modelo. Esses podem ser qualquer tipo de dados necessário para o modelo do SageMaker AI. O PostgreSQL permite especificar até 100 parâmetros de entrada para uma função. Os tipos de dados de matriz devem ser unidimensionais, mas podem conter tantos elementos quantos forem esperados pelo modelo do SageMaker AI. O número de entradas para um modelo do SageMaker AI é restringido apenas pelo limite de tamanho de mensagem de 6 MB do SageMaker AI.
- `model_output`: a saída do modelo do SageMaker AI como texto.

Criação de uma função definida pelo usuário para chamar um modelo do SageMaker AI

Crie uma função definida pelo usuário separada para chamar `aws_sagemaker.invoke_endpoint` para cada um de seus modelos do SageMaker AI. A função definida pelo usuário representa o endpoint do SageMaker AI que hospeda o modelo. A função `aws_sagemaker.invoke_endpoint` é executada dentro da função definida pelo usuário. As funções definidas pelo usuário oferecem muitas vantagens:

- É possível dar um nome próprio a seu modelo do SageMaker AI em vez de somente chamar `aws_sagemaker.invoke_endpoint` para todos os seus modelos do SageMaker AI.
- É possível especificar o URL do endpoint do modelo em apenas um lugar no código do aplicativo SQL.
- É possível controlar privilégios `EXECUTE` para cada função de machine learning do Aurora de forma independente.
- É possível declarar os tipos de entrada e saída do modelo usando tipos SQL. O SQL impõe o número e o tipo de argumentos transmitidos ao modelo do SageMaker AI e executa a conversão de tipo, se necessário. O uso de tipos SQL também converterá `SQL NULL` no valor padrão apropriado esperado pelo modelo do SageMaker AI.
- Você pode reduzir o tamanho máximo do lote se quiser retornar as primeiras linhas um pouco mais rápido.

Para especificar uma função definida pelo usuário, use a instrução da linguagem de definição de dados (DDL) do SQ `CREATE FUNCTION`. Ao definir a função, você especifica o seguinte:

- Os parâmetros de entrada para o modelo.
- O endpoint específico do SageMaker AI a ser invocado.
- O tipo de retorno.

A função definida pelo usuário retorna a inferência computada pelo endpoint do SageMaker AI após a execução do modelo com os parâmetros de entrada. O exemplo a seguir cria uma função definida pelo usuário para um modelo do SageMaker AI com dois parâmetros de entrada.

```
CREATE FUNCTION classify_event (IN arg1 INT, IN arg2 DATE, OUT category INT)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', NULL,
        arg1, arg2                          -- model inputs are separate arguments
        )::INT                              -- cast the output to INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Observe o seguinte:

- A entrada da função `aws_sagemaker.invoke_endpoint` pode ser um ou mais parâmetros de qualquer tipo de dados.

- Este exemplo usa um tipo de saída INT. Se você converter a saída de um tipo `varchar` em outro tipo, ela deverá ser convertida para um tipo escalar integrado do PostgreSQL, como `INTEGER`, `REAL`, `FLOAT` ou `NUMERIC`. Para obter mais informações sobre esses tipos, consulte Tipos de dados na documentação do PostgreSQL.
- Especifique `PARALLEL SAFE` para habilitar o processamento da consulta paralela. Para obter mais informações, consulte Melhorar os tempos de resposta com o processamento de consultas paralelas.
- Especifique `COST 5000` para estimar o custo da execução da função. Use um número positivo dando o custo de execução estimado para a função, em unidades de `cpu_operator_cost`.

Transmissão de uma matriz como entrada para um modelo do SageMaker AI

A função `aws_sagemaker.invoke_endpoint` pode ter até 100 parâmetros de entrada, que é o limite para funções do PostgreSQL. Se o modelo do SageMaker AI exigir mais de 100 parâmetros do mesmo tipo, passe os parâmetros do modelo como uma matriz.

O exemplo a seguir define uma função que transmite uma matriz como entrada para o modelo de regressão do SageMaker AI. A saída é convertida em um valor `REAL`.

```
CREATE FUNCTION regression_model (params REAL[], OUT estimate REAL)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
      'sagemaker_model_endpoint_name',
      NULL,
      params
      )::REAL
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Especificação do tamanho do lote ao chamar um modelo do SageMaker AI

O exemplo a seguir cria uma função definida pelo usuário para um modelo do SageMaker AI que define o padrão do tamanho do lote como NULL. A função também permite fornecer um tamanho de lote diferente ao chamá-la.

```
CREATE FUNCTION classify_event (
    IN event_type INT, IN event_day DATE, IN amount REAL, -- model inputs
    max_rows_per_batch INT DEFAULT NULL,  -- optional batch size limit
    OUT category INT)                     -- model output
```

```
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', max_rows_per_batch,
        event_type, event_day, COALESCE(amount, 0.0)
        )::INT              -- casts output to type INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Observe o seguinte:

- Use o parâmetro opcional `max_rows_per_batch` para fornecer controle sobre o número de linhas para uma chamada de função em modo de lote. Se você usar um valor de NULL, o otimizador de consulta escolherá automaticamente o tamanho máximo do lote. Para obter mais informações, consulte Noções básicas sobre o modo em lote e as funções de machine learning do Aurora.
- Por padrão, passar NULL como o valor de um parâmetro é convertido em uma string vazia antes de ser passado para o SageMaker AI. Para este exemplo, as entradas têm tipos diferentes.
- Se você tiver uma entrada que não seja de texto ou uma entrada de texto que precise ser padronizada para um valor diferente de uma string vazia, use a instrução `COALESCE`. Use `COALESCE` para converter NULL no valor de substituição nulo desejado na chamada para `aws_sagemaker.invoke_endpoint`. Para o parâmetro `amount` neste exemplo, um valor NULL é convertido em 0.0.

Invocação de um modelo do SageMaker AI que tem várias saídas

O exemplo a seguir cria uma função definida pelo usuário para um modelo do SageMaker AI que retorna várias saídas. A função precisa converter a saída da função `aws_sagemaker.invoke_endpoint` em um tipo de dados correspondente. Por exemplo, você pode usar o tipo de ponto integrado do PostgreSQL para pares (x, y) ou um tipo composto definido pelo usuário.

Essa função definida pelo usuário retorna valores de um modelo que retorna várias saídas usando um tipo composto para as saídas.

```
CREATE TYPE company_forecasts AS (
    six_month_estimated_return real,
    one_year_bankruptcy_probability float);
CREATE FUNCTION analyze_company (
    IN free_cash_flow NUMERIC(18, 6),
    IN debt NUMERIC(18,6),
```

```
    IN max_rows_per_batch INT DEFAULT NULL,
    OUT prediction company_forecasts)
AS $$
    SELECT (aws_sagemaker.invoke_endpoint('endpt_name',
       max_rows_per_batch,free_cash_flow, debt))::company_forecasts;


$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Para o tipo composto, use campos na mesma ordem como aparecem na saída do modelo e converta a saída de `aws_sagemaker.invoke_endpoint` para o tipo composto. O chamador pode extrair os campos individuais por nome ou com a notação ".*" do PostgreSQL.

Exportar dados ao Amazon S3 para treinamento de modelos do SageMaker AI (avançado)

Recomendamos que você se familiarize com o machine learning do Aurora e o SageMaker AI utilizando os algoritmos e os exemplos fornecidos em vez de tentar treinar seus próprios modelos. Para obter mais informações, consulte Conceitos básicos do Amazon SageMaker AI.

Para treinar modelos do SageMaker AI, exporte os dados para um bucket do Amazon S3. O bucket do Amazon S3 é usado pelo SageMaker AI para treinar seu modelo antes que ele seja implantado. Você pode consultar dados de um cluster de banco de dados do Aurora PostgreSQL e salvá-los diretamente em arquivos de texto armazenados em um bucket do Amazon S3. O SageMaker AI consome os dados do bucket do Amazon S3 para treinamento. Para obter mais informações sobre o treinamento de modelos do SageMaker AI, consulte Treinar um modelo com o Amazon SageMaker AI.

> **Note**
>
> Quando você cria um bucket do Amazon S3 para treinamento de modelos do SageMaker AI ou para pontuação em lotes, use `sagemaker` no nome do bucket do Amazon S3. Para obter mais informações, consulte Especificar um bucket do Amazon S3 para fazer upload de conjuntos de dados de treinamento e armazenar dados de saída no Guia do desenvolvedor do Amazon SageMaker AI.

Para obter mais informações sobre como exportar os dados, consulte Exportar dados de um cluster de banco de dados do Aurora PostgreSQL para o Amazon S3.

Considerações sobre performance para usar o machine learning do Aurora com o Aurora PostgreSQL

Os serviços do Amazon Comprehend e do SageMaker AI fazem a maior parte do trabalho quando invocados por uma função de machine learning do Aurora. Isso significa que você pode escalar esses recursos conforme necessário, de forma independente. Para o cluster de banco de dados do Aurora PostgreSQL, você pode tornar suas chamadas de função o mais eficientes possível. A seguir, você encontrará algumas considerações sobre performance a serem observadas ao trabalhar com o machine learning do Aurora no Aurora PostgreSQL.

Tópicos

- Noções básicas sobre o modo em lote e as funções de machine learning do Aurora
- Melhorar os tempos de resposta com o processamento de consultas paralelas
- Usar visualizações materializadas e colunas materializadas

Noções básicas sobre o modo em lote e as funções de machine learning do Aurora

Normalmente, o PostgreSQL executa as funções uma linha por vez. O machine learning do Aurora pode reduzir essa sobrecarga, combinando as chamadas ao serviço externo de machine learning do Aurora para várias linhas em lotes com uma abordagem chamada de execução no modo em lotes. No modo em lotes, o machine learning do Aurora recebe as respostas para um lote de linhas de entrada e as entrega de volta para a consulta em execução, uma linha de cada vez. Essa otimização melhora a taxa de transferência de suas consultas do Aurora sem limitar o otimizador de consultas PostgreSQL.

O Aurora usará automaticamente o modo de lote se a função for referenciada na lista `SELECT`, em uma cláusula `WHERE` ou `HAVING`. Observe que as expressões `CASE` simples de nível superior são qualificadas para execução em modo de lote. As expressões `CASE` pesquisadas em nível superior também são qualificadas para execução em modo de lote, desde que a primeira cláusula `WHEN` seja um predicado simples com uma chamada de função em modo de lote.

Sua função definida pelo usuário deve ser uma função `LANGUAGE SQL` e deve especificar `PARALLEL SAFE` e `COST 5000`.

Migração de função da instrução SELECT para a cláusula FROM

Normalmente, uma função do `aws_ml` que é qualificada para execução em modo de lote é migrada automaticamente pelo Aurora para a cláusula `FROM`.

A migração de funções em modo de lote qualificadas para a cláusula FROM pode ser examinada manualmente em um nível por consulta. Para fazer isso, use instruções EXPLAIN (e ANALYZE e VERBOSE) e encontre as informações de “Batch processing (Processamento em lote)” abaixo de cada em modo de lot `Function Scan`. Você também pode usar EXPLAIN (com VERBOSE) sem executar a consulta. Observe se as chamadas para a função aparecem como um `Function Scan` sob uma junção de loop aninhado que não foi especificado na instrução original.

No exemplo a seguir, o operador de junção de loop aninhado no plano mostra que o Aurora migrou a função `anomaly_score`. Ele migrou essa função da lista de SELECT para a cláusula FROM, onde é elegível para execução em modo de lote.

```
EXPLAIN (VERBOSE, COSTS false)
SELECT anomaly_score(ts.R.description) from ts.R;
                          QUERY PLAN
---------------------------------------------------------------
 Nested Loop
   Output: anomaly_score((r.description)::text)
   ->  Seq Scan on ts.r
         Output: r.id, r.description, r.score
   ->  Function Scan on public.anomaly_score
         Output: anomaly_score.anomaly_score
         Function Call: anomaly_score((r.description)::text)
```

Para desabilitar a execução em modo de lote, defina o parâmetro `apg_enable_function_migration` como `false`. Isso impede a migração de funções do `aws_ml` de SELECT para a cláusula `FROM`. Veja a seguir como fazer isso.

```
SET apg_enable_function_migration = false;
```

O parâmetro `apg_enable_function_migration` é um parâmetro GUC (Grand Unified Configuration) reconhecido pela extensão `apg_plan_mgmt` do Aurora PostgreSQL para gerenciamento do plano de consulta. Para desabilitar a migração de funções em uma sessão, use o gerenciamento de plano de consulta para salvar o plano resultante como um plano `approved`. Em tempo de execução, o gerenciamento de plano de consulta impõe o plano `approved` com sua configuração `apg_enable_function_migration`. Essa imposição ocorre independentemente da configuração do parâmetro `apg_enable_function_migration`. Para obter mais informações, consulte Gerenciar planos de execução de consultas do Aurora PostgreSQL.

Usar o parâmetro max_rows_per_batch

As funções `aws_comprehend.detect_sentiment` e `aws_sagemaker.invoke_endpoint` têm um parâmetro `max_rows_per_batch`. Esse parâmetro especifica o número de linhas que podem ser enviadas ao serviço de machine learning do Aurora. Quanto maior for o conjunto de dados processado pela função, maior será o tamanho do lote que poderá ser criado.

Funções no modo em lote melhoram a eficiência, criando lotes de linhas que distribuem o custo das chamadas de função de machine learning do Aurora por um grande número de linhas. No entanto, se uma instrução `SELECT` terminar cedo devido a uma cláusula `LIMIT`, o lote poderá ser construído sobre mais linhas do que as usadas pela consulta. Essa abordagem pode resultar em cobranças adicionais em sua conta da AWS. Para obter os benefícios da execução em modo de lote, mas evitar a criação de lotes que são muito grandes, use um valor menor para o parâmetro `max_rows_per_batch` em suas chamadas de função.

Se você fizer um `EXPLAIN (VERBOSE, ANALYZE)` de uma consulta que usa execução em modo de lote, verá um operador de `FunctionScan` abaixo de uma junção de loop aninhado. O número de loops relatados por `EXPLAIN` equivale ao número de vezes que uma linha foi obtida pelo operador `FunctionScan`. Se uma instrução usar uma cláusula LIMIT, o número de buscas será consistente. Para otimizar o tamanho do lote, defina o parâmetro `max_rows_per_batch` como esse valor. No entanto, se a função em modo de lote for referenciada em um predicado na cláusula `WHERE` ou `HAVING`, você provavelmente não poderá saber o número de buscas antecipadamente. Nesse caso, use os loops como uma diretriz e experimente com `max_rows_per_batch` para encontrar uma configuração que otimize a performance.

Verificar a execução em modo de lote

Para ver se uma função foi executada no modo em lote, use `EXPLAIN ANALYZE`. Se a execução em modo de lote foi usada, o plano de consulta incluirá as informações em uma seção “Batch Processing (Processamento em lote)”.

```
EXPLAIN ANALYZE SELECT user-defined-function();
 Batch Processing: num batches=1 avg/min/max batch size=3333.000/3333.000/3333.000
                                 avg/min/max batch call time=146.273/146.273/146.273
```

Neste exemplo, havia um lote que continha 3.333 linhas, que levou 146,273 ms para ser processado. A seção “Batch Processing (Processamento em lote)” mostra o seguinte:

- Quantos lotes havia para a operação de verificação da função

- O tamanho médio, mínimo e máximo do lote
- O tempo médio, mínimo e máximo de execução do lote

Normalmente, o lote final é menor do que o restante, o que geralmente resulta em um tamanho mínimo de lote que é muito menor do que a média.

Para retornar as primeiras linhas mais rapidamente, defina o parâmetro `max_rows_per_batch` como um valor menor.

Para reduzir o número de chamadas em modo de lote para o serviço de ML ao usar um `LIMIT` na função definida pelo usuário, defina o parâmetro `max_rows_per_batch` como um valor menor.

Melhorar os tempos de resposta com o processamento de consultas paralelas

Para obter resultados o mais rápido possível de um grande número de linhas, é possível combinar o processamento de consultas paralelas com o processamento no modo em lote. Você pode usar o processamento de consultas paralelas para instruções `SELECT`, `CREATE TABLE AS SELECT` e `CREATE MATERIALIZED VIEW`.

> **Note**
>
> O PostgreSQL ainda não é compatível com consultas paralelas para instruções de linguagem de manipulação de dados (DML).

O processamento de consultas paralelas ocorre no banco de dados e dentro do serviço de ML. O número de núcleos na classe da instância de banco de dados limita o nível de paralelismo que pode ser usado ao executar uma consulta. O servidor de banco de dados pode construir um plano de execução de consulta paralela que particiona a tarefa entre um conjunto de operadores paralelos. Cada um desses operadores pode criar solicitações em lote contendo dezenas de milhares de linhas (ou quantas linhas cada serviço permitir).

As solicitações em lote de todos os operadores paralelos são enviadas ao endpoint do SageMaker AI. O nível de paralelismo com o qual o endpoint é compatível é limitado pelo número e pelo tipo de instâncias que o suportam. Para K graus de paralelismo, você precisa de uma classe de instância de banco de dados que tenha pelo menos núcleos K. Você também precisa configurar o endpoint do SageMaker AI para que seu modelo tenha K instâncias iniciais de uma classe de instância de performance suficientemente alta.

Para usar o processamento de consultas paralelas, você pode definir o parâmetro de armazenamento `parallel_workers` da tabela que contém os dados que você planeja passar. Você define `parallel_workers` para uma função em modo de lote como `aws_comprehend.detect_sentiment`. Se o otimizador escolher um plano de consulta paralelo, os serviços de ML da AWS podem ser chamados em lote e em paralelo.

Você pode usar os seguintes parâmetros com a função `aws_comprehend.detect_sentiment` para obter um plano com paralelismo de quatro vias. Se você alterar qualquer um dos dois parâmetros a seguir, deverá reiniciar a instância de banco de dados para que as alterações tenham efeito

```
-- SET max_worker_processes to 8;  -- default value is 8
-- SET max_parallel_workers to 8;  -- not greater than max_worker_processes
SET max_parallel_workers_per_gather to 4;  -- not greater than max_parallel_workers

-- You can set the parallel_workers storage parameter on the table that the data
-- for the Aurora machine learning function is coming from in order to manually
 override the degree of
-- parallelism that would otherwise be chosen by the query optimizer
--
ALTER TABLE yourTable SET (parallel_workers = 4);

-- Example query to exploit both batch-mode execution and parallel query
EXPLAIN (verbose, analyze, buffers, hashes)
SELECT aws_comprehend.detect_sentiment(description, 'en')).*
FROM yourTable
WHERE id < 100;
```

Para obter mais informações sobre como controlar consultas paralelas, consulte Parallel plans (Planos paralelos) na documentação do PostgreSQL.

Usar visualizações materializadas e colunas materializadas

Ao chamar um serviço da AWS, como o SageMaker AI ou o Amazon Comprehend, do banco de dados, sua conta é cobrada de acordo com a política de preço desse serviço. Para minimizar as cobranças em sua conta, é possível materializar o resultado da chamada do serviço da AWS em uma coluna materializada para que o serviço da AWS não seja chamado mais de uma vez por linha de entrada. Se desejar, você pode adicionar uma coluna de timestamp `materializedAt` para registrar a hora em que as colunas foram materializadas.

A latência de uma `INSERT` instrução de linha única comum geralmente é muito menor do que a latência de chamar uma função em modo de lote. Assim, talvez você não consiga atender aos requisitos de latência do seu aplicativo se invocar a função em modo de lote para cada linha única `INSERT` executada pelo aplicativo. Para materializar o resultado da chamada de um serviço da AWS em uma coluna materializada, os aplicativos de alto performance geralmente precisam preencher as colunas materializadas. Para fazer isso, eles emitem periodicamente uma instrução `UPDATE` que opera em um grande lote de linhas ao mesmo tempo.

`UPDATE`A usa um bloqueio em nível de linha que pode afetar um aplicativo em execução. Portanto, você pode precisar usar `SELECT ... FOR UPDATE SKIP LOCKED` ou `MATERIALIZED VIEW`.

As consultas de análise que operam em um grande número de linhas em tempo real podem combinar a materialização no modo em lote com o processamento em tempo real. Para fazer isso, essas consultas montam uma `UNION ALL` dos resultados pré-materializados com uma consulta das linhas que ainda não têm resultados materializados. Em alguns casos, essa `UNION ALL` é necessária em vários locais, ou a consulta é gerada por um aplicativo de terceiros. Nesse caso, você pode criar uma `VIEW` para encapsular a operação `UNION ALL` para que esse detalhe não seja exposto ao restante do aplicativo SQL.

Você pode usar uma visualização materializada para materializar os resultados de uma instrução `SELECT` arbitrária em um snapshot no tempo. Você também pode usá-la para atualizar a visualização materializada a qualquer momento no futuro. Atualmente, o PostgreSQL é compatível com a atualização incremental, portanto, cada vez que a visualização materializada é atualizada, a visualização materializada é totalmente recalculada.

Você pode atualizar visualizações materializadas com a opção `CONCURRENTLY`, que atualiza o conteúdo da visualização materializada sem usar um bloqueio exclusivo. Isso permite que um aplicativo SQL leia a visualização materializada enquanto ela está sendo atualizada.

Monitorar o machine learning do Aurora

Você pode monitorar as funções `aws_ml` definindo o parâmetro `track_functions` em seu grupo de parâmetros de cluster de banco de dados personalizado como `all`. Por padrão, esse parâmetro é definido como `pl`, o que significa que somente as funções da linguagem do procedimento são monitoradas. Ao alterar isso para `all`, as funções `aws_ml` também são monitoradas. Para obter mais informações, consulte Run-time Statistics (Estatísticas de runtime) na documentação do PostgreSQL.

Para obter informações sobre como monitorar a performance das operações do SageMaker chamadas em funções do machine learning do Aurora, consulte Monitorar o Amazon SageMaker AI no Guia do desenvolvedor do Amazon SageMaker AI.

Com `track_functions` definido como `all`, você pode consultar a visualização `pg_stat_user_functions` para obter estatísticas sobre as funções que você define e usa para invocar os serviços de machine learning do Aurora. Para cada função, a visualização fornece o número de `calls`, `total_time` e `self_time`.

Para ver as estatísticas das funções `aws_sagemaker.invoke_endpoint` e `aws_comprehend.detect_sentiment`, você pode filtrar os resultados pelo nome do esquema utilizando a consulta a seguir.

```
SELECT * FROM pg_stat_user_functions
    WHERE schemaname
    LIKE 'aws_%';
```

Para limpar as estatísticas, faça o seguinte.

```
SELECT pg_stat_reset();
```

É possível obter os nomes das funções SQL que chamam a função `aws_sagemaker.invoke_endpoint` consultando o catálogo do sistema `pg_proc` do PostgreSQL. Esse catálogo armazena informações sobre funções, procedimentos e muito mais. Para obter mais informações, consulte pg_proc na documentação do PostgreSQL. Veja a seguir um exemplo de consulta à tabela para obter os nomes das funções (`proname`) cuja fonte (`prosrc`) inclui o texto invoke_endpoint.

```
SELECT proname FROM pg_proc WHERE prosrc LIKE '%invoke_endpoint%';
```

Exemplos de código para o Aurora usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Aurora com um kit de desenvolvimento de software (SDK) da AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para ver uma lista completa dos Guias do desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- Exemplos de código para o Aurora usando SDKs da AWS
  - Olá, Aurora
  - Conheça os conceitos básicos do Aurora com um SDK da AWS
  - Ações para o Aurora usando AWS SDKs
    - Usar CreateDBCluster com um SDK da AWS
    - Usar CreateDBClusterParameterGroup com um SDK da AWS
    - Usar CreateDBClusterSnapshot com um SDK da AWS
    - Usar CreateDBInstance com um SDK da AWS
    - Usar DeleteDBCluster com um SDK da AWS
    - Usar DeleteDBClusterParameterGroup com um SDK da AWS
    - Usar DeleteDBInstance com um SDK da AWS
    - Usar DescribeDBClusterParameterGroups com um SDK da AWS
    - Usar DescribeDBClusterParameters com um SDK da AWS
    - Usar DescribeDBClusterSnapshots com um SDK da AWS
    - Usar DescribeDBClusters com um SDK da AWS

    - Usar DescribeDBEngineVersions com um SDK da AWS
    - Usar DescribeDBInstances com um SDK da AWS
    - Usar DescribeOrderableDBInstanceOptions com o AWS SDK ou a CLI
    - Usar ModifyDBClusterParameterGroup com um SDK da AWS
- Cenários para o Aurora usando AWS SDKs
    - Criar uma API REST de biblioteca de empréstimos
    - Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Exemplos de código para o Aurora usando SDKs da AWS

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Aurora com SDKs da AWS.

Exemplos

- Olá, Aurora
- Conheça os conceitos básicos do Aurora com um SDK da AWS
- Ações para o Aurora usando AWS SDKs
    - Usar CreateDBCluster com um SDK da AWS
    - Usar CreateDBClusterParameterGroup com um SDK da AWS
    - Usar CreateDBClusterSnapshot com um SDK da AWS
    - Usar CreateDBInstance com um SDK da AWS
    - Usar DeleteDBCluster com um SDK da AWS
    - Usar DeleteDBClusterParameterGroup com um SDK da AWS
    - Usar DeleteDBInstance com um SDK da AWS
    - Usar DescribeDBClusterParameterGroups com um SDK da AWS
    - Usar DescribeDBClusterParameters com um SDK da AWS
    - Usar DescribeDBClusterSnapshots com um SDK da AWS
    - Usar DescribeDBClusters com um SDK da AWS
    - Usar DescribeDBEngineVersions com um SDK da AWS
    - Usar DescribeDBInstances com um SDK da AWS
    - Usar DescribeOrderableDBInstanceOptions com o AWS SDK ou a CLI

- Usar ModifyDBClusterParameterGroup com um SDK da AWS

Olá, Aurora

Os exemplos de código a seguir mostram como começar a usar o Aurora.

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();
```

```
        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

Código para o arquivo CMakeLists.txt do CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
```

```
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
```

```
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte DescribeDBClusters na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conheça os conceitos básicos do Aurora com um SDK da AWS

Os exemplos de código a seguir mostram como:

- Crie um grupo de parâmetros de cluster do banco de dados do Aurora e defina os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the
 DescribeDBClusterParametersAsync method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
```

```
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances
method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();
```

```
        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);
```

```
            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
```

```
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family
by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await
auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
```

```
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
```

```
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
 group.");

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
 "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
```

```
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
```

```
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );
```

```
            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }
```

```
        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
```

```
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
```

```
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await
auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
```

```
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
```

```
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;
```

```
/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
```

```
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
```

```
        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }
```

```
                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
```

```
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
```

```
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();
```

```
        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
```

```
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
```

```
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
```

```
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
```

```
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
```

```
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
```

```
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
```

```
   else {
       std::cout << "Let's create a DB cluster." << std::endl;
       const Aws::String administratorName = askQuestion(
               "Enter an administrator username for the database: ");
       const Aws::String administratorPassword = askQuestion(
               "Enter a password for the administrator (at least 8 characters):
");
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 8.  Get a list of engine versions for the parameter group family.
       if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                client)) {
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
           return false;
       }

       std::cout << "The available engines for your parameter group family are:"
                 << std::endl;

       int index = 1;
       for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
           std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                     << std::endl;
           ++index;
       }
       int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
       const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

       engineName = engineVersion.GetEngine();
       engineVersionName = engineVersion.GetEngineVersion();
       std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                 << "' and database '" << DB_NAME << "'.\n"
                 << "The DB cluster is configured to use your custom cluster
parameter group '"
                 << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
```

```
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

       Aws::RDS::Model::CreateDBClusterRequest request;
       request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
       request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
       request.SetEngine(engineName);
       request.SetEngineVersion(engineVersionName);
       request.SetMasterUsername(administratorName);
       request.SetMasterUserPassword(administratorPassword);

       Aws::RDS::Model::CreateDBClusterOutcome outcome =
               client.CreateDBCluster(request);

       if (outcome.IsSuccess()) {
           std::cout << "The DB cluster creation has started."
                     << std::endl;
       }
       else {
           std::cerr << "Error with Aurora::CreateDBCluster. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
           return false;
       }
   }

   std::cout << "Waiting for the DB cluster to become available." << std::endl;

   int counter = 0;
   // 11. Wait for the DB cluster to become available.
   do {
       std::this_thread::sleep_for(std::chrono::seconds(1));
       ++counter;
       if (counter > 900) {
           std::cerr << "Wait for cluster to become available timed out ofter "
                     << counter
                     << " seconds." << std::endl;
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                            DB_CLUSTER_IDENTIFIER, "", client);
           return false;
       }
```

```
        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
""",
                             client);
            return false;
        }
```

```
        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
""",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
```

```
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." <<
std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
```

```
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
```

```
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
```

```
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

```
//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }
```

```
    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
```

```
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
```

```
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
```

```
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
```

```
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup

- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
```

```
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
```

```
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
```

```
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
```

```
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
```

```
   log.Println("Cluster created and available.")
  }
 }
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
 if err != nil {
  panic(err)
 }
 if dbInstance == nil {
  log.Println("Let's create a database instance in your DB cluster.")
  log.Println("First, choose a DB instance type:")
  instOpts, err := scenario.dbClusters.GetOrderableInstances(
   ctx, *cluster.Engine, *cluster.EngineVersion)
  if err != nil {
   panic(err)
  }
  var instChoices []string
  for _, opt := range instOpts {
   instChoices = append(instChoices, *opt.DBInstanceClass)
  }
  slices.Sort(instChoices)
  instChoices = slices.Compact(instChoices)
  instIndex := scenario.questioner.AskChoice(
   "Which DB instance class do you want to use?\n", instChoices)
```

```
  log.Println("Creating a database instance. This typically takes several
 minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
   ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
 *cluster.Engine,
   instChoices[instIndex])
  if err != nil {
   panic(err)
  }
  for *dbInstance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
   if err != nil {
    panic(err)
   }
  }
 }
 log.Println("Instance data:")
 log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *dbInstance.Engine)
 log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
```

```
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
```

```
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
```

```
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
```

```
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

```
// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
```

```
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
```

```
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
```

```
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots

- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
```

```
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
 family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
 identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
 +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
```

```
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
```

```
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
```

```
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }
```

```
   private static SecretsManagerClient getSecretClient() {
       Region region = Region.US_WEST_2;
       return SecretsManagerClient.builder()
               .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
               .build();
   }

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
```

```
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
```

```
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
```

```
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();
```

```
            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
```

```
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
```

```
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
```

```
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()
```

```
    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected
engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
```

```
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
```

```
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
```

```
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
```

```
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
```

```
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.

- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
```

```
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB cluster parameter group that is compatible with a selected
engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
{parameter_group_name}."
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
```

```
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.
```

```
        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
```

```
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
```

```
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
 how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)
```

```
   def create_snapshot(self, cluster_name):
       """
       Shows how to create a DB cluster snapshot and wait until it's available.

       :param cluster_name: The name of a DB cluster to snapshot.
       """
       if q.ask(
           "Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno
       ):
           snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
           print(
               f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
           )
           snapshot = self.aurora_wrapper.create_cluster_snapshot(
               snapshot_id, cluster_name
           )
           while snapshot.get("Status") != "available":
               wait(30)
               snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
           pp(snapshot)
           print("-" * 88)

   def cleanup(self, db_inst, cluster, parameter_group):
       """
       Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
       Before the DB cluster parameter group can be deleted, all associated DB
instances and
       DB clusters must first be deleted.

       :param db_inst: The DB instance to delete.
       :param cluster: The DB cluster to delete.
       :param parameter_group: The DB cluster parameter group to delete.
       """
       cluster_name = cluster["DBClusterIdentifier"]
       parameter_group_name = parameter_group["DBClusterParameterGroupName"]
       if q.ask(
           "\nDo you want to delete the database instance, DB cluster, and
parameter "
           "group (y/n)? ",
           q.is_yesno,
```

```
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
```

```
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

```
   def create_db_cluster(
       self,
       cluster_name,
       parameter_group_name,
       db_name,
       db_engine,
       db_engine_version,
       admin_name,
       admin_password,
   ):
       """
       Creates a DB cluster that is configured to use the specified parameter
group.
       The newly created DB cluster contains a database that uses the specified
engine and
       engine version.

       :param cluster_name: The name of the DB cluster to create.
       :param parameter_group_name: The name of the parameter group to associate
with
                                    the DB cluster.
       :param db_name: The name of the database to create.
       :param db_engine: The database engine of the database that is created,
such as MySql.
       :param db_engine_version: The version of the database engine.
       :param admin_name: The user name of the database administrator.
       :param admin_password: The password of the database administrator.
       :return: The newly created DB cluster.
       """
       try:
           response = self.rds_client.create_db_cluster(
               DatabaseName=db_name,
               DBClusterIdentifier=cluster_name,
               DBClusterParameterGroupName=parameter_group_name,
               Engine=db_engine,
               EngineVersion=db_engine_version,
               MasterUsername=admin_name,
               MasterUserPassword=admin_password,
           )
           cluster = response["DBCluster"]
       except ClientError as err:
           logger.error(
               "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
```

```
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
       :param instance_class: The DB instance class for the newly created DB
instance.
       :return: Data about the newly created DB instance.
       """
       try:
           response = self.rds_client.create_db_instance(
               DBInstanceIdentifier=instance_id,
               DBClusterIdentifier=cluster_id,
               Engine=db_engine,
               DBInstanceClass=instance_class,
           )
           db_inst = response["DBInstance"]
       except ClientError as err:
           logger.error(
               "Couldn't create DB instance %s. Here's why: %s: %s",
               instance_id,
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return db_inst


   def get_engine_versions(self, engine, parameter_group_family=None):
       """
       Gets database engine versions that are available for the specified engine
       and parameter group family.

       :param engine: The database engine to look up.
       :param parameter_group_family: When specified, restricts the returned
list of
                                      engine versions to those that are
compatible with
                                      this parameter group family.
       :return: The list of database engine versions.
       """
       try:
           kwargs = {"Engine": engine}
           if parameter_group_family is not None:
               kwargs["DBParameterGroupFamily"] = parameter_group_family
```

```
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

```
    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

Uma biblioteca contendo as funções específicas para o cenário do Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
```

```
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
```

```
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
```

```
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }
```

```
        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
```

```
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
```

```
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
```

```
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
```

```
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
```

```
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
```

```
   // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
   pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
       let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
       let snapshot = self
           .rds
           .snapshot_cluster(id, format!("{id}_{name}").as_str())
           .await;
       match snapshot {
           Ok(output) => match output.db_cluster_snapshot {
               Some(snapshot) => Ok(snapshot),
               None => Err(ScenarioError::with("Missing Snapshot")),
           },
           Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
       }
   }

   pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
       let mut clean_up_errors: Vec<ScenarioError> = vec![];

       // Delete the instance. rds.DeleteDbInstance.
       let delete_db_instance = self
           .rds
           .delete_db_instance(
               self.db_instance_identifier
                   .as_deref()
                   .expect("instance identifier"),
           )
           .await;
       if let Err(err) = delete_db_instance {
           let identifier = self
               .db_instance_identifier
               .as_deref()
               .unwrap_or("Missing Instance Identifier");
           let message = format!("failed to delete db instance {identifier}");
           clean_up_errors.push(ScenarioError::new(message, &err));
       } else {
           // Wait for the instance to delete
           let waiter = Waiter::default();
           while waiter.sleep().await.is_ok() {
               let describe_db_instances =
self.rds.describe_db_instances().await;
```

```
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
 rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Testes da biblioteca usando automocks em torno do wrapper do RDS Client.

```
use crate::rds::MockRdsImpl;
```

```
use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
```

```
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
```

```
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
```

```
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}
```

```
#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
```

```
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()
```

```
 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
```

```
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
```

```
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}
```

```
#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Um binário para executar o cenário do início ao fim usando o inquiridor para que o usuário possa tomar algumas decisões.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }
```

```
    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
```

```
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
```

```
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
```

```
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
```

```
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
```

```
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
```

```
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Um wrapper do serviço Amazon RDS que permite automocking para testes.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
```

```
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
```

```
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
```

```
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
```

```
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
```

```
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

O Cargo.toml com dependências usadas neste cenário.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
```

```
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para o Aurora usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Aurora com AWS SDKs. Cada exemplo inclui um link para o GitHub, em que é possível encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Aurora, e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Aurora usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Aurora.

Exemplos

- Usar CreateDBCluster com um SDK da AWS
- Usar CreateDBClusterParameterGroup com um SDK da AWS
- Usar CreateDBClusterSnapshot com um SDK da AWS
- Usar CreateDBInstance com um SDK da AWS
- Usar DeleteDBCluster com um SDK da AWS
- Usar DeleteDBClusterParameterGroup com um SDK da AWS
- Usar DeleteDBInstance com um SDK da AWS
- Usar DescribeDBClusterParameterGroups com um SDK da AWS
- Usar DescribeDBClusterParameters com um SDK da AWS
- Usar DescribeDBClusterSnapshots com um SDK da AWS
- Usar DescribeDBClusters com um SDK da AWS
- Usar DescribeDBEngineVersions com um SDK da AWS
- Usar DescribeDBInstances com um SDK da AWS
- Usar DescribeOrderableDBInstanceOptions com o AWS SDK ou a CLI
- Usar ModifyDBClusterParameterGroup com um SDK da AWS

Usar **`CreateDBCluster`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `CreateDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
/// <param name="adminName">The admin username.</param>
/// <param name="adminPassword">The primary admin password.</param>
/// <returns>The cluster object.</returns>
public async Task<DBCluster> CreateDBClusterWithAdminAsync(
    string dbName,
    string clusterIdentifier,
    string parameterGroupName,
    string dbEngine,
    string dbEngineVersion,
    string adminName,
    string adminPassword)
{
    var request = new CreateDBClusterRequest
    {
        DatabaseName = dbName,
        DBClusterIdentifier = clusterIdentifier,
        DBClusterParameterGroupName = parameterGroupName,
        Engine = dbEngine,
```

```
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetEngine(engineName);
    request.SetEngineVersion(engineVersionName);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBClusterOutcome outcome =
            client.CreateDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster creation has started."
                  << std::endl;
```

```
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
```

```
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
```

```
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
```

```
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
```

```
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
```

```
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }
```

```
        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
```

```
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
```

```
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
```

```
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
```

```
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });
```

```
    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte CreateFunction na Referência de API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`CreateDBClusterParameterGroup`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `CreateDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
```

```
    DBParameterGroupFamily:      aws.String(parameterGroupFamily),
    Description:                 aws.String(description),
   })
  if err != nil {
   log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
   return nil, err
  } else {
   return output.DBClusterParameterGroup, err
  }
 }
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.
```

```
        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
               db_cluster_parameter_group: None,
               ..
           }) => {
               return Err(ScenarioError::with(
                   "CreateDBClusterParameterGroup had empty response",
               ));
           }
           Err(error) => {
               if error.code() == Some("DBParameterGroupAlreadyExists") {
                   info!("Cluster Parameter Group already exists, nothing to
do");
               } else {
                   return Err(ScenarioError::new(
                       "Could not create Cluster Parameter Group",
                       &error,
                   ));
               }
           }
           _ => {
               info!("Created Cluster Parameter Group");
           }
       }

       Ok(())
   }
```

```
    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}
```

```
#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Consulte detalhes da API em CreateDBClusterParameterGroup na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    TRY.
        DATA(lo_output) = lo_rds->createdbclusterparamgroup(
          iv_dbclusterparamgroupname = iv_param_group_name
          iv_dbparametergroupfamily = iv_param_group_family
          iv_description = iv_description
        ).
        oo_result = lo_output->get_dbclusterparametergroup( ).
      CATCH /aws1/cx_rdsdbparmgralrexfault.
        " Re-raise exception - parameter group already exists
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbparmgralrexfault.
      CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
        " Re-raise exception - quota exceeded
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrquotaexcd00.
    ENDTRY.
```

- Para ver detalhes da API, consulte CreateDBClusterParameterGroup na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`CreateDBClusterSnapshot`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `CreateDBClusterSnapshot`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   /// <summary>
   /// Create a snapshot of a cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
   /// <returns>DB snapshot object.</returns>
   public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
   {
       var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
           new CreateDBClusterSnapshotRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               DBClusterSnapshotIdentifier = snapshotIdentifier,
           });

       return response.DBClusterSnapshot;
```

```
    }
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
```

```
            }
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
```

```
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Consulte detalhes da API em CreateDBClusterSnapshot na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
```

```
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }

       info!(
           "Started a db cluster: {}",
           self.db_cluster_identifier
               .as_deref()
               .unwrap_or("Missing ARN")
       );
```

```
        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
```

```
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }
```

```
        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
```

```
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte CreateMultipartUpload na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`CreateDBInstance`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `CreateDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
```

```
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
 that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
 created.
        :param db_engine: The database engine of a database to create in the DB
 instance.
```

```
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
```

```
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
```

```
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
```

```
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
```

```
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte CreateFunction na Referência de API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DeleteDBCluster`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DeleteDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   /// <summary>
   /// Delete a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>DB cluster object.</returns>
   public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
   {
       var response = await _amazonRDS.DeleteDBClusterAsync(
           new DeleteDBClusterRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               SkipFinalSnapshot = true
           });

       return response.DBCluster;
   }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
```

```
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte DeleteDBCluster na AWS Referência da API do SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DeleteDBClusterParameterGroup`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DeleteDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
```

```
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
```

```
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
```

```
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DeleteDBClusterParameterGroup na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
TRY.
    lo_rds->deletedbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
    ).
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    " Re-raise exception - invalid state
    RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
ENDTRY.
```

- Para ver detalhes da API, consulte DeleteDBClusterParameterGroup na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DeleteDBInstance`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DeleteDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
```

```
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
```

```
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
```

```
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
```

```
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
```

```
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
```

```
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte DeleteDBInstance na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeDBClusterParameterGroups`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterParameterGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        else:
            return parameter_group
```

- Consulte detalhes da API em DescribeDBClusterParameterGroups na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    TRY.
        DATA(lo_output) = lo_rds->describedbclusterparamgroups(
          iv_dbclusterparamgroupname = iv_param_group_name
        ).
        DATA(lt_param_groups) = lo_output->get_dbclusterparametergroups( ).
        IF lines( lt_param_groups ) > 0.
          oo_result = lt_param_groups[ 1 ].
        ENDIF.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    ENDTRY.
```

- Para ver detalhes da API, consulte DescribeDBClusterParameterGroups na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeDBClusterParameters`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterParameters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
```

```
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,
```

```
 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
```

```
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
```

```
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
```

```
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Consulte detalhes da API em DescribeDBClusterParameters na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
TRY.
    DATA lv_marker TYPE /aws1/rdsstring VALUE ''.
    DATA lt_all_parameters TYPE /aws1/cl_rdsparameter=>tt_parameterslist.

    DO.
      DATA(lo_output) = lo_rds->describedbclusterparameters(
        iv_dbclusterparamgroupname = iv_param_group_name
        iv_source = iv_source
        iv_marker = lv_marker
      ).

      LOOP AT lo_output->get_parameters( ) INTO DATA(lo_param).
        IF iv_name_prefix IS INITIAL OR
           lo_param->get_parametername( ) CP |{ iv_name_prefix }*|.
          APPEND lo_param TO lt_all_parameters.
        ENDIF.
      ENDLOOP.

      lv_marker = lo_output->get_marker( ).
      IF lv_marker IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_parameters = lt_all_parameters.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
ENDTRY.
```

- Para ver detalhes da API, consulte DescribeDBClusterParameters na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar `DescribeDBClusterSnapshots` com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterSnapshots`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   /// <summary>
   /// Return a list of DB snapshots for a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>List of DB snapshots.</returns>
   public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
   {
       var results = new List<DBClusterSnapshot>();

       DescribeDBClusterSnapshotsResponse response;
       DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
       {
```

```
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Consulte detalhes da API em DescribeDBClusterSnapshots na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
```

```
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Consulte detalhes da API em DescribeDBClusterSnapshots na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Consulte detalhes da API em DescribeDBClusterSnapshots na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
```

```
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBClusterSnapshots na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Consulte detalhes da API em DescribeDBClusterSnapshots na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para ter detalhes da API, consulte DescribeDBClusterSnapshots na Referência de API do AWS SDK para Python (Boto3).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeDBClusters`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
```

```
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
```

```
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
```

```
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
```

```
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte DescribeDBClusters na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeDBEngineVersions`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBEngineVersions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                       engine versions to those that are
 compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
```

```
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");
```

```
        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Consulte detalhes da API em DescribeDBEngineVersions na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    " iv_engine                = 'mysql'
    " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
    TRY.
        oo_result = lo_rds->describedbengineversions(
          iv_engine                = iv_engine
          iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
        DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
        MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
    ENDTRY.
```

- Para ver detalhes da API, consulte DescribeDBEngineVersions na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeDBInstances`** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `DescribeDBInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Returns a list of DB instances.
```

```
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Waits until the database instance is available.
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
```

```
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
```

```
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
```

```
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **`DescribeOrderableDBInstanceOptions`** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `DescribeOrderableDBInstanceOptions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
```

```
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
```

```
    }
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK for Java 2.x.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para ter detalhes da API, consulte DescribeOrderableDBInstanceOptions na Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
```

```
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para ter detalhes da API, consulte DescribeOrderableDBInstanceOptions na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
```

```
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
           .rds
           .describe_orderable_db_instance_options(
               DB_ENGINE,
               self.engine_version
                   .as_ref()
                   .expect("engine version for db instance options")
                   .as_str(),
           )
           .await;

       describe_orderable_db_instance_options_items
           .map(|options| {
               options
                   .iter()
                   .filter(|o| o.storage_type() == Some("aurora"))
                   .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                   .collect::<Vec<String>>()
           })
           .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
   }

   pub async fn describe_orderable_db_instance_options(
       &self,
       engine: &str,
       engine_version: &str,
   ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
```

```
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
```

```
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;
```

```
    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Consulte detalhes da API em DescribeOrderableDBInstanceOptions na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
 TYPE 'I'.
    ENDTRY.
```

- Para ver detalhes da API, consulte DescribeOrderableDBInstanceOptions na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ModifyDBClusterParameterGroup** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o `ModifyDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
```

```
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência AWS SDK para .NET da API do.

C++

SDK para C++

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);
```

```
        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência AWS SDK para C++ da API do.

Go

SDK para Go V2

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência AWS SDK para Go da API do.

Java

SDK para Java 2.x

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
```

```
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
 rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência AWS SDK for Java 2.x da API do.

Kotlin

SDK para Kotlin

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }
```

```
    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
```

```
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
```

```
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
```

```
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Consulte detalhes da API em ModifyDBClusterParameterGroup na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWSCode Examples Repository.

```
TRY.
    oo_result = lo_rds->modifydbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      it_parameters = it_update_parameters
    ).
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    " Re-raise exception - invalid state
    RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
ENDTRY.
```

- Para ver detalhes da API, consulte ModifyDBClusterParameterGroup na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para o Aurora usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Aurora com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Aurora ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma API REST de biblioteca de empréstimos
- Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

Python

SDK para Python (Boto3).

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice a fim de criar uma API REST com suporte por um banco de dados do Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Utilize o AWS Secrets Manager para gerenciar credenciais de bancos de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST com tecnologia sem servidor no Amazon API Gateway e no AWS Lambda.
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar o AWS SDK para .NET para desenvolver uma aplicação Web que monitora itens de trabalho no banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end .NET RESTful.

- Integre uma aplicação Web do React com os serviços da AWS.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o script do AWS CloudFormation incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no GitHub.

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

C++

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar a API REST de C++ que consulta os dados do Amazon Aurora Sem Servidor e para uso por uma aplicação React, consulte o exemplo completo no GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar a API Spring REST que consulta os dados do Amazon Aurora Sem Servidor e para uso por uma aplicação React, consulte o exemplo completo no GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que utiliza a API JDBC, consulte o exemplo completo no GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar uma aplicação Web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end Node.js Express.

- Integre uma aplicação Web React.js com Serviços da AWS.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o script do AWS CloudFormation incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no GitHub.

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar a API Spring REST que consulta os dados do Amazon Aurora Sem Servidor e para uso por uma aplicação React, consulte o exemplo completo no GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

PHP

SDK para PHP

Mostra como usar o AWS SDK para PHP para construir uma aplicação Web que monitora itens de trabalho no banco de dados do Amazon RDS e envia relatórios por e-mail usando o

Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end construído com React.js para interagir com um back-end PHP RESTful.

- Integre uma aplicação Web React.js com os serviços da AWS.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o script do AWS CloudFormation incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no GitHub.

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Python

SDK para Python (Boto3).

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho no banco de dados do Amazon Aurora Sem Servidor e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa o framework da Web do Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço REST do Flask que se integre a Serviços da AWS.
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um segredo do AWS Secrets Manager que contenha credenciais do banco de dados e use-o a fim de autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no GitHub.

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Para obter uma lista completa dos Guias do desenvolvedor do AWS SDK e exemplos de código, consulte Using this service with an AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Melhores práticas do Amazon Aurora

A seguir, você encontra informações sobre as melhores práticas e as opções gerais para usar ou migrar dados para um cluster de bancos de dados Amazon Aurora.

Algumas melhores práticas do Amazon Aurora são específicas a um mecanismo de banco de dados específico. Para obter mais informações sobre as melhores práticas do Aurora específicas de um mecanismo de banco de dados, consulte:

Mecanismo do banco de dados	Práticas recomendadas
Amazon Aurora MySQL	Consulte Melhores práticas do Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Consulte Práticas recomendadas do Amazon Aurora PostgreSQL

> **Note**
>
> Para obter as recomendações comuns para o Aurora, consulte Recomendações do Amazon Aurora.

Tópicos

- Diretrizes operacionais básicas do Amazon Aurora
- Recomendações de RAM para a instância de banco de dados
- Drivers de banco de dados da AWS
- Monitoramento do Amazon Aurora
- Como trabalhar com grupos de parâmetros de banco de dados e grupos de parâmetros de cluster de banco de dados
- Vídeo de práticas recomendadas do Amazon Aurora

Diretrizes operacionais básicas do Amazon Aurora

As diretrizes operacionais básicas a seguir devem ser seguidas por todos ao trabalhar com o Amazon Aurora. O Acordo de Nível de Serviço do Amazon RDS exige que você siga estas diretrizes:

- Monitore sua memória, CPU e uso de armazenamento. O Amazon CloudWatch pode ser configurado para lhe notificar quando os padrões de uso mudam ou quando você se aproxima da capacidade de implantação. Dessa maneira, é possível manter a disponibilidade e a performance do sistema.
- Se o seu aplicativo cliente estiver armazenando em cache os dados do Serviço de Nome de Domínio (DNS) de suas instâncias de banco de dados, defina um valor de tempo de vida (TTL) de menos de 30 segundos. O endereço IP subjacente de uma instância de banco de dados pode ser alterado após um failover. Portanto, armazenar em cache os dados de DNS por um tempo prolongado poderá resultar em falhas de conexão se a aplicação tentar se conectar a um endereço IP que não está mais em serviço. Os clusters de bancos de dados Aurora com várias réplicas de leitura também podem sofrer falhas de conexão quando as conexões usam o endpoint de leitor e uma das instâncias da réplica de leitura está em manutenção ou foi excluída.
- Teste o failover do cluster de banco de dados para entender quanto tempo o processo leva para seu caso de uso. O teste de failover pode ajudar a garantir que a aplicação que acessa o cluster de banco de dados possa se conectar automaticamente ao novo cluster de banco de dados após o failover.

Recomendações de RAM para a instância de banco de dados

Para otimizar o desempenho, aloque RAM suficiente para que seu conjunto de trabalho resida quase que completamente na memória. Para determinar se o seu conjunto de trabalho está quase todo na memória, examine as seguintes métricas no Amazon CloudWatch:

- `VolumeReadIOPS` – essa métrica calcula o número médio de operações de E/S de leitura de um volume de cluster, relatado em intervalos de cinco minutos. O valor de `VolumeReadIOPS` deve ser pequeno e estável. Em alguns casos, você pode achar que a E/S de leitura está apresentando picos ou é maior do que a habitual. Se esse for o caso, investigue as instâncias de banco de dados no cluster de banco de dados para ver quais instâncias de banco de dados estão causando o aumento de E/S.

> **Tip**
>
> Se seu cluster do Aurora MySQL usar consulta paralela, você poderá ver um aumento nos valores de `VolumeReadIOPS`. As consultas paralelas não usam o grupo de buffers. Assim, embora as consultas sejam rápidas, esse processamento otimizado pode resultar em aumento nas operações de leitura e nas cobranças associadas.

- `BufferCacheHitRatio` – essa métrica mede a porcentagem de solicitações que são atendidas pelo cache do buffer de uma instância de banco de dados em seu cluster de banco de dados. Essa métrica fornece informações sobre a quantidade de dados que estão sendo obtidos da memória.

  Uma alta taxa de acerto indica que sua instância de banco de dados tem memória suficiente disponível. Uma alta de acertos baixa indica que as consultas nessa instância de banco de dados estão acessando o disco com maior frequência. Investigue a workload para ver quais consultas estão causando esse comportamento.

Se, após a investigação da carga de trabalho, você descobrir que precisa de mais memória, considere expandir a classe de instância de banco de dados para uma classe com mais RAM. Depois disso, você poderá investigar as métricas discutidas anteriormente acima e continuar a expansão, se necessário. Se o cluster do Aurora for maior do que 40 TB, não use classes de instância db.t2, db.t3 ou db.t4g.

Para obter mais informações, consulte Métricas do Amazon CloudWatch para o Amazon Aurora.

Drivers de banco de dados da AWS

Recomendamos o pacote de drivers da AWS para conectividade de aplicações. Os drivers foram projetados para comportar tempos mais rápidos de transição e failover, além de autenticação com o AWS Secrets Manager, o AWS Identity and Access Management (IAM) e identidades federadas. Os drivers da AWS dependem do monitoramento do status do cluster de banco de dados e do conhecimento da topologia do cluster para determinar o novo gravador. Essa abordagem reduz os tempos de transição e de failover para segundos de um dígito, em comparação com dezenas de segundos para drivers de código aberto.

Como novos recursos do serviço são introduzidos, o objetivo do pacote de drivers da AWS é ter suporte integrado para esses recursos do serviço.

Para obter mais informações, consulte Conectar-se aos clusters de banco de dados do Aurora com os drivers da AWS.

Monitoramento do Amazon Aurora

O Amazon Aurora fornece várias métricas e informações que você pode monitorar para determinar a integridade e a performance do cluster de bancos de dados do Aurora. É possível usar várias ferramentas, como o Console de gerenciamento da AWS, a AWS CLI e a API do CloudWatch para visualizar métricas do Aurora. É possível visualizar as métricas combinadas do Insights de Performance e do CloudWatch no painel do Insights de Performance e monitorar a instância de banco de dados. Para usar essa visualização de monitoramento, o Insights de Performance deve estar ativado para a instância de banco de dados. Para obter mais informações sobre essa visualização de monitoramento, consulte Visualizar métricas combinadas com o painel do Insights de Performance.

Você pode criar um relatório de análise de performance para um período específico e visualizar os insights identificados e as recomendações para resolver os problemas. Para ter mais informações, consulte Criar um relatório de análise de performance no Insights de Performance.

Como trabalhar com grupos de parâmetros de banco de dados e grupos de parâmetros de cluster de banco de dados

Recomendamos que você experimente fazer mudanças de parameter groups de banco de dados e de parameter groups de cluster de banco de dados em um cluster de banco de dados de teste, antes de aplicar alterações de parameter group ao cluster de banco de dados de produção. A definição incorreta de parâmetros do mecanismo de banco de dados pode ter efeitos adversos não intencionais, inclusive a diminuição no desempenho e a instabilidade do sistema.

Sempre tenha cuidado ao modificar os parâmetros do mecanismo de banco de dados e faça backup do cluster de banco de dados antes de modificar um grupo de parâmetros de banco de dados. Para obter informações sobre o backup do cluster de banco de dados, consulte Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora.

Vídeo de práticas recomendadas do Amazon Aurora

O canal das AWS Online Tech Talks no YouTube inclui uma apresentação em vídeo sobre as práticas recomendadas para criar e configurar um cluster de bancos de dados do Amazon Aurora

mais seguro e altamente disponível. Consulte Práticas recomendadas para alta disponibilidade do Amazon Aurora.

Executar uma prova de conceito com o Amazon Aurora

Veja a explicação a seguir sobre como configurar e executar uma prova de conceito para o Aurora. Uma prova de conceito é uma investigação que você executa para verificar se o Aurora é adequado para seu aplicativo. A prova de conceito pode ajudá-lo a compreender os recursos do Aurora no contexto de seus próprios aplicativos de banco de dados e como o Aurora se compara com seu ambiente de banco de dados atual. Ela também pode mostrar o nível de esforço necessário para mover dados, portar código SQL, ajustar a performance e adaptar seus procedimentos de gerenciamento atuais.

Neste tópico, você pode encontrar uma visão geral e uma descrição detalhada dos procedimentos de alto nível e das decisões envolvidas na execução de uma prova de conceito, que são listados a seguir. Para obter instruções detalhadas, você pode seguir os links para a documentação completa de assuntos específicos.

Visão geral de uma prova de conceito do Aurora

Ao conduzir uma prova de conceito do Amazon Aurora, você aprende o que é necessário para portar seus dados e os aplicativos SQL existentes para o Aurora. Você pratica os aspectos importantes do Aurora em escala, usando um volume de dados e atividades que são representativas para seu ambiente de produção. O objetivo é sentir-se confiante de que os pontos fortes do Aurora coincidem bem com os desafios que fazem com que você aumente excessivamente a infraestrutura de seu banco de dados anterior. No final de uma prova de conceito, você tem um plano sólido para fazer comparações de performance em escala maior e testar aplicativos. Neste ponto, você compreende os itens de trabalho maiores em seu caminho para uma implantação de produção.

O conselho a seguir sobre práticas recomendadas pode ajudar você a evitar erros comuns que provocam problemas durante a comparação. Porém, este tópico não abrange o processo detalhado de como executar comparações e executar o ajuste de performance. Esses procedimentos variam de acordo com a workload e os recursos do Aurora que você usa. Para obter informações detalhadas, consulte a documentação relacionada a performance, como Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora, Melhorias de performance do Amazon Aurora MySQL, Performance e escalabilidade do Amazon Aurora PostgreSQL e Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.

As informações deste tópico se aplicam principalmente aos aplicativos em que sua organização grava o código e projeta o esquema e que oferecem suporte aos mecanismos de banco de dados

de código aberto do MySQL e do PostgreSQL. Se estiver testando um aplicativo comercial ou código gerado por um framework de aplicação, talvez você não tenha a flexibilidade para aplicar todas as diretrizes. Nesses casos, verifique com o representante da AWS para saber se há práticas recomendadas ou estudos de caso do Aurora para seu tipo de aplicação.

1. Identifique seus objetivos

Ao avaliar o Aurora como parte de uma prova de conceito, você escolhe quais medidas tomar e como avaliar o sucesso do exercício.

Você deve garantir que toda a funcionalidade de seu aplicativo seja compatível com o Aurora. Como as versões primárias do Aurora são compatíveis com as versões principais correspondentes do MySQL e do PostgreSQL, a maioria das aplicações desenvolvidas para esses mecanismos também é compatível com o Aurora. No entanto, você ainda deve validar a compatibilidade com cada aplicativo.

Por exemplo, algumas das opções de configuração que você faz ao configurar um cluster do Aurora definem se você pode ou deve usar determinados recursos do banco de dados. Você pode começar com o tipo de uso mais geral de cluster do Aurora, conhecido como provisionado. Então você pode decidir se uma configuração especializada, como uma consulta sem servidor ou paralela, oferece benefícios para sua workload.

Use as seguintes questões para ajudar a identificar e a quantificar seus objetivos:

- O Aurora oferece suporte a todos os casos de uso funcionais de sua workload?
- Qual tamanho de conjunto de dados ou nível de carga você deseja? Você pode dimensionar para esse nível?
- Quais são seus requisitos específicos de taxa de transferência ou latência de consultas? Você pode atingi-los?
- Qual á e quantidade mínima aceitável de tempo limite planejado ou não planejado para sua workload? Você pode atingi-la?
- Quais são as métricas necessárias para a eficiência operacional? Você pode monitorá-las com exatidão?
- O Aurora oferece suporte a suas metas de negócios específicas, como redução de custos, aumento na implantação ou velocidade de provisionamento? Você tem uma maneira de quantificar essas metas?

- Você pode atender a todos os requisitos de segurança e conformidade de sua workload?

Dedique algum tempo para conhecer os mecanismos de banco de dados Aurora e as capacidades da plataforma, e revise a documentação do serviço. Anote todos os recursos que podem ajudá-lo a obter os resultados desejados. Um desses recursos pode ser a consolidação de workloads, descrita na publicação do Blog de banco de dados da AWS Como planejar e otimizar o Amazon Aurora com compatibilidade com o MySQL para workloads consolidadas. Outro pode ser a escalabilidade com base em demanda, descrita em Amazon Aurora Auto Scaling com réplicas do Aurora no Guia do usuário do Amazon Aurora. Outros podem ser ganhos de performance ou operações de banco de dados simplificadas.

2. Entenda as características de workloads

Avalie o Aurora no contexto do caso de uso pretendido. O Aurora é uma boa opção para workloads de processamento de transações online (OLTP). Também é possível executar relatórios no cluster que mantém os dado OLTP em tempo real sem provisionar um cluster de data warehouse separado. Você pode reconhecer se seu caso de uso se encaixa nessas categorias verificando as seguintes características:

- Simultaneidade alta, com dezenas, centenas ou milhares de clientes simultâneos.
- Grande volume de consultas de latência alta (milissegundos a segundos).
- Transações curtas em tempo real.
- Padrões de consulta altamente seletivos, com pesquisas baseadas em índice.
- Para HTAP, consultas de análise que podem se beneficiar da consulta paralela do Aurora.

Um dos principais fatores que afetam suas opções de banco de dados é a velocidade dos dados. A velocidade alta envolve a inserção e a atualização de dados com muita frequência. Esse sistema pode ter milhares de conexões e centenas de milhares de consultas simultâneas lendo e gravando no banco de dados. As consultas em sistemas de velocidade alta afetam um número relativamente pequeno de linhas e, normalmente, acessam várias colunas na mesma linha.

O Aurora foi desenvolvido para lidar com dados em velocidade alta. Dependendo da workload, um cluster do Aurora com um uma única instância de banco de dados r4.16xlarge pode processar mais de 600.000 instruções `SELECT` por segundo. Novamente, dependendo da workload, esse cluster pode processar 200.000 instruções `INSERT`, `UPDATE` e `DELETE` por segundo. O Aurora é um banco

de dados de armazenamento de linhas e é idealmente adequado para alto volume, alta taxa de transferência e workloads OLTP altamente paralelizadas.

O Aurora também pode executar consultas de relatórios no mesmo cluster que lida com a workload OLTP. O Aurora oferece suporte para até 15 réplicas, estando cada uma, em média, de 10 a 20 milissegundos da instância primária. Os analistas podem consultar dados OLTP em tempo real, sem copiar os dados em um cluster de data warehouse separado. Com os clusters do Aurora usando o recurso de consulta paralela, é possível deslocar muito do trabalho de processamento, filtragem e agregação para o subsistema de armazenamento massivamente distribuído do Aurora.

Use essa fase de planejamento para se familiarizar com os recursos do Aurora, outros produtos da AWS, o Console de gerenciamento da AWS e a AWS CLI. Verifique também como eles funcionam com as outras ferramentas que você planeja usar na prova de conceito.

3. Praticar com o Console de gerenciamento da AWS ou a AWS CLI

Como uma próxima etapa, pratique com o Console de gerenciamento da AWS ou a AWS CLI para familiarizar-se com essas ferramentas e com o Aurora.

Praticar com o Console de gerenciamento da AWS

As seguintes atividades iniciais com clusters de banco de dados do Aurora servem principalmente para você se familiarizar com o ambiente do Console de gerenciamento da AWS e praticar a configuração e a modificação de clusters do Aurora. Se você usa mecanismos de banco de dados compatíveis com o MySQL e com o PostgreSQL com o Amazon RDS, poderá se basear nesse conhecimento ao usar o Aurora.

Aproveitando o modelo e os recursos de armazenamento compartilhado do Aurora, como replicação e snapshots, é possível tratar clusters de banco de dados inteiros como outro tipo de objeto que você manipula livremente. Você pode configurar, descartar e alterar a capacidade dos clusters do Aurora frequentemente durante a prova de conceito. Você não é bloqueado por opções anteriores sobre capacidade, configurações de banco de dados e layout físico de dados.

Para começar a usar, configure um cluster vazio do Aurora. Escolha o tipo de capacidade provisionada e o local regional para seus experimentos iniciais.

Conecte-se a esse cluster usando um programa cliente, como um aplicativo de linha de comando SQL. Inicialmente, você se conecta usando o endpoint do cluster. Você se conecta a esse endpoint

para executar qualquer operação de gravação, como instruções de linguagem de definição de dados (DDL - data definition language) e processos de extração, transformação e carregamento (ETL - extract, transform, load). Mais tarde, na prova de conceito, você se conecta a sessões de uso intensivo de consultas usando o endpoint do leitor, que distribui a workload de consulta entre várias instâncias de banco de dados no cluster.

Ampliar o cluster adicionando mais réplicas do Aurora Para esses procedimentos, consulte Replicação com o Amazon Aurora. Amplie ou reduza as instâncias de banco de dados alterando a classe de instância da AWS. Compreenda como o Aurora simplifica esses tipos de operação, de forma que, se suas estimativas iniciais para a capacidade do sistema forem precisas, você possa ajustar posteriormente sem começar de novo.

Crie um snapshot e restaure-o em outro cluster.

Examine as métricas do cluster para verificar as atividades ao longo do tempo, e como as métricas se aplicam às instâncias de banco de dados no cluster.

Familiarizar-se com como fazer essas coisas por meio do Console de gerenciamento da AWS é útil no início. Depois de compreender o que você pode fazer com o Aurora, você pode continuar para a automatização dessas operações usando a AWS CLI. Nas sessões a seguir, você pode encontrar mais detalhes sobre os procedimentos e as práticas recomendadas para essas atividades durante o período de prova de conceito.

Praticar com o AWS CLI

Recomendamos automatizar a implantação e os procedimentos de gerenciamento, mesmo em uma configuração de prova de conceito. Para isso, familiarize-se com a AWS CLI, se ainda não estiver familiarizado. Se você usa mecanismos de banco de dados compatíveis com o MySQL e com o PostgreSQL com o Amazon RDS, poderá se basear nesse conhecimento ao usar o Aurora.

O Aurora normalmente envolve grupos de instâncias de banco de dados organizadas em clusters. Portanto, muitas operações envolvem a determinação de quais instâncias de banco de dados estão associadas a um cluster e a execução de operações administrativas em um loop para todas as instâncias.

Por exemplo, você pode automatizar etapas, como a criação de clusters do Aurora, a ampliação deles com classes de instância maiores ou o aumento deles com instâncias de banco de dados adicionais. Isso ajuda você a repetir qualquer etapa em sua prova de conceito e a explorar cenários hipotéticos com diferentes tipos ou configurações de clusters do Aurora.

Conheça as capacidades e as limitações de ferramentas de implantação de infraestrutura, como o AWS CloudFormation. Você pode descobrir que as atividades que você realiza em uma prova de conceito não são adequadas para uso em produção. Por exemplo, o comportamento do CloudFormation para modificação é criar uma nova instância e excluir a atual, inclusive seus dados. Para obter mais detalhes sobre esse comportamento, consulte Update behaviors of stack resources (Comportamentos de atualização de recursos de pilhas) no Guida do usuário do AWS CloudFormation.

4. Crie seu cluster do Aurora

Com o Aurora, você pode explorar cenários hipotéticos adicionando instâncias de banco de dados ao cluster e ampliando as instâncias de banco de dados para classes de instâncias mais avançadas. Também é possível criar clusters com definições de configuração diferentes para executar a mesma workload lado a lado. Com o Aurora, você tem muita flexibilidade para configurar, descartar e reconfigurar clusters de bancos de dados. Com isso, é útil praticar essas técnicas nas etapas iniciais do processo da prova de conceito. Para obter os procedimentos gerais para criar clusters do Aurora, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Quando prático, inicie com um cluster usando as seguintes configurações. Ignore esta etapa apenas se você tiver certos casos de uso específicos em mente. Por exemplo, você pode ignorar esta etapa se seu caso de uso exigir um tipo especializado de cluster do Aurora. Ou poderá ignorá-la se você precisar de uma combinação específica de mecanismo e versão de banco de dados.

- Desative a opção Easy create (Criação fácil). Para a prova de conceito, recomendamos que você esteja ciente de todas as configurações escolhidas para que possa criar clusters idênticos ou levemente diferentes mais tarde.
- Use uma versão de mecanismo de banco de dados recente. Essas combinações de mecanismo e versão de banco de dados têm ampla compatibilidade com outros recursos do Aurora e utilização substancial dos clientes em aplicações de produção.
  - Aurora MySQL versão 3.x (compatível com o MySQL 8.0)
  - Aurora PostgreSQL versão 15.x ou 16.x
- Escolha o modelo de Dev/Test (Desenvolvimento/teste). Essa opção não é significativa para as atividades de sua prova de conceito.
- Em Instance Size (Tamanho da instância), escolha Memory optimized classes (Classes otimizadas para memória) e uma das classes de instância xlarge. Você pode ajustar a classe da instância para cima ou para baixo posteriormente.

- Em Multi-AZ Deployment (Implantação Multi-AZ), escolha Create an Aurora Replica or Reader node in a different AZ (Criar uma réplica do Aurora ou nó de leitor em uma AZ diferente). Muitos dos aspectos mais úteis do Aurora envolvem clusters de várias instâncias da banco de dados. Faz sentido começar sempre com pelo menos duas instâncias de banco de dados em qualquer cluster novo. O uso de outra zona de disponibilidade para a segunda instância de banco de dados ajuda a testar cenários de alta disponibilidade diferentes.
- Ao escolher nomes para as instâncias de banco de dados, use uma convenção de nomenclatura genérica. Não faça referência a nenhuma instância de banco de dados de cluster como o "gravador", pois instâncias de banco de dados diferentes assumem esses perfis conforme necessários. Recomendamos usar algo como `clustername-az-serialnumber`, por exemplo `myprodappdb-a-01`. Esses nomes identificam exclusivamente a instância de banco de dados e sua localização.
- Defina uma retenção de backup alta para o cluster do Aurora. Com um período de retenção longo, é possível executar uma recuperação point-in-time (PITR -point-in-time recovery) por um período de até 35 dias. Você pode redefinir seu banco de dados para um estado conhecido depois de executar testes que envolvem instruções DDL e linguagem de manipulação de dados (DML - data manipulation language). Também é possível recuperá-lo, se você excluir ou alterar dados por engano.
- Ative recursos adicionais de recuperação, log e monitoramento na criação do cluster. Ative todas as opções em Backtrack, Performance Insights, Monitoramento e Exportações de log. Com esses recursos habilitados, você pode testar a adequação de recursos, como retrocesso, monitoramento avançado e Performance Insights, à sua workload. Você também pode investigar facilmente a performance e solucionar problemas durante a prova de conceito.

5. Configure seu esquema

No cluster do Aurora, configure bancos de dados, tabelas, índices, chaves estrangeiras e outros objetos de esquema para seu aplicativo. Se você estiver mudando de outro sistema de banco de dados compatível com MySQL ou PostgreSQL, espere que esse estágio seja simples e direto. Você usa a mesma sintaxe e linha de comando SQL ou outros aplicativos cliente, com os quais está familiarizado, em seu novo mecanismo de banco de dados.

Para emitir instruções SQL no cluster, localize o endpoint do cluster e forneça esse valor como o parâmetro de conexão a seu aplicativo cliente. Você encontra o endpoint do cluster na guia Connectivity (Conectividade) da página de detalhes do cluster. O endpoint do cluster é o rotulado como Writer (Gravador). O outro endpoint, rotulado como Reader (Leitor), representa uma conexão

somente leitura que você pode fornecer aos usuários finais que executam relatórios ou outras consultas somente leitura. Para obter ajuda com qualquer problema relativo à conexão ao cluster, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Se você estiver portando seu esquema e dados de outro sistema de banco de dados, espere fazer algumas alterações no esquema neste ponto. Essas alterações no esquema são para fazer a correspondência com a sintaxe e os recursos do SQL disponíveis no Aurora. Você pode excluir determinadas colunas, restrições, triggers ou outros objetos de esquema neste ponto. Isso pode ser útil principalmente se esses objetos exigirem retrabalho para compatibilidade com o Aurora e não forem significativos para os objetivos de sua prova de conceito.

Se você estiver migrando de um sistema de banco de dados com um mecanismo subjacente diferente do Aurora, considere o uso do AWS Schema Conversion Tool (AWS SCT) para simplificar o processo. Para obter detalhes, consulte o Guia do usuário do AWSSchema Conversion Tool. Para obter detalhes gerais sobre as atividades de migração e de portabilidade, consulte o whitepaper da AWS Migrar seus bancos de dados para o Amazon Aurora.

Durante esta etapa, é possível avaliar se há ineficiências na configuração de seu esquema, por exemplo, em sua estratégia de indexação ou em outras estruturas de tabelas, como tabelas particionadas. Essas ineficiências podem ser ampliadas quando você implanta seu aplicativo em um cluster com várias instâncias de banco de dados e com uma workload pesada. Considere se você pode fazer um ajuste fino desses aspectos de performance agora ou durante atividades posteriores, como em um teste de comparação completo.

6. Importe seus dados

Durante a prova de conceito, você traz os dados, ou uma amostra representativa dos dados, de seu sistema de banco de dados antigo. Se for prático, configure pelo menos alguns dados em cada uma de suas tabelas. Isso ajuda a testar a compatibilidade de todos os tipos de dados e recursos de esquema. Depois de exercitar os recursos básicos do Aurora, aumente a quantidade de dados. No momento da conclusão da prova de conceito, você deve testar suas ferramentas de ETL, consultas e workload geral com um conjunto de dados que seja grande o suficiente para obter conclusões precisas.

Você pode usar várias técnicas para importar dados de backup físico ou lógico para o Aurora. Para obter detalhes, consulte Migrar dados para um cluster de banco de dados do Amazon Aurora MySQL ou Migrar dados para o Amazon Aurora com compatibilidade com o PostgreSQL, dependendo do mecanismo de banco de dados que você está usando na prova de conceito.

Experimente com as ferramentas de ETL e as tecnologias que estão sendo consideradas. Veja qual delas atende às suas necessidades. Considere tanto a taxa de transferência quanto a flexibilidade. Por exemplo, algumas ferramentas de ETL executam uma transferência uma vez, e outras envolvem a replicação contínua do sistema antigo para o Aurora.

Se estiver migrando de um sistema compatível com MySQL para o Aurora MySQL, você poderá usar as ferramentas de transferência de dados nativas. O mesmo se aplicará, ao migrar de um sistema compatível com o PostgreSQL para o Aurora PostgreSQL. Se estiver migrando de um sistema de banco de dados que usa um mecanismo subjacente diferente do usado pelo Aurora, você poderá experimentar com o AWS Database Migration Service (AWS DMS). Para obter detalhes sobre o AWS DMS, consulte o Guia do usuário do AWS Database Migration Service.

Para obter detalhes sobre as atividades de migração e de portabilidade, consulte o whitepaper da AWS Aurora migration handbook (Manual de migração).

7. Porte seu código SQL

O teste do SQL e dos aplicativos associados exige níveis diferentes de esforço, dependendo dos diferentes casos. Especificamente, o nível de esforço depende de se a mudança é de um sistema compatível com MySQL ou PostgreSQL ou de outro tipo.

- Se você estiver mudando do RDS for MySQL ou do RDS for PostgreSQL, as alterações no SQL serão suficientemente pequenas, permitindo que você teste o código SQL original com o Aurora e incorpore as alterações necessárias manualmente.
- De forma semelhante, se você mudar de um banco de dados local compatível com o MySQL ou o PostgreSQL, poderá testar o código SQL original e incorporar alterações manualmente.
- Se estiver partindo de um banco de dados comercial diferente, alterações necessárias do SQL serão mais extensivas. Nesse caso, considere usar o AWS SCT.

Durante esta etapa, é possível avaliar se há ineficiências na configuração de seu esquema, por exemplo, em sua estratégia de indexação ou em outras estruturas de tabelas, como tabelas particionadas. Considere se você pode fazer um ajuste fino desses aspectos de performance agora ou durante atividades posteriores, como em um teste de comparação completo.

Você pode verificar a lógica de conexão do banco de dados em seu aplicativo. Para aproveitar o processamento distribuído do Aurora, pode ser necessário usar conexões separadas para operações de leitura e de gravação e usar sessões relativamente curtas para operações de consulta. Para obter informações sobre conexões, consulte 9. Conectar-se ao Aurora.

Considere os compromissos e trocas que você talvez tenha precisado fazer para resolver problemas em seu banco de dados de produção. Reserve tempo na programação da prova de conceito para fazer melhorias em seu design de esquema e consultas. Para determinar se é possível obter ganhos fáceis em performance, custo operacional e escalabilidade, tente as aplicações originais e modificadas lado a lado em diferentes clusters do Aurora.

Para obter detalhes sobre as atividades de migração e de portabilidade, consulte o whitepaper da AWS Aurora migration handbook (Manual de migração).

8. Especifique definições da configuração

Você também pode rever os parâmetros da configuração de seu banco de dados como parte do exercício de prova de conceito do Aurora. Talvez você já tenha as definições de configuração do MySQL ou do PostgreSQL ajustadas para performance e escalabilidade em seu ambiente atual. O subsistema de armazenamento do Aurora é adaptado e ajustado para um ambiente distribuído baseado na nuvem com um subsistema de armazenamento de velocidade alta. Como resultado, muitas configurações do mecanismo de banco de dados anterior não se aplicam. Recomendamos conduzir experimentos iniciais com as definições de configuração padrão do Aurora. Reaplique as configurações de seu ambiente atual apenas se você encontrar gargalos de performance e de escalabilidade. Se tiver interesse, você poderá analisar esse assunto mais profundamente em Apresentação do mecanismo de armazenamento do Aurora no blog de banco de dados da AWS.

O Aurora facilita a reutilização das definições da configuração ideais para um aplicativo específico ou caso de uso. Em vez de editar um arquivo de configuração separado para cada instância de banco de dados, você gerencia os conjuntos de parâmetros que atribui a clusters inteiros ou a instâncias de banco de dados específicas. Por exemplo, a configuração do fuso horário se aplica a todas as instâncias de banco de dados no cluster, e você pode ajustar o tamanho do cache de página para cada instância de banco de dados.

Você começa com um dos conjuntos de parâmetros padrão e aplica alterações apenas aos parâmetros que você precisa ajustar. Para obter detalhes sobre como trabalhar com grupos de parâmetros, consulte Parâmetros do cluster de banco de dados e da instância de bancos de dados Amazon Aurora. Para obter as definições da configuração que são ou não aplicáveis aos clusters do Aurora, consulte Parâmetros de configuração do Aurora MySQL ou Amazon Aurora PostgreSQL parameters dependendo de seu mecanismo de banco de dados.

9. Conectar-se ao Aurora

Como você descobre ao criar seu esquema e configuração de dados iniciais e executar consultas de exemplo, é possível conectar-se a endpoints diferentes em um cluster do Aurora. O endpoint a ser usado depende de se a operação é uma leitura, como uma instrução `SELECT`, ou uma gravação, como uma instrução `CREATE` ou `INSERT`. Conforme você aumenta a workload em um cluster do Aurora e experimenta com os recursos do Aurora, é importante que seu aplicativo atribua cada operação ao endpoint adequado.

Ao usar o endpoint do cluster para operações de gravação, você sempre se conecta a uma instância de banco de dados no cluster que tem capacidade de leitura/gravação. Por padrão, apenas uma instância de banco de dados em um cluster do Aurora tem capacidade de leitura/gravação. Essa instância de banco de dados é chamada de instância principal. Se a instância principal original se tornar indisponível, o Aurora ativa um mecanismo de failover e uma outra instância de banco de dados se torna a principal.

De forma semelhante, com o direcionamento de instruções `SELECT` para o endpoint de leitura, você distribui o trabalho de processamento de consultas entre as instâncias de banco de dados no cluster. Cada conexão de leitor é atribuída a uma instância de banco de dados diferente usando a resolução round-robin do DNS. A realização da maior parte do trabalho de consulta nas réplicas de banco de dados somente leitura do Aurora reduz a carga da instância principal, liberando-a para lidar com instruções DDL e DML.

O uso desses endpoints reduz a dependência de nomes de host embutidos em código, e ajuda seu aplicativo a se recuperar mais rapidamente de falhas na instância de banco de dados.

> **Note**
>
> O Aurora também tem endpoints personalizados criados por você. Esses endpoints não são necessários durante uma prova de conceito.

As réplicas do Aurora estão sujeitas a um atraso, embora esse atraso normalmente seja de 10 a 20 milissegundos. Você pode monitorar o atraso da replicação e decidir se ele está dentro do intervalo de seus requisitos de consistência de dados. Em alguns casos, suas consultas de leitura podem exigir consistência forte de leitura (consistência de leitura após gravação). Nesses casos, você pode continuar a usar o endpoint do cluster para elas e não o endpoint de leitor.

Pera beneficiar-se completamente das capacidades do Aurora para execução paralela distribuída, pode ser necessário alterar a lógica da conexão. O objetivo é evitar o envio de todas as solicitações de leitura para a instância principal. As réplicas somente leitura do Aurora ficam em espera, com todos os mesmos dados, pronta para tratar instruções `SELECT`. Codifique a lógica de seu aplicativo para usar o endpoint adequado para cada tipo de operação. Siga estas diretrizes gerais:

- Evite usar uma única string de conexão embutida em código para todas as sessões do banco de dados.
- Se for prático, coloque as operações de gravação, como instruções DDL e DML, em funções em seu código de aplicativo cliente. Dessa forma, você pode fazer com que diferentes tipos de operações usem conexões específicas.
- Faça funções separadas para operações de consulta. O Aurora atribui cada nova conexão ao endpoint leitor a uma réplica diferente do Aurora para balancear a carga para aplicações com uso intensivo de leitura.
- Para operações que envolvem conjuntos de consultas, feche e reabra a conexão ao endpoint do leitor quando cada conjunto de consultas relacionadas for concluído. Use pooling de conexão se esse recurso estiver disponível em sua pilha de software. O direcionamento de consultas para diferentes conexões ajuda o Aurora a distribuir a workload de leitura entre as instâncias de banco de dados no cluster.

Para obter informações gerais sobre o gerenciamento de conexões e endpoints do Aurora, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora. Para se aprofundar no assunto, consulte o Manual do administrador do banco de dados Aurora MySQL – gerenciamento de conexões

10. Execute sua workload

Depois que as definições do esquema, dos dados e da configuração estiverem estabelecidas, você pode exercitar o cluster executando sua workload. Use uma workload na prova de conceito que espelhe os aspectos principais de suas workload de produção. Recomendamos que a tomada de decisões sobre a performance sempre seja feita usando testes e workloads reais em vez de benchmarking sintético, como o sysbench e o TPC-C. Sempre que for prático, recolha medidas com base em seu próprio esquema, padrões de consulta e volume de uso.

No máximo possível, replique as condições atuais sob as qual o aplicativo será executado. Por exemplo, normalmente, você executa seu código de aplicação em instâncias do Amazon EC2

na mesma região da AWS e na mesma nuvem privada virtual (VPC) que o cluster do Aurora. Se seu aplicativo de produção executar em várias instâncias do EC2 abrangendo várias zonas de disponibilidade, configure o ambiente de sua prova de conceito da mesma forma. Para obter mais informações sobre regiões da AWS, consulte Regiões e zonas de disponibilidade no Guia do usuário da Amazon RDS. Para saber mais sobre o serviço de Amazon VPC, consulte O que é a Amazon VPC? no Guia do usuário da Amazon VPC.

Depois que você verificar se os recursos básicos de seu aplicativo funcionam e puder acessar os dados por meio do Aurora, você poderá exercitar os aspectos do cluster do Aurora. Alguns recursos que você pode tentar são conexões simultâneas com balanceamento de carga, transações simultâneas e replicação automática.

Neste ponto, os mecanismos de transferência de dados já devem ser familiares e, portanto, você pode executar testes com uma proporção maior de dados de amostra.

Este estágio é quando você pode ver os efeitos da alteração da definição da configuração, como os limites de memória e os limites de conexão. Reexamine os procedimentos que você explorou em 8. Especifique definições da configuração.

Você também pode experimentar com mecanismos, como a criação e a restauração de snapshots. Por exemplo, é possível criar clusters com diferentes classes de instância da AWS, números de réplicas da AWS e assim por diante. Depois, em cada cluster, você pode restaurar o mesmo snapshot que contém o esquema e todos os dados. Para obter detalhes desse ciclo, consulte Criar um snapshot de cluster de banco de dados e Restauração de um snapshot de um cluster de banco de dados.

11. Avalie a performance

As melhores práticas desta área são desenvolvidas para garantir que todas as ferramentas e processos corretos estejam configurados para isolar rapidamente comportamentos anormais durante operações de workload. Elas também são configuradas para ver que você pode identificar com confiança todas as causas aplicáveis.

Sempre é possível ver o estado atual do cluster ou examinar tendências ao longo do tempo examinando a guia Monitoring (Monitoramento). Essa guia está disponível na página de detalhes do console para cada cluster ou instância de banco de dados Aurora. Ela exibe as métricas do serviço de monitoramento do Amazon CloudWatch na forma de gráficos. As métricas podem ser filtradas por nome, por instância de banco de dados e por período.

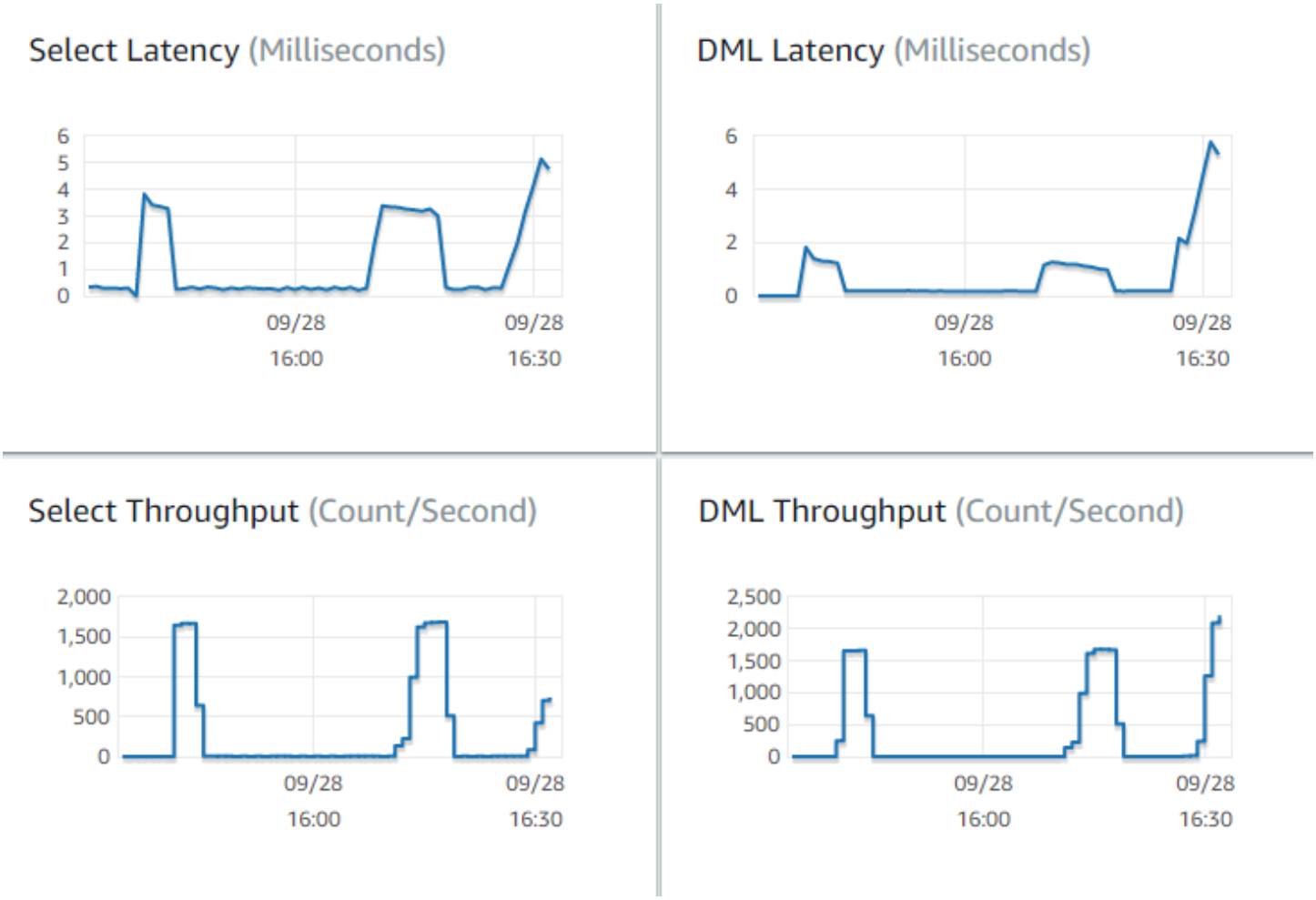
Para ter mais opções na guia Monitoring (Monitoramento), habilite o Monitoramento avançado e o Performance Insights nas configurações do cluster. Essas opções também podem ser habilitadas posteriormente, se você não as escolheu ao configurar o cluster.

Para medir a performance, você depende na maior parte dos gráficos que mostram as atividades de todo o cluster do Aurora. É possível verificar se as réplicas do Aurora têm carga e tempos de resposta semelhantes. Você também pode ver como o trabalho é dividido entre a instância principal de leitura/gravação e as réplicas de somente leitura do Aurora. Se houver desequilíbrio entre as instâncias de banco de dados ou um problema que afeta apenas uma instância de banco de dados, você poderá examinar a guia Monitoring (Monitoramento) da instância específica.

Depois de configurar o ambiente e a workload real para emular seu aplicativo de produção, você poderá medir se a performance do Aurora está boa. As questões mais importantes a serem respondidas são:

- Quantas consultas por segundo o Aurora está processando? Você pode examinar as métricas de Throughput (Taxa de transferência) para ver os valores de vários tipos de operações.
- Quanto tempo é necessário, em média, para o Aurora processar uma determinada consulta? Você pode examinar as métricas de Latency (Latência) para ver os valores de vários tipos de operações.

Para visualizar as métricas de throughput e de latência, confira a guia Monitoramento de determinado cluster do Aurora no console do Amazon RDS. A captura de tela a seguir mostra um exemplo das métricas Selecione Latência, Latência DML, Selecione o throughput e Taxa de transferência DML na guia Monitoramento.



Se possível, estabeleça valores de linha de base para essas métricas em seu ambiente atual. Se isso não for possível, crie uma linha de base no cluster do Aurora executando uma workload equivalente para seu aplicativo de produção. Por exemplo, execute sua workload do Aurora com um número semelhante de usuários e consultas semelhantes. Observe como os valores são alterados conforme você experimenta com diferentes classes de instância, tamanho de cluster, definições da configuração e assim por diante.

Se os números da taxa de transferência forem menores que o esperado, investigue mais os fatores que afetam a performance do banco de dados para sua workload. De forma semelhante, se os números de latência forem mais altos do que o esperado, investigue mais. Para isso, monitore as métricas secundárias do servidor de banco de dados (CPU, memória etc.) Você pode ver se as instâncias de banco de dados estão próximas de seus limites. Você também pode ver a quantidade de capacidade extra que suas instâncias de banco de dados têm para lidar com mais consultas simultâneas, consultas em tabelas grandes e assim por diante.

> **Tip**
>
> Para detectar os valores das métricas que estão além dos intervalos esperados, configure alarmes do CloudWatch.

Ao avaliar o tamanho ideal do cluster do Aurora e da capacidade, você pode localizar a configuração que atinge picos de performance do aplicativo sem provisionar os recursos em excesso. Um fator importante é encontrar o tamanho adequado para as instâncias de banco de dados no cluster do Aurora. Comece selecionando um tamanho de instância que tenha capacidade semelhante de CPU e memória para seu ambiente de produção atual. Colete os números de taxa de transferência e latência para a workload desse tamanho de instância. Aumente a instância para o próximo tamanho maior. Verifique se os números de taxa de transferência e latência melhoram. Também reduza o tamanho da instância e verifique se os números de latência e taxa de transferência permanecem os mesmos. Sua meta é obter a taxa de transferência mais alta, com a latência mais baixa, na menor instância possível.

> **Tip**
>
> Dimensione seus clusters do Aurora e as instâncias de banco de dados associadas com capacidade existente suficiente para lidar com picos de tráfego repentino e imprevisíveis. Para bancos de dados de missão crítica, deixe pelo menos 20 por cento de capacidade adicional para CPU e memória.

Execute testes de performance longos o suficiente para medir a performance do banco de dados em um estado quente e estável. Pode ser necessário executar a workload por muitos minutos ou por até algumas horas para atingir o estado estável. No início de uma execução, é normal ter alguma variação. Essa variação ocorre porque cada réplica do Aurora aquece seus caches com base nas consultas `SELECT` que trata.

O Aurora executa melhor com workloads transacionais que envolvem vários usuários e consultas simultâneos. Para garantir que você está direcionando carga suficiente para performance ideal, execute comparações que usam multithreading ou execute várias instâncias dos testes de performance simultaneamente. Meça a performance com centenas ou até milhares de threads de clientes simultâneos. Simule o número de threads simultâneos que você espera ter em seu ambiente de produção. Também é possível executar testes de stress adicionais com mais threads para medir a escalabilidade do Aurora.

12. Exercite a alta disponibilidade do Aurora

Muitos dos principais recursos do Aurora envolvem alta disponibilidade. Esses recursos incluem replicação automática, failover automático, backups automáticos com restauração point-in-time e habilidade de adicionar instâncias de banco de dados ao cluster. A segurança e a confiabilidade de recursos como esses são importantes para aplicativos de missão crítica.

Para avaliar esses recursos é necessário ter uma certa visão. Em atividades anteriores, como na medição de performance, você observa com o sistema desempenha quando tudo funciona corretamente. O teste de alta disponibilidade exige que você imagine o comportamento de pior caso. Você deve considerar vários tipos de falhas, mesmo que essas condições sejam raras. Você deve introduzir problemas intencionalmente para ter certeza de que o sistema se recupera correta e rapidamente.

> **Tip**
>
> Para uma prova de conceito, configure todas as instâncias de banco de dados em um cluster do Aurora com a mesma classe de instância da AWS. Isso possibilita testar os recursos de disponibilidade do Aurora sem maiores alterações na performance e na escalabilidade quando você desativar instâncias de banco de dados para simular falhas.

Recomendamos usar pelo menos duas instâncias em cada cluster do Aurora. As instâncias de banco de dados em um cluster do Aurora podem abranger até três zonas de disponibilidade (AZs). Localize cada uma das duas ou três primeiras instâncias de banco de dados em uma AZ diferente. Quando começar a usar clusters maiores, espalhe as instâncias de banco de dados em todas as AZs em sua região da AWS. Isso aumenta a capacidade de tolerância a falhas. Mesmo que um problema afete uma AZ inteira, o Aurora poderá fazer failover para uma instância de banco de dados em outra AZ. Se você executar um cluster com mais de três instâncias, distribua as instâncias de banco de dados o mais uniformemente quanto possível sobre todas as três AZs.

> **Tip**
>
> O armazenamento de um cluster do Aurora é independente das instâncias de banco de dados. O armazenamento de cada cluster do Aurora sempre abrange três AZs.
> Ao testar recursos de alta disponibilidade, sempre use instâncias de banco de dados com capacidade idêntica em seu cluster de teste. Isso evita alterações imprevisíveis na

performance, na latência e assim por diante sempre que uma instância de banco de dados toma o lugar de outra.

Para saber mais sobre como simular condições de falha para testar recursos de alta disponibilidade, consulte. Testar o Amazon Aurora MySQL usando consultas de injeção de falhas.

Como parte do exercício de prova de conceito, um objetivo é encontrar o número ideal de instâncias de banco de dados e a classe de instância ideal para essas instâncias de banco de dados. Fazer isso exige o balanceamento dos requisitos de alta disponibilidade e performance.

No Aurora, quando mais instâncias de banco de dados você tiver em um cluster, maiores serão os benefícios para a alta disponibilidade. Ter mais instâncias de banco de dados também melhora a escalabilidade de aplicações com uso intensivo de leitura. O Aurora pode distribuir várias conexões para consultas `SELECT` entre as réplicas apenas para leitura do Aurora.

Por outro lado, a limitação do número de instâncias de banco de dados reduz o tráfego da replicação no nó principal. O tráfego de replicação consome largura de banda de rede, que é outro aspecto da performance e da escalabilidade gerais. Portanto, para aplicativos OLTP com uso intensivo de gravação, prefira ter um número menor de instâncias de banco de dados grandes em vez de ter muitas instâncias de banco de dados pequenas.

Em um cluster típico do Aurora, uma instância de banco de dados (a instância principal) trata todas as instruções DDL e DML. As outras instâncias de banco de dados (as réplicas do Aurora) tratam apenas de instruções `SELECT`. Embora as instâncias de banco de dados não executem exatamente a mesma quantidade de trabalho, recomendamos usar a mesma classe de instância para todas as instâncias de banco de dados no cluster. Dessa forma, se ocorrer uma falha, e o Aurora promover uma das instâncias de banco de dados somente leitura para que seja a nova instância principal, a instância principal terá a mesma capacidade que a anterior.

Se você precisar usar instâncias de banco de dados de diferentes capacidades no mesmo cluster, configure camadas de failover para as instâncias de banco de dados. Essas camadas determinam a ordem em que as réplicas do Aurora são promovidas pelo mecanismo de failover. Coloque as instâncias de banco de dados que forem muito maiores ou menores que as outras em uma camada de failover inferior. Isso garante que elas sejam escolhidas por último para promoção.

Exercite os recursos de recuperação de dados do Aurora, como restauração automática em um ponto no tempo, snapshots e restaurações manuais e retrocesso de cluster. Se for adequado, copie

snapshots em outras regiões da AWS e restaure em outras regiões da AWS para simular cenários de DR.

Investigue os requisitos de sua organização para o objetivo de tempo de restauração RTO (restore time objective), o objetivo de ponto de restauração (RPO - restore point objective) e a redundância geográfica. A maioria das organizações agrupam esses itens sob a ampla categoria de recuperação de desastres. Avalie os recursos de alta disponibilidade do Aurora nesta seção no contexto de seu processo de recuperação de desastres para garantir que seus requisitos de RTO e RPO sejam atendidos.

13. O que fazer em seguida

No final de um processo de prova de conceito bem-sucedido, você confirma que o Aurora é uma solução adequada para você com base na workload antecipada. Em todos os processos anteriores, você verificou como o Aurora funciona em um ambiente operacional realístico e o avaliou em relação a seus critérios de sucesso.

Depois que tiver seu ambiente de banco de dados ativo e em execução com o Aurora, você poderá avançar para etapas de avaliação mais detalhadas, resultando em sua migração final e implantação em produção. Dependendo de sua situação, essas outras etapas podem ou não serem incluídas no processo de prova de conceito. Para obter detalhes sobre as atividades de migração e de portabilidade, consulte o whitepaper da AWS Aurora migration handbook (Manual de migração).

Em outra próxima etapa, considere as configurações de segurança relevantes para sua workload e projetadas para atender a seus requisitos de segurança em um ambiente de produção. Planeje os controles a serem estabelecidos para proteger o acesso às credenciais do usuário mestre do cluster do Aurora. Defina as funções e responsabilidades dos usuários do banco de dados para controlar o acesso aos dados armazenados no cluster do Aurora. Considere os requisitos de acesso ao banco de dados para aplicativos, scripts e ferramentas ou serviços de terceiros. Explore os serviços e os recursos da AWS, como a autenticação do AWS Secrets Manager e do AWS Identity and Access Management (IAM).

Neste ponto, você deve entender os procedimentos e as melhores práticas para executar testes de comparação com o Aurora. Você pode descobrir que é necessário fazer ajuste de performance adicional. Para obter detalhes, consulte Como gerenciar a performance e a escalabilidade de clusters de banco de dados do Aurora, Melhorias de performance do Amazon Aurora MySQL, Performance e escalabilidade do Amazon Aurora PostgreSQL e Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora. Ao fazer ajuste adicional, esteja familiarizado com as

métricas que coletou durante a prova de conceito. Em uma próxima etapa, você pode criar novos clusters com diferentes opções de definições de configuração, mecanismo de banco de dados e versão de banco de dados. Ou você pode criar tipos especializados de clusters do Aurora para que correspondam às necessidades de casos de uso específicos.

Por exemplo, você pode explorar os clusters de consulta paralela do Aurora para aplicativos de processamento transacional/analítico híbrido (HTAP - hybrid transaction/analytical processing). Se uma distribuição geográfica ampla for essencial para a recuperação de desastres ou para minimizar a latência, você poderá explorar os bancos de dados globais do Aurora. Se sua workload for intermitente ou se você estiver usando o Aurora em um cenário de desenvolvimento/teste, você poderá explorar os clusters do Aurora Serverless.

Seus clusters de produção também podem precisar lidar com volumes altos de conexões de entrada. Para saber mais sobre essas técnicas, consulte o whitepaper da AWSAurora MySQL database administrator's handbook – Connection management (Manual do administrador do banco de dados Aurora MySQL: Gerenciamento de conexões).

Se, depois da prova de conceito, você decidir que seu caso de uso não é adequado para o Aurora, considere estes outros serviços da AWS:

- Para casos de uso puramente analíticos, as workloads se beneficiam de um formato de armazenamento colunar e de outros recursos mais adequados às workloads OLAP. Os serviços do AWS que abordam esses casos de uso incluem o seguinte:
  - Amazon Redshift
  - Amazon EMR
  - Amazon Athena
- Muitas workloads se beneficiam de uma combinação do Aurora com um ou mais desses serviços. Você pode mover dados entre esses serviços usando:
  - AWS Glue
  - AWS DMS
  - Importar do Amazon S3, conforme descrito no Guia do usuário do Amazon Aurora
  - Exportar para o Amazon S3, conforme descrito no Guia do usuário do Amazon Aurora
  - Muitas outras ferramentas de ETL populares

Segurança no Amazon Aurora

A segurança na nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de um data center e de uma arquitetura de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos programas de conformidade da AWS. Para saber mais sobre os programas de compatibilidade que se aplicam ao Amazon Aurora (Aurora), consulte Serviços da AWS no escopo pelo programa de conformidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, inclusive a confidencialidade dos dados, os requisitos da organização, as leis e as regulamentações vigentes.

Essa documentação ajuda a entender como aplicar o Modelo de Responsabilidade Compartilhada ao usar o Amazon Aurora. Os tópicos a seguir mostram como configurar o Amazon Aurora para atender aos seus objetivos de segurança e conformidade. Saiba também como usar outros serviços da AWS que ajudam a monitorar e proteger os recursos do Amazon Aurora.

Você pode gerenciar o acesso a seus recursos do Amazon Aurora e a seus bancos de dados em um cluster de banco de dados. O método usado para gerenciar o acesso depende do tipo de tarefa que o usuário precisa realizar com o Amazon Aurora:

- Execute o cluster de banco de dados em uma nuvem privada virtual (VPC) baseada no serviço da Amazon VPC para obter o maior controle possível de acesso à rede. Para ter mais informações sobre como criar um cluster de banco de dados em uma VPC, consulte Amazon VPC e Amazon Aurora.
- Use políticas do AWS Identity and Access Management (IAM) para atribuir permissões que determinam quem tem permissão para gerenciar os recursos do Amazon Aurora. Por exemplo, você pode usar o IAM para determinar quem tem permissão para criar, descrever, modificar e excluir clusters de banco de dados, marcar recursos ou modificar grupos de segurança.

Para examinar exemplos de política do IAM, consulte Exemplos de políticas baseadas em identidade do Amazon Aurora.

- Use grupos de segurança para controlar quais endereços IP ou instâncias do Amazon EC2 podem se conectar aos seus bancos de dados em um cluster de banco de dados. Quando você cria um cluster de banco de dados, seu firewall impede qualquer acesso ao banco de dados, exceto por meio de regras especificadas por um grupo de segurança associado.
- Use conexões Secure Socket Layer (SSL) ou Transport Layer Security (TLS) com clusters de banco de dados que executam o Aurora MySQL ou o Aurora PostgreSQL. Para ter mais informações sobre como usar o SSL/TLS com um cluster de banco de dados, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.
- Use a criptografia do Amazon Aurora para proteger seus clusters de banco de dados e snapshots em repouso. A criptografia do Amazon Aurora usa o algoritmo de criptografia AES-256 padrão do setor para criptografar seus dados no servidor que hospeda o cluster de banco de dados. Para ter mais informações, consulte Criptografar recursos do Amazon Aurora .
- Use os recursos de segurança do seu mecanismo de banco de dados para controlar quem pode fazer login nos bancos de dados em um cluster de banco de dados. Esses recursos funcionam como se o banco de dados estivesse em sua rede local.

  Para ter informações sobre segurança com o Aurora MySQL, consulte Segurança com o Amazon Aurora MySQL. Para ter informações sobre segurança com o Aurora PostgreSQL, consulte Segurança com o Amazon Aurora PostgreSQL.

O Aurora faz parte do serviço de banco de dados gerenciado Amazon Relational Database Service (Amazon RDS). O Amazon RDS é um serviço da Web que facilita a configuração, operação e escala de um banco de dados relacional na nuvem. Se você não estiver familiarizado com o Amazon RDS, consulte o Guia do usuário do Amazon RDS.

O Aurora inclui um subsistema de armazenamento de alta performance. Seus mecanismos de banco de dados compatíveis com o MySQL e o PostgreSQL são personalizados para tirar proveito do rápido armazenamento distribuído. O Aurora também automatiza e padroniza a clusterização e a replicação de bancos de dados que, normalmente, são os aspectos mais desafiantes da configuração e da administração de bancos de dados.

Tanto no Amazon RDS como no Aurora, você pode acessar a API do RDS programaticamente, além de usar a AWS CLI para acessar interativamente a API do RDS. Algumas operações da API do RDS e comandos da AWS CLI se aplicam tanto ao Amazon RDS como ao Aurora, enquanto outras

se aplicam ao Amazon RDS ou ao Aurora. Para obter informações sobre as operações da API do RDS, consulte Referência de API do Amazon RDS. Para obter mais informações sobre a AWS CLI, consulte a Referência da AWS Command Line Interface para o Amazon RDS.

> **Note**
>
> Basta configurar a segurança para os casos de uso. Não é necessário configurar o acesso de segurança para os processos que o Amazon Aurora gerencia. Isso inclui a criação de backups, o failover automático e outros processos.

Para ter mais informações sobre como gerenciar o acesso a recursos do Amazon Aurora e os bancos de dados de um cluster de banco de dados, consulte os tópicos a seguir.

Tópicos

- Autenticação de banco de dados com o Amazon Aurora
- Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager
- Proteção de dados no Amazon RDS
- Gerenciamento de identidade e acesso no Amazon Aurora
- Registrar em log e monitorar no Amazon Aurora
- Validação de conformidade do Amazon Aurora
- Resiliência no Amazon Aurora
- Segurança da infraestrutura no Amazon Aurora
- API do Amazon RDS e endpoints da VPC de interface (AWS PrivateLink)
- Práticas recomendadas de segurança do Amazon Aurora
- Controlar acesso com grupos de segurança
- Privilégios da conta de usuário mestre
- Usar funções vinculadas ao serviço do Amazon Aurora
- Amazon VPC e Amazon Aurora

Autenticação de banco de dados com o Amazon Aurora

O Amazon Aurora permite autenticar usuários do banco de dados de várias maneiras.

A autenticação por senha está disponível por padrão para todos os clusters de banco de dados. No Aurora MySQL e no Aurora PostgreSQL, você também pode adicionar a autenticação de banco de dados do IAM e a autenticação Kerberos (uma ou ambas) para o mesmo cluster de banco de dados.

As autenticações de banco de dados do IAM, do Kerberos e por senha usam diferentes métodos de autenticação no banco de dados. Portanto, um usuário específico pode fazer login em um banco de dados usando apenas um método de autenticação.

No PostgreSQL, use apenas uma das seguintes configurações de função para um usuário de um banco de dados específico:

- Para usar a autenticação de banco de dados do IAM, atribua a função `rds_iam` ao usuário.
- Para usar a autenticação do Kerberos, atribua a função `rds_ad` ao usuário.
- Para usar a autenticação por senha, não atribua as funções `rds_iam` ou `rds_ad` ao usuário.

Não atribua ambas as funções `rds_iam` e `rds_ad` a um usuário de um banco de dados PostgreSQL direta ou indiretamente por acesso de concessão aninhado. Se a função `rds_iam` for adicionada ao usuário mestre, a autenticação do IAM terá precedência sobre a autenticação por senha, então o usuário mestre terá que fazer login como um usuário do IAM.

> **Important**
>
> É altamente recomendável não usar o usuário mestre diretamente nas aplicações. Em vez disso, siga as práticas recomendadas de usar um usuário do banco de dados criado com os privilégios mínimos obrigatórios para a aplicação.

Tópicos

- Autenticação com senha
- Autenticação do banco de dados do IAM
- Autenticação de Kerberos

Autenticação com senha

Com a autenticação com senha, seu banco de dados executa toda a administração de contas de usuário. Crie usuários com instruções SQL, como `CREATE USER`, usando a cláusula apropriada exigida pelo mecanismo de banco de dados para especificar senhas. Por exemplo, no MySQL, a

instrução é `CREATE USER name IDENTIFIED BY password`, enquanto, no PostgreSQL, ela é `CREATE USER name WITH PASSWORD password`.

Com a autenticação com senha, seu banco de dados controla e autentica contas de usuário. Se um mecanismo de banco de dados tiver recursos de gerenciamento de senhas fortes, ele poderá aumentar a segurança. A autenticação de banco de dados pode ser mais fácil de administrar usando autenticação com senha quando você tem pequenas comunidades de usuários. Como as senhas de texto não criptografado são geradas nesse caso, a integração com o AWS Secrets Manager pode aumentar a segurança.

Para ter informações sobre como usar o Secrets Manager com o Amazon Aurora, consulte Creating a basic secret e Rotating secrets for supported Amazon RDS databases no Guia do usuário do AWS Secrets Manager. Para obter informações sobre como recuperar os segredos de forma programática nas aplicações personalizadas, consulte Recuperar o valor do segredo no Guia do usuário do AWS Secrets Manager.

Autenticação do banco de dados do IAM

Você pode se autenticar no cluster de banco de dados usando a autenticação de banco de dados do AWS Identity and Access Management (IAM). Com esse método de autenticação, você não precisa usar uma senha ao conectar-se a um cluster de banco de dados. Em vez disso, você usa um token de autenticação.

Para ter mais informações sobre autenticação de banco de dados do IAM, incluindo informações sobre disponibilidade de mecanismos de banco de dados específicos, consulte Autenticação do banco de dados do IAM.

Autenticação de Kerberos

O Amazon Aurora oferece suporte à autenticação externa de usuários de banco de dados usando o Kerberos e o Microsoft Active Directory. O Kerberos é um protocolo de autenticação de rede que usa tíquetes e criptografia de chave simétrica para eliminar a necessidade de transmitir senhas pela rede. O Kerberos foi integrado ao Active Directory e foi projetado para autenticar usuários em recursos de rede, como bancos de dados.

O suporte do Amazon Aurora ao Kerberos e ao Active Directory oferece os benefícios de autenticação única e autenticação centralizada dos usuários do banco de dados. Você pode manter suas credenciais de usuário no Active Directory. O Active Directory oferece um lugar centralizado para armazenar e gerenciar credenciais para vários clusters de banco de dados.

Para usar as credenciais do Active Directory autogerenciado, é necessário configurar uma relação de confiança com o Directory Service para Microsoft Active Directory ao qual o cluster de banco de dados está associado.

O Aurora PostgreSQL e o Aurora MySQL comportam relacionamentos de confiança florestal unidirecionais e bidirecionais com autenticação em toda a floresta ou autenticação seletiva.

Em alguns casos, é possível configurar a autenticação Kerberos em uma relação de confiança externa. Isso exige que o Active Directory autogerenciado tenha configurações adicionais. Isso inclui, entre outros, o Kerberos Forest Search Order.

O Aurora é compatível com a autenticação Kerberos para clusters de banco de dados do Aurora MySQL e do Aurora PostgreSQL. Para obter mais informações, consulte Usar a autenticação Kerberos para Aurora MySQL e Usar a autenticação Kerberos com o Aurora PostgreSQL .

Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager

O Amazon Aurora integra-se ao Secrets Manager para gerenciar senhas do usuário principal para suas .

Tópicos

- Disponibilidade de regiões e versões
- Limitações da integração do Secrets Manager com o Amazon Aurora
- Visão geral do gerenciamento de senhas do usuário principal com AWS Secrets Manager
- Benefícios do gerenciamento de senhas do usuário principal com o Secrets Manager
- Permissões necessárias para a integração do Secrets Manager
- Impor o gerenciamento da senha do usuário principal pelo Aurora no AWS Secrets Manager
- Gerenciar a senha do usuário principal para um cluster de banco de dados com o Secrets Manager
- Alternar o segredo da senha do usuário principal para um cluster de banco de dados
- Visualizar os detalhes sobre um segredo para um cluster de banco de dados

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados e entre Regiões da AWS. Para ter mais informações sobre a disponibilidade de versões e regiões com a integração do Secrets Manager com o Amazon Aurora, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a integração com o Secrets Manager.

Limitações da integração do Secrets Manager com o Amazon Aurora

O gerenciamento de senhas do usuário principal com o Secrets Manager não é compatível com os seguintes recursos:

- Implantações azul/verde do Amazon RDS
- Clusters de banco de dados que fazem parte de um banco de dados global do Aurora.
- Aurora Serverless v1Clusters de banco de dados do

- Réplicas de leitura entre regiões
- Replicação externa de logs binários

Visão geral do gerenciamento de senhas do usuário principal com AWS Secrets Manager

Com o AWS Secrets Manager, é possível substituir credenciais codificadas em seu código, inclusive senhas de banco de dados, por uma chamada de API ao Secrets Manager para recuperar o segredo de forma programática. Para ter mais informações sobre o Secrets Manager, consulte o Guia do usuário do AWS Secrets Manager.

Quando você armazena segredos de banco de dados no Secrets Manager, sua Conta da AWS incorre em cobranças. Para obter mais informações sobre definição de preços, consulte Definição de preço do AWS Secrets Manager.

Você pode especificar que o Aurora gerencie a senha de usuário principal no Secrets Manager para um cluster de banco de dados do Amazon Aurora ao realizar uma das seguintes operações:

- Criar um cluster de banco de dados
- Modificar um cluster de banco de dados
- Restaurar um cluster de banco de dados do Amazon S3 (somente Aurora MySQL)

Quando você especifica que o Aurora gerencie a senha do usuário principal no Secrets Manager, o Aurora gera a senha e a armazena no Secrets Manager. Você pode interagir diretamente com o segredo para recuperar as credenciais do usuário principal. Você também pode especificar uma chave gerenciada pelo cliente para criptografar o segredo ou usar a chave do KMS fornecida pelo Secrets Manager.

O Aurora gerencia as configurações do segredo e o alterna a cada sete dias por padrão. Você pode modificar algumas configurações, como o cronograma de alternância. Se você excluir um cluster de banco de dados que gerencie um segredo no Secrets Manager, o segredo e seus metadados associados também serão excluídos.

Para conectar-se a uma com as credenciais em um segredo, você pode recuperar o segredo do Secrets Manager. Para ter mais informações, consulte Recuperar segredos do AWS Secrets Manager e Conecte-se a um banco de dados SQL com credenciais em um segredo do AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

Benefícios do gerenciamento de senhas do usuário principal com o Secrets Manager

O gerenciamento de senhas do usuário principal pelo Aurora com o Secrets Manager oferece os seguintes benefícios:

- O Aurora gera automaticamente credenciais de banco de dados.
- O Aurora armazena e gerencia automaticamente as credenciais do banco de dados no AWS Secrets Manager.
- O Aurora alterna as credenciais do banco de dados regularmente, sem exigir alterações na aplicação.
- O Secrets Manager protege as credenciais do banco de dados do acesso humano e da visualização de texto sem formatação.
- O Secrets Manager permite a recuperação de credenciais do banco de dados em segredos para conexões de banco de dados.
- O Secrets Manager permite um controle refinado do acesso às credenciais do banco de dados em segredos com o uso do IAM.
- Você também pode separar a criptografia do banco de dados da criptografia de credenciais com chaves do KMS diferentes.
- É possível eliminar o gerenciamento e a alternância manuais das credenciais do banco de dados.
- Você pode monitorar facilmente as credenciais do banco de dados com o AWS CloudTrail e o Amazon CloudWatch.

Para obter mais informações sobre os benefícios do Secrets Manager, consulte o Guia do usuário do AWS Secrets Manager.

Permissões necessárias para a integração do Secrets Manager

Os usuários devem ter as permissões necessárias para realizar operações relacionadas à integração do Secrets Manager. É possível criar políticas do IAM que concedam permissões para realizar operações de API específicas nos recursos especificados necessários. Depois, você pode anexar essas políticas aos conjuntos de permissões do IAM ou às funções que exigem essas permissões. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

Para operações de criação, modificação ou restauração, o usuário que especifica que o Aurora gerencie a senha do usuário principal no Secrets Manager deve ter permissões para realizar as seguintes operações:

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

A permissão `kms:DescribeKey` é necessária para acessar sua chave gerenciada pelo cliente para o `MasterUserSecretKmsKeyId` e para descrever `aws/secretsmanager`.

Para operações de criação, modificação ou restauração, o usuário que especifica a senha do usuário principal para criptografar o segredo no Secrets Manager deve ter permissões para realizar as seguintes operações:

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`

Para operações de modificação, o usuário que alterna a senha de usuário principal no Secrets Manager deve ter permissões para realizar a seguinte operação:

- `secretsmanager:RotateSecret`

Impor o gerenciamento da senha do usuário principal pelo Aurora no AWS Secrets Manager

Você pode usar as chaves de condição do IAM para impor o gerenciamento pelo Aurora da senha do usuário principal no AWS Secrets Manager. A política a seguir não permite que os usuários criem nem restaurem instâncias de banco de dados ou clusters de banco de dados ou a menos que a senha do usuário principal seja gerenciada pelo Aurora no Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> Esta política impõe o gerenciamento de senhas no AWS Secrets Manager no momento da criação. No entanto, você ainda pode desativar a integração do Secrets Manager e definir manualmente uma senha principal modificando o cluster.
> Para evitar isso, inclua `rds:ModifyDBInstance` e `rds:ModifyDBCluster` no bloco "Ação" da política. Esteja ciente de que isso impede que o usuário aplique quaisquer modificações adicionais aos clusters existentes que não têm a integração com o Secrets Manager habilitada.

Para ter mais informações sobre como usar as chaves de condição em políticas do IAM, consulte Chaves de condição de políticas do Aurora e Políticas de exemplo: usar chaves de condição.

Gerenciar a senha do usuário principal para um cluster de banco de dados com o Secrets Manager

Você pode configurar o gerenciamento pelo Aurora da senha do usuário principal no Secrets Manager ao realizar as seguintes ações:

- Criar um cluster de bancos de dados do Amazon Aurora
- Modificar um cluster de bancos de dados Amazon Aurora

- Migrar dados de um banco de dados MySQL externo para um cluster de banco de dados do Amazon Aurora MySQL

Você pode usar o console do RDS, a AWS CLI ou a API do RDS para realizar essas ações.

Console

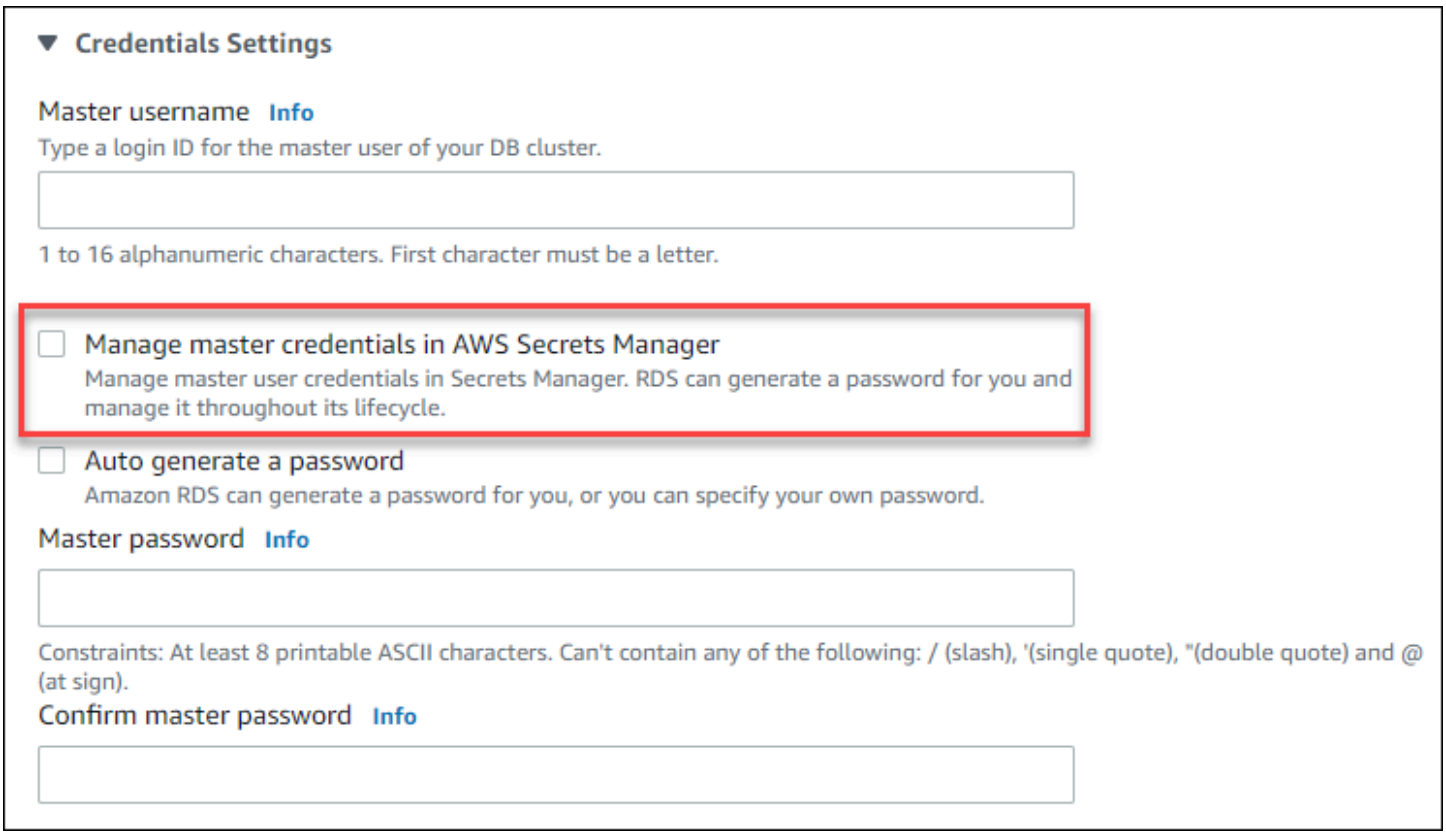
Siga as instruções para criar ou modificar um cluster de banco de dados com o console do RDS:

- Criar um cluster de banco de dados
- Modificar uma instância de banco de dados em um cluster de banco de dados

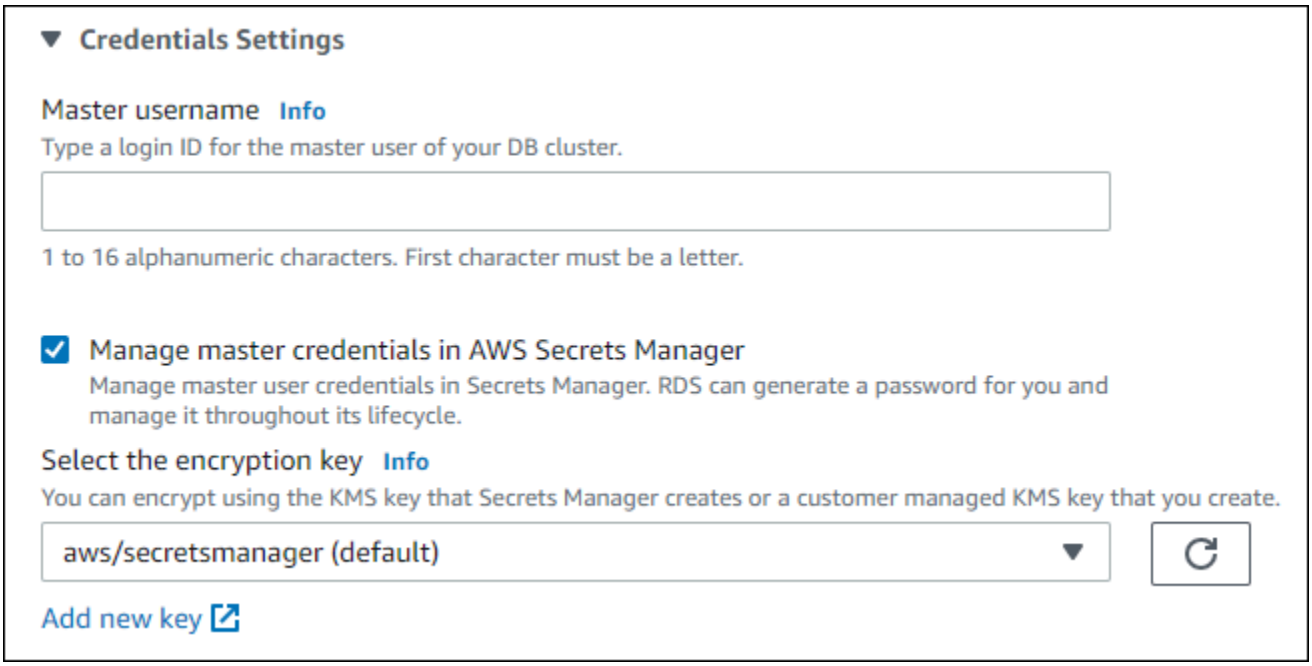
  No console do RDS, você pode modificar qualquer instância de banco de dados para especificar as configurações de gerenciamento de senha do usuário principal para todo o cluster de banco de dados.
- Restaurar um cluster de banco de dados do Amazon Aurora MySQL de um bucket do Amazon S3

Ao usar o console do RDS para realizar uma dessas operações, você pode especificar que a senha do usuário principal seja gerenciada pelo Aurora no Secrets Manager. Para fazer isso ao criar ou restaurar um cluster de banco de dados, selecione Gerenciar credenciais principais no AWS Secrets Manager em Configurações de credenciais. Quando estiver modificando um cluster de banco de dados, selecione Gerenciar credenciais principais no AWS Secrets Manager em Configurações.

A imagem a seguir é um exemplo da configuração Gerenciar credenciais principais no AWS Secrets Manager quando você está criando ou restaurando um cluster de banco de dados.



Quando você seleciona essa opção, o Aurora gera a senha do usuário principal e a gerencia durante todo o ciclo de vida no Secrets Manager.



Você pode optar por criptografar o segredo com uma chave do KMS fornecida pelo Secrets Manager ou com uma chave gerenciada pelo cliente criada por você. Depois que o Aurora estiver gerenciando

as credenciais de banco de dados de um cluster de banco de dados, não será possível alterar a chave do KMS usada para criptografar o segredo.

Você pode escolher outras configurações para atender às suas necessidades.

Para ter mais informações sobre as configurações disponíveis ao criar um cluster de banco de dados, consulte Configurações de clusters de bancos de dados do Aurora. Para ter mais informações sobre as configurações disponíveis ao modificar um cluster de banco de dados, consulte Configurações do Amazon Aurora.

AWS CLI

Para especificar que o Aurora gerencie a senha do usuário principal no Secrets Manager, especifique a opção `--manage-master-user-password` em um dos seguintes comandos:

- create-db-cluster
- modify-db-cluster
- restore-db-cluster-from-s3

Quando você especifica a opção `--manage-master-user-password` nesses comandos, o Aurora gera a senha do usuário principal e a gerencia durante todo o ciclo de vida no Secrets Manager.

Para criptografar o segredo, você pode especificar uma chave gerenciada pelo cliente ou usar a chave do KMS fornecida pelo Secrets Manager. Use a opção `--master-user-secret-kms-key-id` para especificar uma chave gerenciada pelo cliente. O identificador de chave do AWS KMS é o ARN da chave, o ID da chave, o ARN do alias ou o nome do alias da chave do KMS. Para usar uma chave do KMS em outra Conta da AWS, é necessário usar o ARN da chave ou o ARN do alias. Depois que o Aurora estiver gerenciando as credenciais de banco de dados de um cluster de banco de dados, não será possível alterar a chave do KMS usada para criptografar o segredo.

Você pode escolher outras configurações para atender às suas necessidades.

Para ter mais informações sobre as configurações disponíveis ao criar um cluster de banco de dados, consulte Configurações de clusters de bancos de dados do Aurora. Para ter mais informações sobre as configurações disponíveis ao modificar um cluster de banco de dados, consulte Configurações do Amazon Aurora.

Este exemplo cria um cluster de banco de dados e especifica que o Aurora gerencie a senha no Secrets Manager. O segredo é criptografado usando a chave do KMS fornecida pelo Secrets Manager.

Example

Para Linux, macOS ou Unix:

```
aws rds create-db-cluster \
     --db-cluster-identifier sample-cluster \
     --engine aurora-mysql \
     --engine-version 8.0 \
     --master-username admin \
     --manage-master-user-password
```

Para Windows:

```
aws rds create-db-cluster ^
     --db-cluster-identifier sample-cluster ^
     --engine aurora-mysql ^
     --engine-version 8.0 ^
     --master-username admin ^
     --manage-master-user-password
```

API do RDS

Para especificar que o Aurora gerencie a senha do usuário principal no Secrets Manager, defina o parâmetro `ManageMasterUserPassword` como `true` em uma das seguintes operações:

- CreateDBCluster
- ModifyDBCluster
- RestoreDBClusterFromS3

Quando você define o parâmetro `ManageMasterUserPassword` como `true` em uma dessas operações, o Aurora gera a senha do usuário principal e a gerencia durante todo o ciclo de vida no Secrets Manager.

Para criptografar o segredo, você pode especificar uma chave gerenciada pelo cliente ou usar a chave do KMS fornecida pelo Secrets Manager. Use o parâmetro `MasterUserSecretKmsKeyId` para especificar uma chave gerenciada pelo cliente. O identificador de chave do AWS KMS é o ARN da chave, o ID da chave, o ARN do alias ou o nome do alias da chave do KMS. Para usar uma chave do KMS em outra Conta da AWS, é necessário usar o ARN da chave ou o ARN do alias. Depois que o Aurora estiver gerenciando as credenciais de banco de dados de um cluster de banco de dados, não será possível alterar a chave do KMS usada para criptografar o segredo.

Alternar o segredo da senha do usuário principal para um cluster de banco de dados

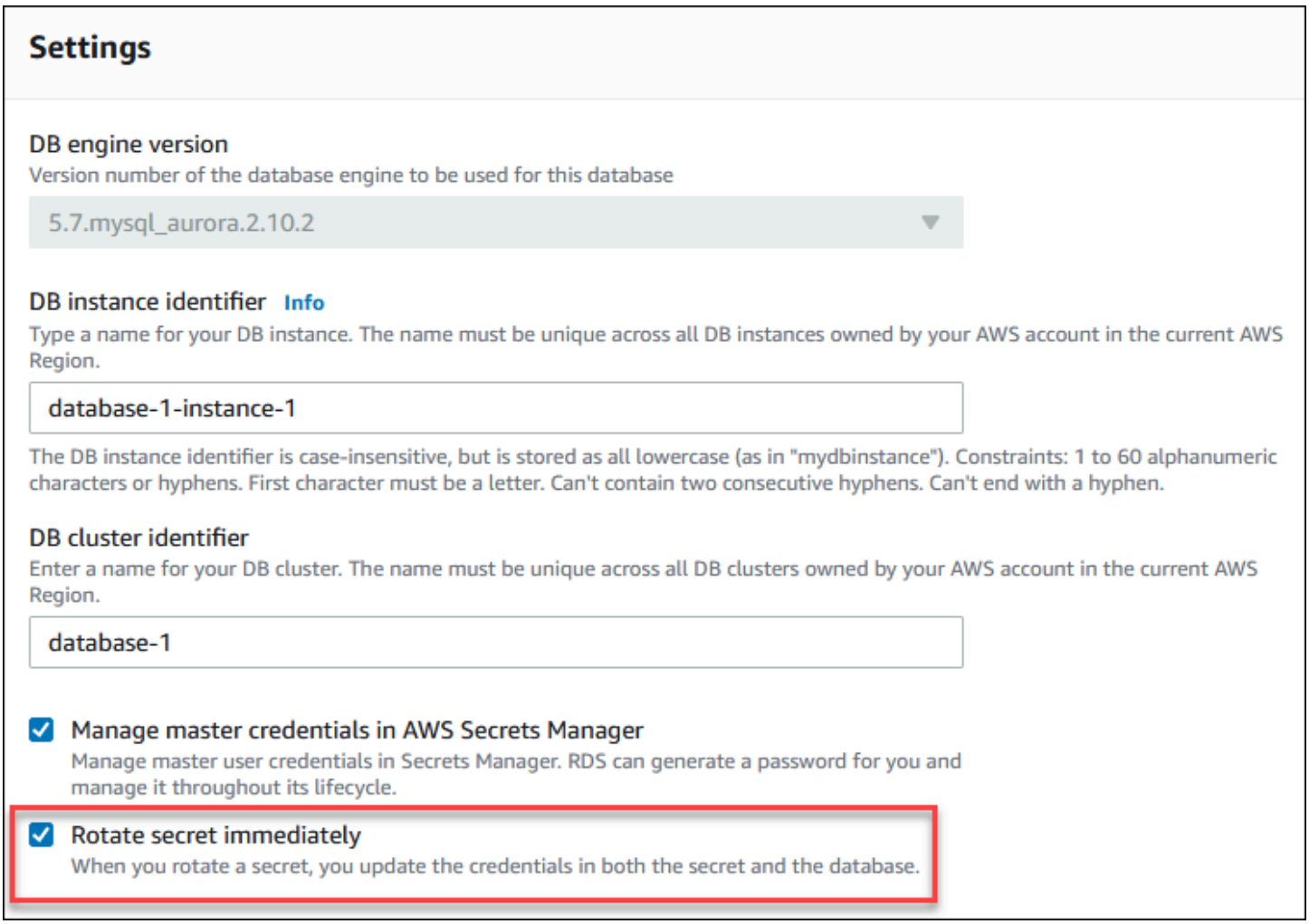
Quando o Aurora alterna um segredo de senha do usuário principal, o Secrets Manager gera uma nova versão para o segredo existente. A nova versão do segredo contém a nova senha do usuário principal. O Aurora altera a senha do usuário principal do cluster de banco de dados para corresponder à nova versão do segredo.

Você pode alternar um segredo imediatamente em vez de esperar por uma alternância programada. Para alternar o segredo de uma senha do usuário principal no Secrets Manager, modifique o cluster de banco de dados . Para obter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Você pode alternar o segredo de uma senha do usuário principal imediatamente com o console do RDS, a AWS CLI ou a API do RDS. A nova senha tem sempre 28 caracteres e contém pelo menos um caractere maiúsculo e um minúsculo, um número e um sinal de pontuação.

Console

Para alternar o segredo de uma senha do usuário principal usando o console do RDS, modifique o cluster de banco de dados e selecione Rotate secret immediately (Alternar segredo imediatamente) em Settings (Configurações).



Siga as instruções para modificar um cluster de banco de dados com o console do RDS em Modificar o cluster de banco de dados usando o console, a CLI e a API. Você deve escolher Apply immediately (Aplicar imediatamente) na página de confirmação.

AWS CLI

Para alternar o segredo de uma senha do usuário principal usando a AWS CLI, utilize o comando modify-db-cluster e especifique a opção `--rotate-master-user-password`. Você deve especificar a opção `--apply-immediately` ao alternar a senha principal.

Este exemplo alterna o segredo de uma senha do usuário principal.

Example

Para Linux, macOS ou Unix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

API do RDS

Você pode alternar o segredo de uma senha do usuário principal usando a operação ModifyDBCluster e definindo o parâmetro `RotateMasterUserPassword` como `true`. Você deve definir o parâmetro `ApplyImmediately` como `true` ao alternar a senha principal.

Visualizar os detalhes sobre um segredo para um cluster de banco de dados

É possível recuperar seus segredos usando o console (https://console.aws.amazon.com/secretsmanager/) ou a AWS CLI (comando get-secret-value do Secrets Manager).

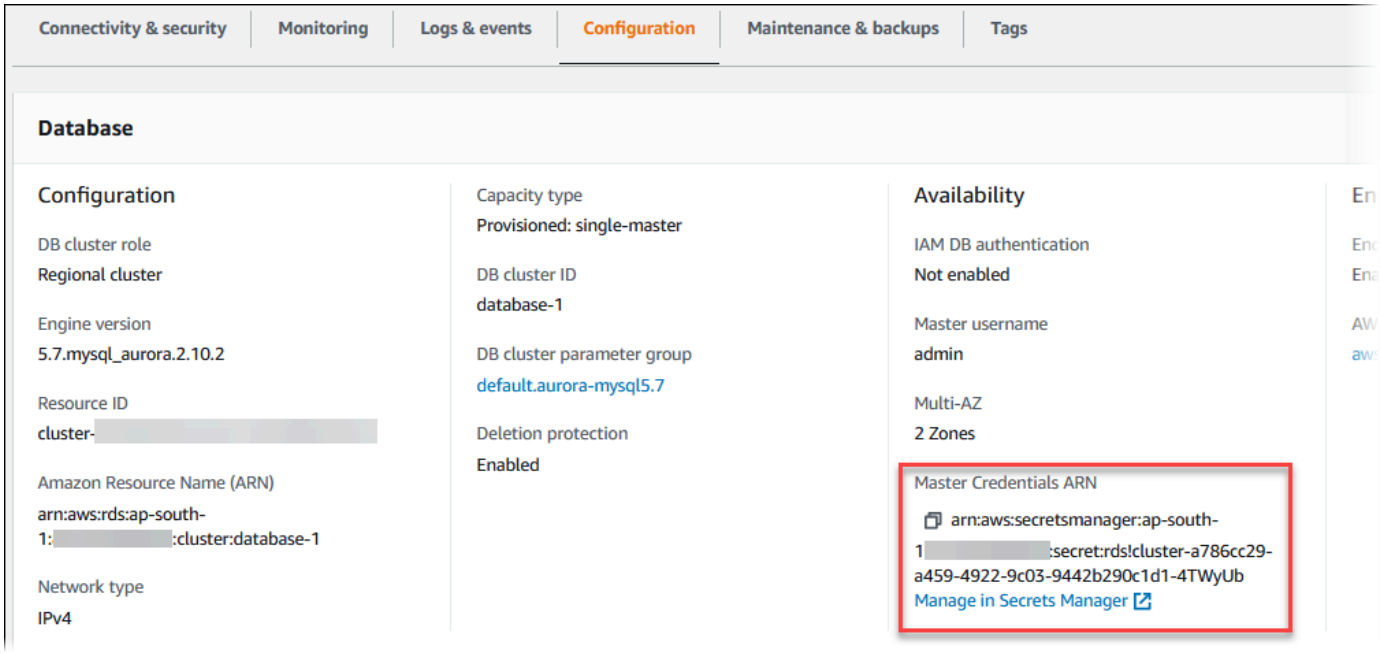
Você pode encontrar o nome do recurso da Amazon (ARN) de um segredo gerenciado pelo Aurora no Secrets Manager com o console do RDS, a AWS CLI o ou a API do RDS.

Console

Como visualizar os detalhes sobre um segredo gerenciado pelo Aurora no Secrets Manager

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Selecione o nome do cluster de banco de dados para mostrar os detalhes.
4. Escolha a guia Configuração.

   Em Master Credentials ARN (ARN das credenciais principais), você pode ver o ARN do segredo.



Você pode seguir o link Manage in Secrets Manager (Gerenciar no Secrets Manager) para visualizar e gerenciar o segredo no console do Secrets Manager.

AWS CLI

Você pode usar o comando describe-db-clusters da AWS CLI do RDS para encontrar as seguintes informações sobre um segredo gerenciado pelo Aurora no Secrets Manager:

- `SecretArn`: o ARN do segredo
- `SecretStatus`: o status do segredo

  Os valores de status possíveis são os seguintes:

  - `creating`: o segredo está sendo criado.
  - `active`: o segredo está disponível para uso e alternância normais.
  - `rotating`: o segredo está sendo alternado.
  - `impaired`: o segredo pode ser usado para acessar as credenciais do banco de dados, mas não pode ser alternado. Um segredo pode ter esse status se, por exemplo, as permissões forem alteradas para que o RDS não possa mais acessar o segredo nem a chave do KMS do segredo.

    Quando um segredo tem esse status, você pode corrigir a condição que o causou. Se você corrigir a condição que causou o status, ele permanecerá `impaired` até a próxima alternância. Como alternativa, você pode modificar o cluster de banco de dados para desativar o

gerenciamento automático das credenciais do banco de dados e, depois, modificar o cluster de banco de dados novamente para ativar o gerenciamento automático das credenciais do banco de dados. Para modificar o cluster de banco de dados, use a opção `--manage-master-user-password` no comando modify-db-cluster.

- `KmsKeyId`: o ARN da chave do KMS usada para criptografar o segredo

Especifique a opção `--db-cluster-identifier` para mostrar a saída de um cluster de banco de dados específico. Este exemplo mostra a saída de um segredo usado por um cluster de banco de dados.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

O exemplo a seguir mostra a saída de um segredo:

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

Quando você tiver o ARN do segredo, poderá visualizar detalhes sobre o segredo usando o comando da CLI get-secret-value do Secrets Manager.

Este exemplo mostra os detalhes do segredo na saída de exemplo anterior.

Example

Para Linux, macOS ou Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Para Windows:

```
aws secretsmanager get-secret-value ^
```

```
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

API do RDS

Você pode visualizar o ARN, o status e a chave do KMS de um segredo gerenciado pelo Aurora no Secrets Manager usando a operação DescribeDBClusters do RDS e definindo o parâmetro `DBClusterIdentifier` como um identificador de cluster de banco de dados. Detalhes sobre o segredo estão incluídos na saída.

Quando você tiver o ARN do segredo, poderá visualizar detalhes sobre o segredo usando o comando da CLI GetSecretValue do Secrets Manager.

Proteção de dados no Amazon RDS

O modelo de responsabilidade compartilhada da AWS aplica-se à proteção de dados no Amazon Relational Database Service. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Data Privacy FAQ. Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog AWS Shared Responsibility Model and RGPD no Blog de segurança da AWS.

Para fins de proteção de dados, recomendamos que você proteja as credenciais da Conta da AWS e configure as contas de usuário individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure os logs de API e atividade do usuário com AWS CloudTrail. Para obter informações sobre como usar as trilhas do CloudTrail para capturar atividades da AWS, consulte Working with CloudTrail trails no Guia do usuário do AWS CloudTrail.
- Use as soluções de criptografia AWS, juntamente com todos os controles de segurança padrão em Serviços da AWS.

- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar a AWS por meio de uma interface de linha de comandos ou de uma API, use um endpoint do FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sigilosas, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso também vale para o uso do Amazon RDS ou de outros Serviços da AWS com o console, a API, a AWS CLI ou os AWS SDKs. Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Tópicos

- Proteção de dados usando criptografia
- Privacidade do tráfego entre redes

Proteção de dados usando criptografia

O Amazon Aurora criptografa os recursos do banco de dados na camada de armazenamento. Também é possível criptografar conexões com clusters de banco de dados.

Tópicos

- Criptografar recursos do Amazon Aurora
- AWS KMS keyGerenciamento de
- Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados
- Alternar o certificado SSL/TLS

Criptografar recursos do Amazon Aurora

O Amazon Aurora protege os dados em repouso e em trânsito, seja quando eles são movidos entre clientes on-premises e o Amazon Aurora ou entre o Amazon Aurora e outros recursos da AWS.

O Amazon Aurora criptografa todos os dados do usuário em seus clusters de banco de dados do Amazon Aurora, inclusive logs, backups automatizados e snapshots.

Após a criptografia dos dados, o Amazon Aurora lida com a autenticação do acesso e a decodificação dos dados de forma transparente com um impacto mínimo sobre o desempenho. Você não precisa modificar suas aplicações cliente de banco de dados para usar a criptografia.

> **Note**
>
> Para clusters, os dados em trânsito entre a origem e as réplicas de leitura são criptografados, até mesmo quando a replicação ocorre entre regiões da AWS.

Tópicos

- Visão geral sobre criptografia em recursos do Amazon Aurora
- Criptografar um cluster de banco de dados do Amazon Aurora
- Verificar se a criptografia está habilitada para um cluster de banco de dados
- Disponibilidade da criptografia do Amazon Aurora
- Criptografia em trânsito
- Limitações dos clusters de banco de dados criptografados do Amazon Aurora

Visão geral sobre criptografia em recursos do Amazon Aurora

Os clusters de banco de dados criptografados do Amazon Aurora fornecem uma camada adicional de proteção de dados, protegendo seus dados contra o acesso não autorizado ao armazenamento subjacente. Todos os novos clusters de banco de dados criados em ou após 18 de fevereiro de 2026 e no Amazon Aurora são criptografados em repouso usando a criptografia AES-256 padrão do setor. Essa criptografia é realizada automaticamente em segundo plano, protegendo os dados sem exigir nenhuma ação de sua parte. Isso também ajuda a reduzir a carga e complexidade operacionais necessárias para proteger dados sensíveis. Com a criptografia de dados em repouso, é possível proteger aplicações que exigem conformidade e alto nível de segurança contra ameaças acidentais e mal-intencionadas, bem como atender a requisitos regulamentares.

O Amazon Aurora usa uma chave do AWS Key Management Service para criptografar esses recursos. O AWS KMS combina hardware e software seguros e altamente disponíveis para fornecer um sistema de gerenciamento de chaves escalado para a nuvem. Ao criar um cluster de banco de

dados, o Amazon Aurora usa criptografia do lado do servidor (SSE) com uma chave de propriedade da AWS por padrão. No entanto, é possível escolher entre três tipos de criptografia com base em suas necessidades de segurança e conformidade:

- Chave de propriedade da AWS (SSE-RDS): uma chave de criptografia totalmente controlada pela AWS que é usada automaticamente pelo Aurora para criptografia padrão e não pode ser visualizada nem gerenciada pelo usuário.
- Chave gerenciada pela AWS (AMK): essa chave, criada e gerenciada pela AWS, é visível em sua conta, mas não pode ser personalizada. Não há taxa mensal, mas cobranças de API do AWS KMS serão aplicadas.
- Chave gerenciada pelo cliente (CMK): a chave é armazenada na sua conta e é você que a cria, detém e gerencia. Você tem controle total sobre a chave do KMS (cobranças do AWS KMS são aplicáveis).

As chaves gerenciadas pela AWS são uma opção de criptografia legada que permanece disponível para oferecer compatibilidade com versões anteriores. Por padrão, o Amazon Aurora usa chaves de propriedade da AWS para criptografar os dados, o que oferece uma sólida proteção à segurança sem cobranças adicionais nem custos indiretos de gerenciamento. Para a maioria dos casos de uso, recomendamos usar a chave de propriedade da AWS padrão para simplificar e economizar ou uma chave gerenciada pelo cliente (CMK) se você precisar de controle total sobre suas chaves de criptografia. Para saber mais sobre tipos de chave, consulte as informações sobre chaves gerenciadas pelo cliente e chaves gerenciadas pela AWS.

> **Note**
>
> Importante: para instâncias ou clusters de banco de dados de origem ou clusters criados antes de 18 de fevereiro de 2026 e em , nos quais você não optou por criptografia, snapshots, clones e réplicas do Amazon Aurora (instância de leitura) criadas com base nessas fontes, permanecerão sem criptografia. No entanto, as operações de restauração e a replicação lógica fora do cluster do Amazon Aurora produzirão instâncias criptografadas.

Em um cluster de banco de dados criptografado do Amazon Aurora, todas as instâncias de banco de dados, logs, backups e snapshots são criptografados. Consulte mais informações sobre a disponibilidade e as limitações da criptografia em Disponibilidade da criptografia do Amazon Aurora e Limitações dos clusters de banco de dados criptografados do Amazon Aurora.

Ao criar um cluster de banco de dados criptografado, é possível escolher uma chave gerenciada pelo cliente ou a Chave gerenciada pela AWS para que o Amazon Aurora criptografe o cluster de banco de dados. Se você não especificar o identificador de chave para uma chave gerenciada pelo cliente, o Amazon Aurora usará a Chave gerenciada pela AWS para seu novo cluster de banco de dados. O Amazon Aurora cria uma Chave gerenciada pela AWS para o Amazon Aurora para a sua conta da AWS. A sua conta da AWS tem uma Chave gerenciada pela AWS diferente para o Amazon Aurora para cada região da AWS.

Para gerenciar as chaves gerenciadas pelo cliente usadas para criptografia e descriptografia de recursos do Amazon Aurora, use o AWS Key Management Service (AWS KMS).

Com o AWS KMS, é possível criar chaves gerenciadas pelo cliente e definir as políticas para controlar o uso dessas chaves. O AWS KMS é compatível com o CloudTrail, o que permite auditar o uso da chave do KMS para verificar se as chaves gerenciadas pelo cliente estão sendo usadas adequadamente. Você pode usar as chaves gerenciadas com o Amazon Aurora e serviços compatíveis da AWS, como o Amazon S3, Amazon EBS e Amazon Redshift. Para obter uma lista de serviços integrados ao AWS KMS, consulte Integração de serviços da AWS. Algumas considerações sobre o uso de chaves do KMS:

- Assim que uma instância de banco de dados criptografada é criada, não é possível alterar a chave do KMS usada por essa instância de banco de dados. Determine os requisitos da chave do KMS antes de criar a instância de banco de dados criptografada. Se você precisar alterar a chave de criptografia do cluster de banco de dados, siga estas etapas:
  - Crie um snapshot manual do cluster.
  - Restaure o snapshot e habilite a criptografia com a chave do KMS desejada durante a operação de restauração.
- Se você restaurar um snapshot não criptografado e optar por não criptografar, o cluster de banco de dados criado será criptografado usando a criptografia padrão em repouso (chave de propriedade da AWS).
- Não é possível compartilhar um snapshot que foi criptografado usando a Chave gerenciada pela AWS da conta da AWS que o compartilhou.
- Cada instância de banco de dados no cluster de banco de dados tem em comum o mesmo armazenamento criptografado com a mesma chave do KMS.

Important

O Amazon Aurora pode perder o acesso à chave do KMS para um cluster de banco de dados quando você desabilita a chave do KMS. Nesses casos, o cluster de banco de dados criptografada entra no estado `inaccessible-encryption-credentials-recoverable`. O cluster de banco de dados permanece nesse estado por sete dias, durante os quais a instância é interrompida. As chamadas de API feitas ao cluster de banco de dados durante esse período podem não ser bem-sucedidas. Para recuperar o cluster de banco de dados, habilite a chave do KMS e reinicie esse cluster. É possível habilitar a chave do KMS no Console de gerenciamento da AWS, na AWS CLI ou na API do RDS. Reinicie o cluster de banco de dados utilizando o comando start-db-cluster da AWS CLI ou o Console de gerenciamento da AWS.

O estado `inaccessible-encryption-credentials-recoverable` só se aplica a clusters de banco de dados que podem ser interrompidos. Esse estado recuperável não se aplica a instâncias que não podem ser interrompidas, como cluster com réplicas de leitura entre regiões. Para obter mais informações, consulte the section called “Limitações”.

Se o cluster de banco de dados não for recuperado em até sete dias, ele entrará no estado `inaccessible-encryption-credentials` do terminal. Nesse estado, o cluster de banco de dados não pode mais ser usado e só é possível restaurá-lo por um backup. É recomendável sempre habilitar backups para evitar a perda de dados em seus bancos de dados.

Durante a criação de um cluster de banco de dados, o Aurora verifica se a entidade principal da chamada tem acesso à chave do KMS e gera uma concessão da chave do KMS, que é usada durante toda a vida útil do cluster de banco de dados. A revogação do acesso da entidade principal da chamada à chave do KMS não afeta um banco de dados em execução. Ao usar chaves do KMS em cenários entre contas, como copiar um snapshot para outra conta, a chave do KMS precisa ser compartilhada com a outra conta. Se você criar um cluster de banco de dados com base no snapshot sem especificar uma chave do KMS diferente, o novo cluster usará a chave do KMS da conta de origem. A revogação do acesso à chave após a criação do cluster de banco de dados não afeta o cluster. No entanto, a desabilitação da chave afeta todos os clusters de banco de dados criptografados com essa chave. Para evitar isso, especifique uma chave diferente durante a operação de cópia do snapshot.

Para ter mais informações sobre chaves do KMS, consulte AWS KMS keys no Guia do desenvolvedor do AWS Key Management Service e AWS KMS keyGerenciamento de.

Criptografar um cluster de banco de dados do Amazon Aurora

Todos os novos clusters de banco de dados criados em ou após 18 de fevereiro de 2026 são criptografados por padrão com uma chave de propriedade da AWS.

Para criptografar um novo cluster de banco de dados usando uma Chave gerenciada pela AWS ou uma chave gerenciada pelo cliente, escolha a opção no console. Para obter informações sobre como criar com um cluster de banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Se você usar o comando create-db-cluster da AWS CLI para criar um cluster de banco de dados criptografado, defina o parâmetro `--storage-encrypted`. Se você usar a operação da API CreateDBCluster, defina o parâmetro `StorageEncrypted` como true.

Depois de criar um cluster de banco de dados criptografado, não será possível alterar a chave do KMS usada por esse cluster de banco de dados. Portanto, certifique-se de determinar os requisitos da chave do KMS antes de criar o seu cluster de banco de dados criptografado.

Se você usar o comando AWS CLI da `create-db-cluster` para criar um cluster de banco de dados criptografado com uma chave gerenciada pelo cliente, defina o parâmetro `--kms-key-id` para qualquer identificador da chave do KMS. Se você usar a operação `CreateDBInstance` da API do Amazon RDS, defina o parâmetro `KmsKeyId` para qualquer identificador de chave do KMS. Para usar uma chave gerenciada pelo cliente em outra conta da AWS, especifique o ARN da chave ou o ARN do alias.

Verificar se a criptografia está habilitada para um cluster de banco de dados

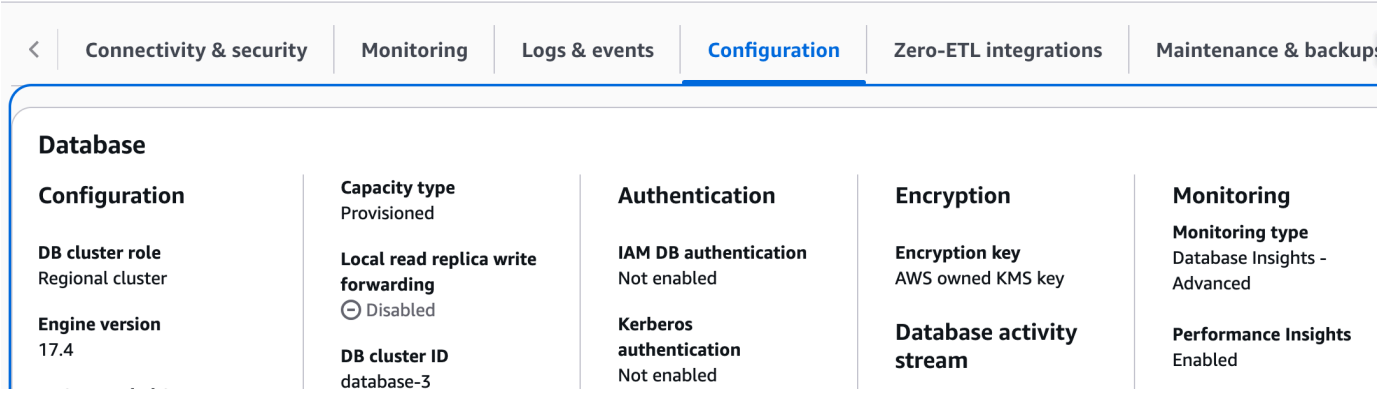
É possível utilizar o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para verificar se a criptografia em repouso está habilitada para um cluster de banco de dados.

Console

Para verificar se a criptografia em repouso está habilitada para um cluster de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha no nome do cluster de banco de dados que você deseja verificar para mostrar os detalhes.

4. Escolha a guia Configuration (Configuração) e verifique o valor Encryption (Criptografia).



AWS CLI

Para verificar se a criptografia em repouso está habilitada para um cluster de banco de dados usando a AWS CLI, chame o comando describe-db-clusters comando com a seguinte opção:

- `--db-cluster-identifier`: o nome do cluster de banco de dados.

O exemplo a seguir utiliza uma consulta para retornar `TRUE` ou `FALSE` referente à criptografia em repouso para o cluster de banco de dados `mydb`.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

API do RDS

Para verificar se a criptografia em repouso está habilitada para um cluster de banco de dados usando a API do Amazon RDS, chame a operação DescribeDBClusters com este parâmetro:

- `DBClusterIdentifier`: o nome do cluster de banco de dados.

Disponibilidade da criptografia do Amazon Aurora

No momento, a criptografia do Amazon Aurora está disponível para todos os mecanismos de banco de dados e tipos de armazenamento.

> **Note**
>
> A criptografia do Amazon Aurora não está disponível para a classe de instância de banco de dados db.t2.micro.

Criptografia em trânsito

Criptografia na camada física

Todos os dados que fluem pelas Regiões da AWS por meio da rede global da AWS são automaticamente criptografados na camada física antes de saírem das instalações seguras da AWS. Todo o tráfego entre AZs é criptografado. Camadas adicionais de criptografia, inclusive as listadas nesta seção, podem oferecer mais proteções.

Criptografia fornecida pelo emparelhamento da Amazon VPC e do Transit Gateway entre regiões

Todo o tráfego entre regiões que usa o emparelhamento da Amazon VPC e do Transit Gateway é automaticamente criptografado em massa ao sair de uma região. Uma camada adicional de criptografia é fornecida automaticamente à camada física para todo tráfego antes que ele saia das instalações seguras da AWS.

Criptografia entre instâncias

A AWS fornece conectividade privada e segura entre instâncias de banco de dados de todos os tipos. Além disso, alguns tipos de instância usam os recursos de descarregamento do hardware subjacente Nitro System para criptografar automaticamente o tráfego em trânsito entre instâncias. Essa criptografia usa algoritmos de criptografia autenticada com dados associados (AEAD) com criptografia de 256 bits. Não há impacto na performance da rede. Para oferecer suporte a essa criptografia adicional de tráfego em trânsito entre instâncias, os seguintes requisitos devem ser atendidos:

- As instâncias usam os seguintes tipos de instância:
  - Uso geral: M6i, M6id, M6in, M6idn, M7g
  - Otimizadas para memória: R6i, R6id, R6in, R6idn, R7g, X2idn, X2iedn, X2iezn
- As instâncias estão na mesma Região da AWS.
- As instâncias estão na mesma VPC ou VPCs emparelhadas, e o tráfego não passa por um dispositivo ou serviço de rede virtual, como um balanceador de carga ou um gateway de trânsito.

Limitações dos clusters de banco de dados criptografados do Amazon Aurora

As seguintes limitações existem para clusters criptografados de banco de dados do Amazon Aurora:

- Não é possível desativar a criptografia em um cluster de banco de dados criptografado.
- Se você tiver um cluster não criptografado, todos os snapshots criados com base nesse cluster também não serão criptografados. Para criar um snapshot criptografado com base em um cluster não criptografado, é necessário copiar o snapshot e especificar uma chave gerenciada pelo cliente durante a operação de cópia. Não é possível criar um snapshot criptografado com base em um snapshot não criptografado sem especificar uma chave gerenciada pelo cliente.
- Também não é possível criar um snapshot criptografado de uma de banco de dados não criptografada.
- Um snapshot de uma instância de banco de dados criptografo deve ser criptografado usando a mesma chave do KMS que a instância de banco de dados.
- Não é possível converter um cluster de banco de dados não criptografado em um criptografado. Entretanto, é possível restaurar um snapshot não criptografado em um cluster de banco de dados criptografado do Aurora. Para fazê-lo, especifique uma chave do KMS ao fazer uma restauração no snapshot não criptografado.
- Se você tiver um cluster não criptografado, qualquer réplica do Amazon Aurora (instância de leitura) criada com base nesse cluster também não será criptografada. Para criar um cluster criptografado em um cluster não criptografado, é necessário restaurar o cluster de banco de dados. O cluster restaurado será criptografado por padrão após a operação de restauração.
- Para copiar um snapshot criptografado de uma região da AWS para outra, é necessário especificar a chave do KMS na região da AWS de destino. Isso ocorre porque as chaves do KMS são específicas da região da AWS em que são criadas.

  O snapshot de origem permanece criptografado ao longo do processo de cópia. O Amazon Aurora usa criptografia de envelope para proteger os dados durante o processo de cópia. Para ter mais informações sobre a criptografia de envelope, consulte Criptografia de envelope no Guia do desenvolvedor do AWS Key Management Service.
- Não é possível descriptografar um cluster de banco de dados criptografado. No entanto, é possível exportar dados de um cluster de banco de dados criptografado e importar os dados para um cluster de banco de dados não criptografado.

AWS KMS keyGerenciamento de

O Amazon Aurora integra-se automaticamente ao AWS Key Management Service (AWS KMS) para gerenciar chaves. O Amazon Aurora usa criptografia envelopada. Para ter mais informações sobre a criptografia de envelope, consulte Criptografia de envelope no Guia do desenvolvedor do AWS Key Management Service.

Você pode usar dois tipos de chave do AWS KMS para criptografar clusters de banco de dados.

- Se você quiser o controle total de uma chave do KMS, precisará criar uma chave gerenciada pelo cliente. Para obter mais informações sobre chaves gerenciadas pelo cliente, consulte Chaves gerenciadas pelo cliente no Guia do desenvolvedor do AWS Key Management Service.
- Chaves gerenciadas pela AWS são chaves do KMS em sua conta que são criadas, gerenciadas e usadas em seu nome por um produto da AWS integrado ao AWS KMS. Por padrão, a Chave gerenciada pela AWS do RDS (`aws/rds`) é usada para criptografia. Você não pode gerenciar, nem rotacionar, nem excluir a Chave gerenciada pela AWS do RDS. Para obter mais informações sobre as Chaves gerenciadas pela AWS, consulte Chaves gerenciadas pela AWS no Guia do desenvolvedor do AWS Key Management Service.

Para gerenciar as chaves do KMS usadas para clusters de banco de dados criptografados do Amazon Aurora, use o AWS Key Management Service (AWS KMS) no console do AWS KMS, a AWS CLI ou a API do AWS KMS. Para visualizar logs de auditoria de cada ação realizada com uma chave gerenciada pela AWS ou pelo cliente, use o AWS CloudTrail. Para obter mais informações sobre a alternância de chaves, consulte Alternância de chaves do AWS KMS.

Como autorizar o uso de uma chave gerenciada pelo cliente

Quando o Aurora utiliza uma chave gerenciada pelo cliente em operações criptográficas, ele atua em nome do usuário que está criando ou alterando o recurso do Aurora.

Para criar um recurso do Aurora usando uma chave gerenciada pelo cliente, um usuário deve ter permissões para acionar as seguintes operações na chave gerenciada pelo cliente:

- `kms:CreateGrant`
- `kms:DescribeKey`

Você pode especificar essas permissões em uma política de chaves ou em uma política do IAM, se a política de chaves permitir.

> **Important**
>
> Ao usar instruções de negação explícitas para todos os recursos (*) nas políticas de chave do AWS KMS com serviços gerenciados como o Amazon RDS, você deve especificar uma condição para dar permissão à conta proprietária do recurso. As operações podem falhar sem essa condição, mesmo que a regra de negação inclua exceções para seu usuário do IAM.

> **Tip**
>
> Para seguir o princípio de menor privilégio, não permita acesso total a `kms:CreateGrant`. Em vez disso, use a chave de condição kms:ViaService para permitir que o usuário crie concessões na chave do KMS somente quando a concessão for criada em nome do usuário por um serviço da AWS.

É possível tornar a política do IAM mais rígida de várias maneiras. Por exemplo, se você quiser permitir que a chave gerenciada pelo cliente seja usada somente para solicitações provenientes do Aurora, poderá utilizar a chave de condição kms:ViaService com o valor `rds.<region>.amazonaws.com`. Também é possível usar as chaves ou valores em Contexto de criptografia do Amazon RDS como uma condição para usar a chave gerenciada pelo cliente para criptografia.

Para obter mais informações, consulte Permitir que usuários de outras contas usem uma chave do KMS no Guia do desenvolvedor do AWS Key Management Service e em Políticas de chave no AWS KMS.

Contexto de criptografia do Amazon RDS

Quando o Aurora usa sua chave do KMS ou quando o Amazon EBS usa a chave do KMS em nome do Aurora, o serviço especifica um contexto de criptografia. O contexto de criptografia é dados autenticados adicionados (AAD) que o AWS KMS usa para garantir a integridade dos dados. Quando um contexto de criptografia é especificado para uma operação de criptografia, o serviço deve especificar esse mesmo contexto para a operação de descriptografia. Caso contrário, ocorrerá uma falha na descriptografia. O contexto de criptografia é também gravado nos logs do AWS CloudTrail para ajudar você a compreender por que uma determinada chave do KMS foi usada. Os logs do CloudTrail podem conter muitas entradas que descrevem o uso de uma chave do KMS, mas o

contexto de criptografia em cada entrada de log pode ajudar a determinar o motivo desse uso específico.

No mínimo, o Aurora sempre usa o ID do cluster de banco de dados para o contexto de criptografia, como no seguinte exemplo em formato JSON:

```
{ "aws:rds:dbc-id": "cluster-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

Esse contexto de criptografia pode ajudar a identificar para qual cluster de banco de dados a chave do KMS foi usada.

Quando a chave do KMS é usada em determinado cluster de banco de dados e em um volume do Amazon EBS específico, o ID do cluster de banco de dados e o ID do volume do Amazon EBS são usados no contexto de criptografia, como no seguinte exemplo em formato JSON:

```
{
  "aws:rds:dbc-id": "cluster-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados

É possível usar o Security Socket Layer (SSL) ou o Transport Layer Security (TLS) na aplicação para criptografar uma conexão com um cluster de banco de dados que executa o Aurora MySQL ou o Aurora PostgreSQL.

As conexões SSL/TLS oferecem uma camada de segurança criptografando dados que se movem entre o cliente e o cluster de banco de dados. Opcionalmente, sua conexão SSL/TLS pode realizar a verificação da identidade do servidor validando o certificado do servidor instalado no banco de dados. Para exigir a verificação da identidade do servidor, siga este processo geral:

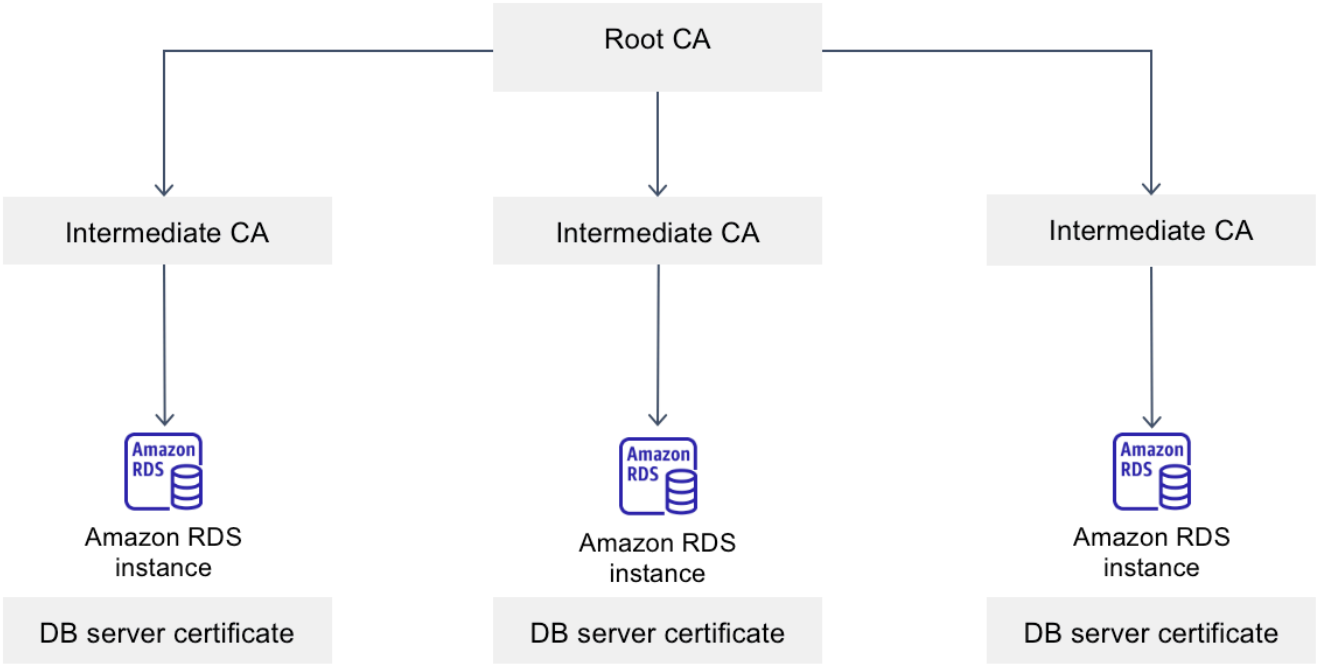
1. Escolha a Autoridade de certificação (CA) que assina ocertificado de servidor de banco de dados para seu banco de dados. Para ter mais informações sobre autoridades de certificação, consulte Autoridades certificadoras.
2. Baixe um pacote de certificados para usar quando você estiver se conectando com o banco de dados. Para baixar um pacote de certificados, consulte Pacote de certificados por Região da AWS.

> **Note**
>
> Todos os certificados somente estão disponíveis para download usando conexões SSL/TLS.

3. Conecte-se ao banco de dados usando o processo do mecanismo de banco de dados para implementar conexões SSL/TLS. Cada mecanismo de banco de dados tem seu próprio processo de implementação do SSL/TLS. Para saber como implementar o SSL/TLS para o banco de dados, siga o link correspondente ao seu mecanismo de banco de dados:
   - Segurança com o Amazon Aurora MySQL
   - Segurança com o Amazon Aurora PostgreSQL

Autoridades certificadoras

A autoridade de certificação (CA) é o certificado que identifica a CA raiz no início da cadeia de certificados. A CA assina o certificado do servidor de banco de dados, que é instalado em cada instância de banco de dados. O certificado do servidor de banco de dados identifica a instância de banco de dados como um servidor confiável.



O Amazon RDS fornece as CAs a seguir para assinar o certificado do servidor de banco de dados para um banco de dados.

Autoridade certificadora (CA)	Descrição	Nome comum (CN)
rds-ca-rsa2048-g1	Utiliza uma autoridade de certificação com algoritmo de chave privada RSA 2048 e algoritmo de assinatura SHA256 na maioria das Regiões da AWS. Nas AWS GovCloud (US) Regions, essa CA utiliza uma autoridade de certifica ção com algoritmo de chave privada RSA 2048 e algoritmo de assinatura SHA384. Essa CA é compatível com a alternância automática de certificados do servidor.	Amazon RDS *region-id entifier* Root CA RSA2048 G1
rds-ca-rsa4096-g1	Utiliza uma autoridade de certificação com algoritmo de chave privada RSA 4096 e algoritmo de assinatura SHA384. Essa CA é compatível com a alternância automática de certificados do servidor.	Amazon RDS *region-id entifier* Root CA RSA4096 G1
rds-ca-ecc384-g1	Utiliza uma autoridade de certificação com algoritmo de chave privada ECC 384 e algoritmo de assinatura SHA384. Essa CA é compatível com a alternância automática de certificados do servidor.	Amazon RDS *region-id entifier* Root CA ECC384 G1

> **Note**
>
> Se você estiver usando a AWS CLI, poderá ver as validades das autoridades de certificação listadas acima usando describe-certificates.

Esses certificados CA estão incluídos no pacote de certificados regionais e globais. Quando você usa a CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 ou rds-ca-ecc384-g1 com um banco de dados, o RDS gerencia o certificado do servidor no banco de dados. O RDS alterna automaticamente o certificado do servidor de banco de dados antes que ele expire.

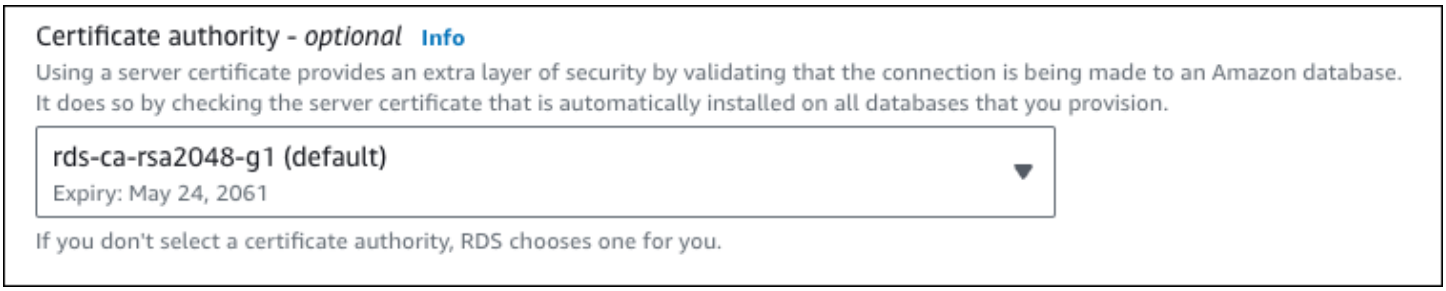
Configurar a CA do banco de dados

Você pode definir a CA para um banco de dados ao realizar as seguintes tarefas:

- Crie um cluster de banco de dados do Aurora: você pode definir a CA para uma instância de banco de dados em um cluster do Aurora ao criar a primeira instância de banco de dados no cluster de banco de dados usando a AWS CLI ou a API do RDS. Atualmente, você não pode definir a CA para as instâncias de banco de dados em um cluster de banco de dados ao criar o cluster de banco de dados com o uso do console do RDS. Para instruções, consulte Criar um cluster de bancos de dados do Amazon Aurora .
- Modificar uma instância de banco de dados: você pode definir a CA para uma instância de banco de dados em um cluster de banco de dados modificando-a. Para instruções, consulte Modificar uma instância de banco de dados em um cluster de banco de dados .

> **Note**
>
> A CA padrão é definida como rds-ca-rsa2048-g1. Você pode substituir a CA padrão para sua Conta da AWS usando o comando modify-certificates.

As CAs disponíveis dependem do mecanismo de banco de dados e da versão do mecanismo de banco de dados. Ao usar o Console de gerenciamento da AWS, você pode selecionar a CA usando a configuração Certificate authority (Autoridade de certificação), conforme mostrado na imagem a seguir.



O console mostra apenas as CAs que estão disponíveis para o mecanismo de banco de dados e a versão do mecanismo de banco de dados. Se estiver usando a AWS CLI, você poderá definir a CA para uma instância de banco de dados usando o comando create-db-instance ou modify-db-instance.

Se estiver usando a AWS CLI, você poderá ver as CAs disponíveis para sua conta usando o comando describe-certificates. Esse comando também mostra a data de expiração de cada CA em `ValidTill` na saída. Você pode encontrar as CAs que estão disponíveis para uma versão específica do mecanismo de banco de dados e do mecanismo de banco de dados usando o comando describe-db-engine-versions.

O exemplo a seguir mostra as CAs disponíveis para a versão padrão do mecanismo de banco de dados do RDS para PostgreSQL.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

A saída é semelhante à seguinte. As CAs disponíveis estão listadas em `SupportedCACertificateIdentifiers`. A saída também mostra se a versão do mecanismo de banco de dados é compatível com a alternância do certificado sem reiniciar em `SupportsCertificateRotationWithoutRestart`.

```
{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
```

```
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

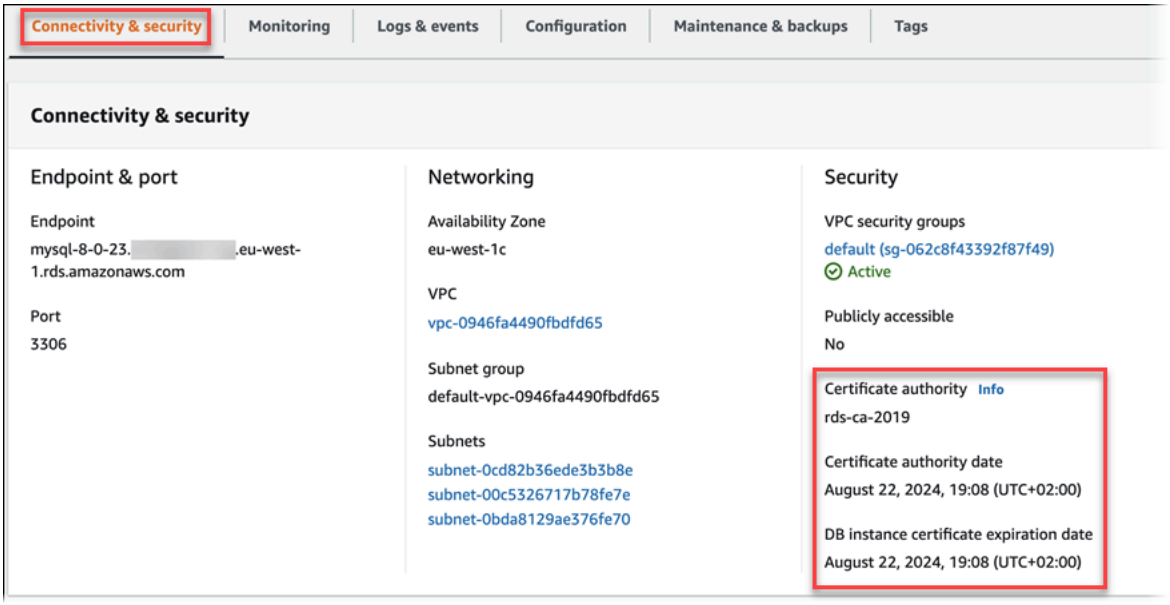
Validades do certificado do servidor de banco de dados

A validade do certificado do servidor de banco de dados depende do mecanismo de banco de dados e da versão do respectivo mecanismo. Se a versão do mecanismo comportar a alternância de certificado sem reinicialização, a validade do certificado do mecanismo será de um ano. Caso contrário, será de três anos.

Para obter mais informações sobre alternância de certificados de servidor de banco de dados, consulte Alternância automática de certificados do servidor.

Visualizar a CA da instância de banco de dados

É possível visualizar os detalhes sobre a CA para um banco de dados na guia Conectividade e segurança no console, como na imagem a seguir.



Se estiver usando a AWS CLI, você poderá visualizar os detalhes da CA de uma instância de banco de dados usando o comando describe-db-instances.

Baixar pacotes de certificados para o Aurora

Quando você se conecta ao banco de dados com SSL ou TLS, a instância do banco de dados exige um certificado de confiança do Amazon RDS. Selecione o link apropriado na tabela a seguir para baixar o pacote que corresponde à Região da AWS onde você hospeda o banco de dados.

Pacote de certificados por Região da AWS

Os pacotes de certificados para todas as Regiões da AWS e as regiões GovCloud (EUA) contêm os seguintes certificados de CA raiz:

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

Os certificados `rds-ca-rsa4096-g1` e `rds-ca-ecc384-g1` não estão disponíveis nas seguintes regiões:

- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Melbourne)
- Oeste do Canadá (Calgary)
- Europa (Zurique)
- Europa (Espanha)
- Israel (Tel Aviv)

O armazenamento confiável da aplicação só precisa registrar o certificado CA raiz. Não registre os certificados CA intermediários em seu armazenamento confiável, pois isso poderá causar problemas de conexão quando o RDS alternar automaticamente o certificado do servidor de banco de dados.

> **Note**
>
> O Amazon RDS Proxy e Aurora Serverless v1 usam certificados do AWS Certificate Manager (ACM). Se você estiver usando o RDS Proxy, não será necessário baixar os certificados do Amazon RDS nem atualizar as aplicações que usam conexões do RDS Proxy. Para ter mais informações, consulte Usar TLS/SSL com o RDS Proxy .

Se você estiver usando o Aurora Serverless v1, não será necessário baixar certificados do Amazon RDS. Para ter mais informações, consulte Usar TLS/SSL com o Aurora Serverless v1 .

Para baixar um pacote de certificados para uma Região da AWS, selecione o link da Região da AWS que hospeda o banco de dados na tabela a seguir.

AWS Região da	Pacote de certificados (PEM)	Pacote de certificados (PKCS7)
Qualquer Região da AWS comercial.	global-bundle.pem	global-bundle.p7b
Leste dos EUA (Norte da Virgínia)	us-east-1-bundle.pem	us-east-1-bundle.p7b
Leste dos EUA (Ohio)	us-east-2-bundle.pem	us-east-2-bundle.p7b
Oeste dos EUA (N. da Califórnia)	us-west-1-bundle.pem	us-west-1-bundle.p7b
Oeste dos EUA (Oregon)	us-west-2-bundle.pem	us-west-2-bundle.p7b
África (Cidade do Cabo)	af-south-1-bundle.pem	af-south-1-bundle.p7b
Ásia-Pacífico (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
Ásia-Pacífico (Hyderabad)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
Ásia-Pacífico (Jacarta)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
Ásia-Pacífico (Malásia)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
Ásia-Pacífico (Melbourne)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
Ásia-Pacífico (Mumbai)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b

AWS Região da	Pacote de certificados (PEM)	Pacote de certificados (PKCS7)
Ásia-Pacífico (Tailândia)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
Ásia-Pacífico (Tóquio)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Ásia-Pacífico (Seul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
Ásia-Pacífico (Singapura)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
Ásia-Pacífico (Sydney)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b
Canadá (Central)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
Oeste do Canadá (Calgary)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
Europa (Frankfurt)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
Europa (Irlanda)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
Europa (Londres)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europa (Milão)	eu-south-1-bundle.pem	eu-south-1-bundle.p7b
Europa (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
Europa (Espanha)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
Europa (Estocolmo)	eu-north-1-bundle.pem	eu-north-1-bundle.p7b
Europa (Zurique)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
Israel (Tel Aviv)	il-central-1-bundle.pem	il-central-1-bundle.p7b
México (Centro)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
Oriente Médio (Bahrein)	me-south-1-bundle.pem	me-south-1-bundle.p7b
Oriente Médio (Emirados Árabes Unidos)	me-central-1-bundle.pem	me-central-1-bundle.p7b

AWS Região da	Pacote de certificados (PEM)	Pacote de certificados (PKCS7)
América do Sul (São Paulo)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
Quaisquer AWS GovCloud (US) Regions.	global-bundle.pem	global-bundle.p7b
AWS GovCloud (Leste dos EUA)	us-gov-east-1-bundle.pem	us-gov-east-1-bundle.p7b
AWS GovCloud (Oeste dos EUA)	us-gov-west-1-bundle.pem	us-gov-west-1-bundle.p7b

Visualizar o conteúdo do certificado CA

Para verificar o conteúdo do pacote de certificados da CA, use o seguinte comando:

```
keytool -printcert -v -file global-bundle.pem
```

Alternar o certificado SSL/TLS

Os certificados rds-ca-2019 da autoridade de certificação do Amazon RDS expiraram em agosto de 2024. Se você usa ou planeja usar Secure Sockets Layer (SSL) ou Transport Layer Security (TLS) com verificação de certificado para se conectar às instâncias de banco de dados do RDS ,considere usar um dos novos certificados CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 ou rds-ca-ecc384-g1. Se você não usa SSL/TLS com verificação de certificado no momento, é possível que ainda tenha algum certificado de CA expirado e precise atualizá-lo para um novo certificado de CA se planeja usar SSL/TLS com verificação de certificado para se conectar aos bancos de dados do RDS.

O Amazon RDS fornece novos certificados CA como uma prática recomendada de segurança da AWS. Para ter informações sobre os novos certificados e as regiões da AWS compatíveis, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

Para atualizar o certificado CA do seu banco de dados, use os seguintes métodos:

- Atualizar o certificado CA modificando a instância de banco de dados
- Atualizar seu certificado CA aplicando manutenção

Antes de atualizar as instâncias de banco de dados para usar o novo certificado CA, atualize os clientes ou as aplicações que se conectam aos bancos de dados do RDS.

Considerações sobre a troca de certificados

Pense nas seguintes situações antes de trocar seu certificado:

- O Amazon RDS Proxy e Aurora Serverless v1 usam certificados do AWS Certificate Manager (ACM). Se estiver usando o RDS Proxy, ao trocar o certificado SSL/TLS, não será necessário atualizar as aplicações que usam conexões do RDS Proxy. Para ter mais informações, consulte Usar TLS/SSL com o RDS Proxy .
- Se você estiver usando o Aurora Serverless v1, não será necessário baixar certificados do Amazon RDS. Para ter mais informações, consulte Usar TLS/SSL com o Aurora Serverless v1 .
- Se estiver usando uma aplicação Go versão 1.15 com uma instância de banco de dados criada ou atualizada para o certificado rds-ca-2019 antes de 28 de julho de 2020, você deverá atualizar o certificado novamente. Atualize o certificado para rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 ou rds-ca-ecc384-g1, dependendo do seu mecanismo .

  Use o comando `modify-db-instance` usando o novo identificador de certificado CA. Você pode encontrar as CAs que estão disponíveis para uma versão específica do mecanismo de banco de dados e do mecanismo de banco de dados usando o comando `describe-db-engine-versions`.

  Caso você tenha criado a instância de banco de dados ou atualizado o certificado dela após 28 de julho de 2020, nenhuma ação será necessária. Para obter mais informações, consulte Go GitHub issue #39568.

Atualizar o certificado CA modificando a instância de banco de dados

O exemplo a seguir atualiza o certificado de CA de rds-ca-2019 para rds-ca-rsa2048-g1. Você pode escolher um certificado diferente. Para ter mais informações, consulte Autoridades certificadoras..

Atualize seu repositório confiável de aplicações para reduzir o tempo de inatividade associado à atualização do certificado de CA. Consulte mais informações sobre reinicializações associadas à alternância de certificados de CA em Alternância automática de certificados do servidor.

Como atualizar o certificado CA modificando a instância de banco de dados

1. Baixe o novo certificado SSL/TLS conforme descrito em Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.
2. Atualize as aplicações para usarem o novo certificado SSL/TLS.

   Os métodos para atualizar aplicações para novos certificados SSL/TLS dependem de suas aplicações específicas. Trabalhe com os desenvolvedores de aplicações para atualizar os certificados SSL/TLS para suas aplicações.

   Para obter informações sobre como verificar conexões SSL/TLS e atualizar aplicações para cada mecanismo de banco de dados, consulte os seguintes tópicos:

   - Atualizar aplicações para conexão com clusters de banco de dados do Aurora MySQL usando novos certificados TLS
   - Atualizar aplicações para conexão com clusters de banco de dados PostgreSQL do Aurora usando novos certificados SSL/TLS

   Para conhecer um script de exemplo que atualiza um armazenamento confiável para um sistema operacional Linux, consulte Script de exemplo para importar certificados para o seu armazenamento confiável.

   > **Note**
   >
   > O pacote de certificados contém certificados tanto para a CA antiga como para a nova, portanto, é possível atualizar a aplicação de maneira segura e manter a conectividade durante o período de transição. Se você estiver usando o AWS Database Migration Service a fim de migrar um banco de dados para um cluster de banco de dados, recomendamos o uso do pacote de certificados para garantir a conectividade durante a migração.

3. Modifique a instância de banco de dados para alterar a CA de rds-ca-2019 para rds-ca-rsa2048-g1. Para verificar se o banco de dados requer reinicialização para atualizar os certificados de CA, use o comando describe-db-engine-versions e verifique o sinalizador `SupportsCertificateRotationWithoutRestart`.

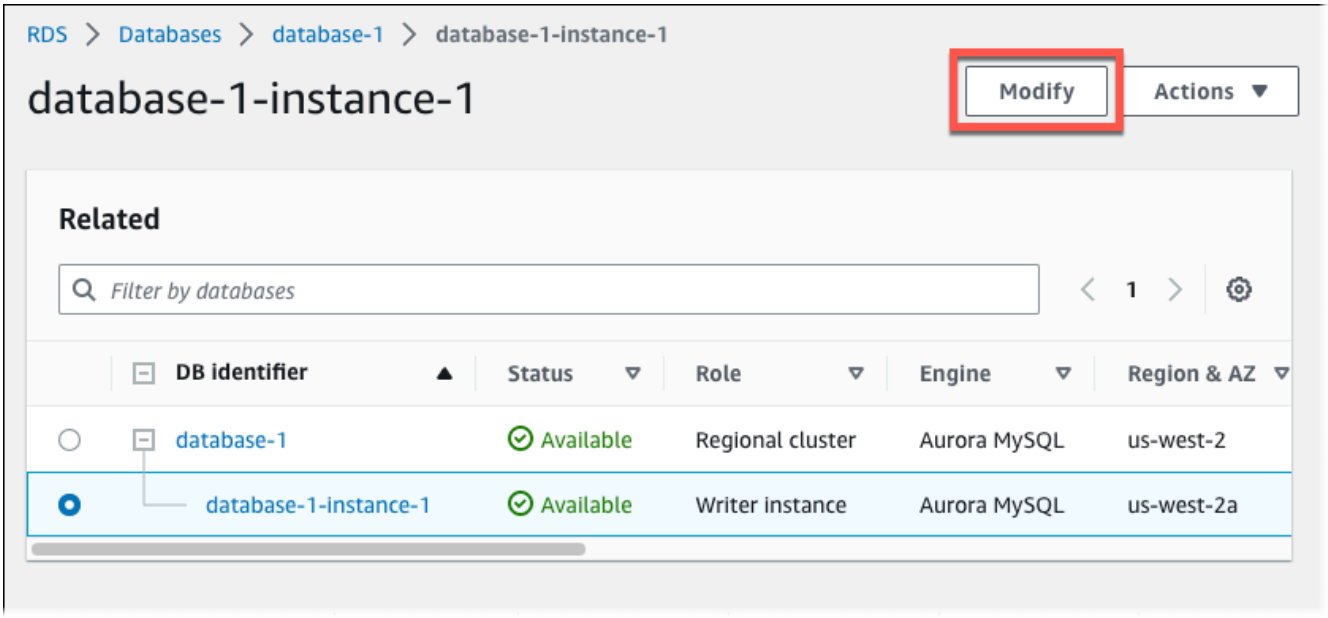
> **Note**
>
> Reinicialize o cluster do Babelfish depois de modificar para atualizar o certificado CA.

> **Important**
>
> Se você estiver enfrentando problemas de conectividade após a expiração do certificado, especifique a opção Apply immediately (Aplicar imediatamente) no console ou a opção `--apply-immediately` usando a AWS CLI. Por padrão, essa operação é programada para ser executada durante a próxima janela de manutenção.
> Para definir uma substituição de CA do clusterda instância que é diferente da CA padrão do RDS, use o comando modify-certificates da CLI.

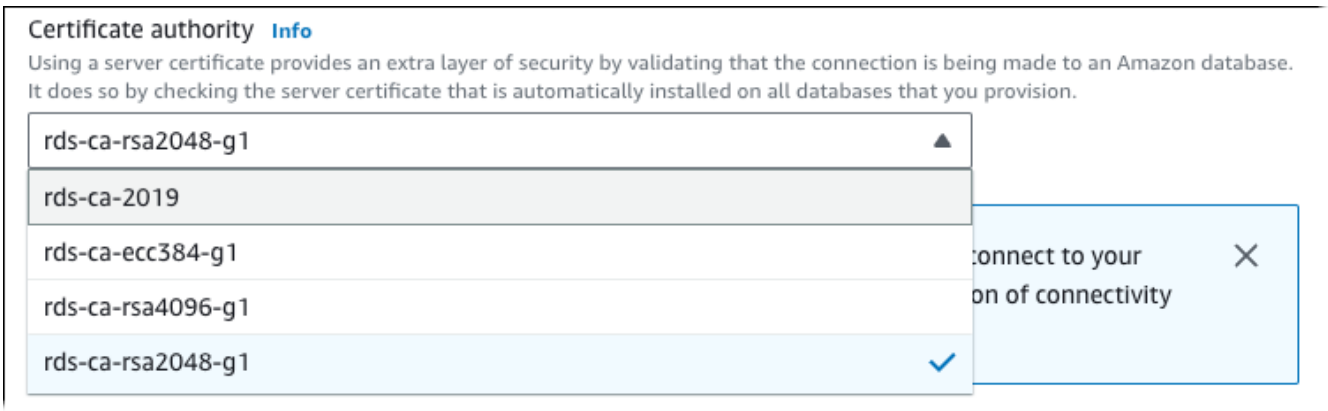
É possível usa o Console de gerenciamento da AWS ou a AWS CLI para alterar o certificado CA de rds-ca-2019 para rds-ca-rsa2048-g1 para uma instância de banco de dados .

Console

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Bancos de dados e selecione a instância de banco de dados que você deseja modificar.
3. Escolha Modificar.



4. Na seção Conectividade, escolha rds-ca-rsa2048-g1.



5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. Para aplicar as alterações imediatamente, escolha Apply immediately.
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modificar instância de banco de dados para salvar as alterações.

> **Important**
>
> Ao programar essa operação, certifique-se de ter atualizado o armazenamento de confiança do lado do cliente com antecedência.

Ou escolha Back (Voltar) para editar as alterações ou Cancel (Cancelar) para cancelar as alterações.

AWS CLI

Para usar a AWS CLI a fim de alterar a CA de rds-ca-2019 para rds-ca-rsa2048-g1 para uma instância de banco de dados , chame o comando modify-db-instance ou modify-db-cluster. Especifique o identificador da instância de banco de dados e a opção `--ca-certificate-identifier`.

Use o parâmetro `--apply-immediately` para aplicar a atualização imediatamente. Por padrão, essa operação é programada para ser executada durante a próxima janela de manutenção.

> **Important**
>
> Ao programar essa operação, certifique-se de ter atualizado o armazenamento de confiança do lado do cliente com antecedência.

Example

O exemplo a seguir modifica `mydbinstance` definindo o certificado CA como `rds-ca-rsa2048-g1`.

Para Linux, macOS ou Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> Se a instância exigir reinicialização, você poderá usar o comando modify-db-instance da CLI e especificar a opção `--no-certificate-rotation-restart`.

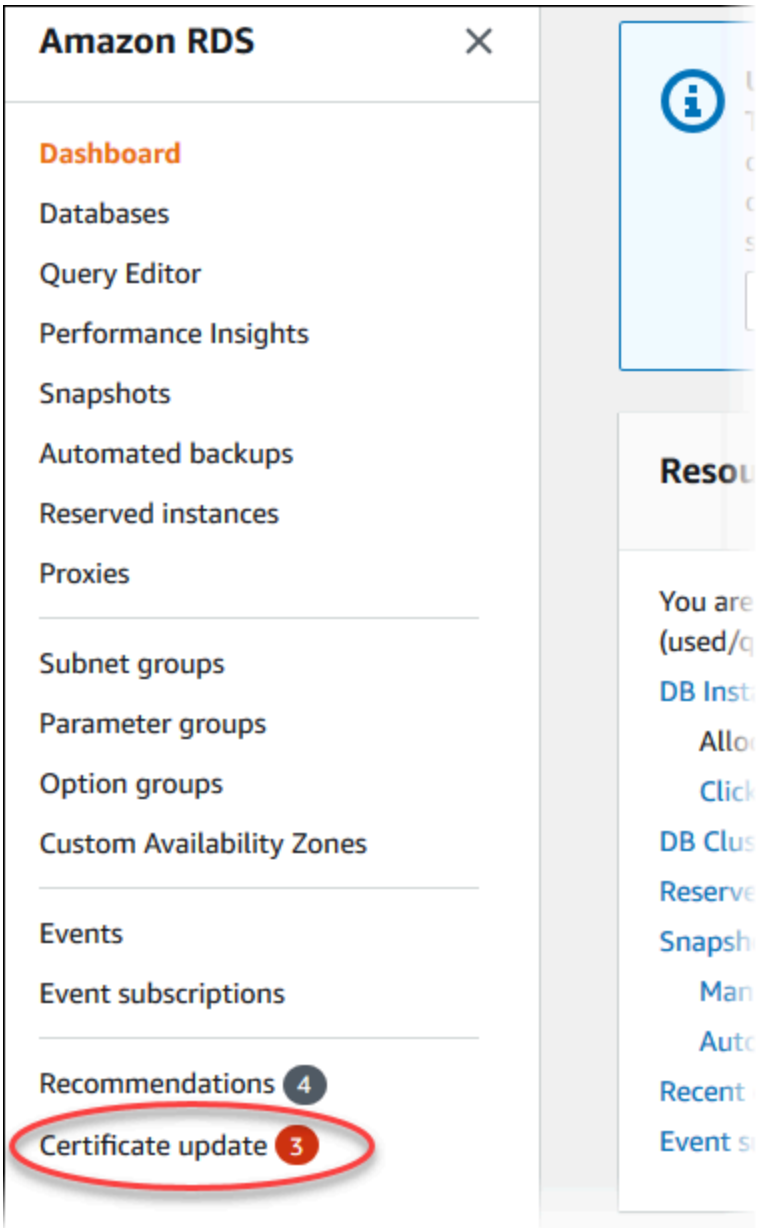
Atualizar seu certificado CA aplicando manutenção

Siga as etapas a seguir para atualizar o certificado CA aplicando a manutenção.

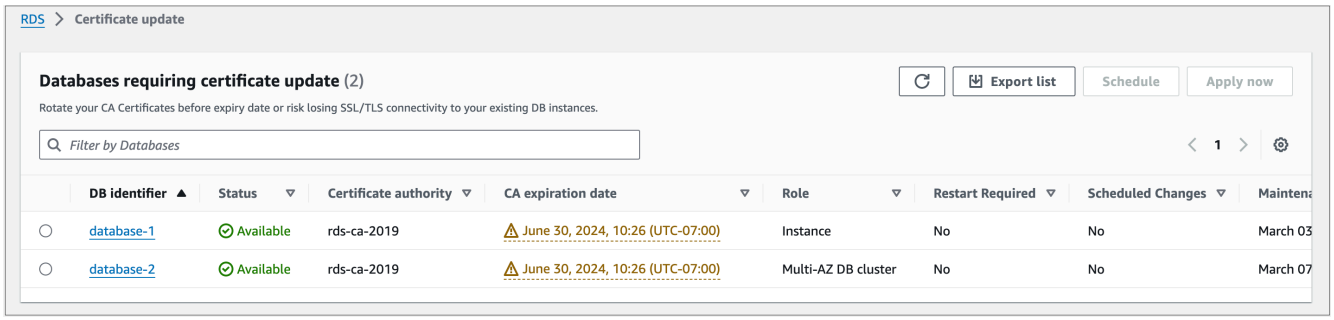
Console

Como atualizar o certificado CA aplicando a manutenção

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, selecione Atualização de certificado.



A página Bancos de dados que precisam de atualização de certificado é exibida.



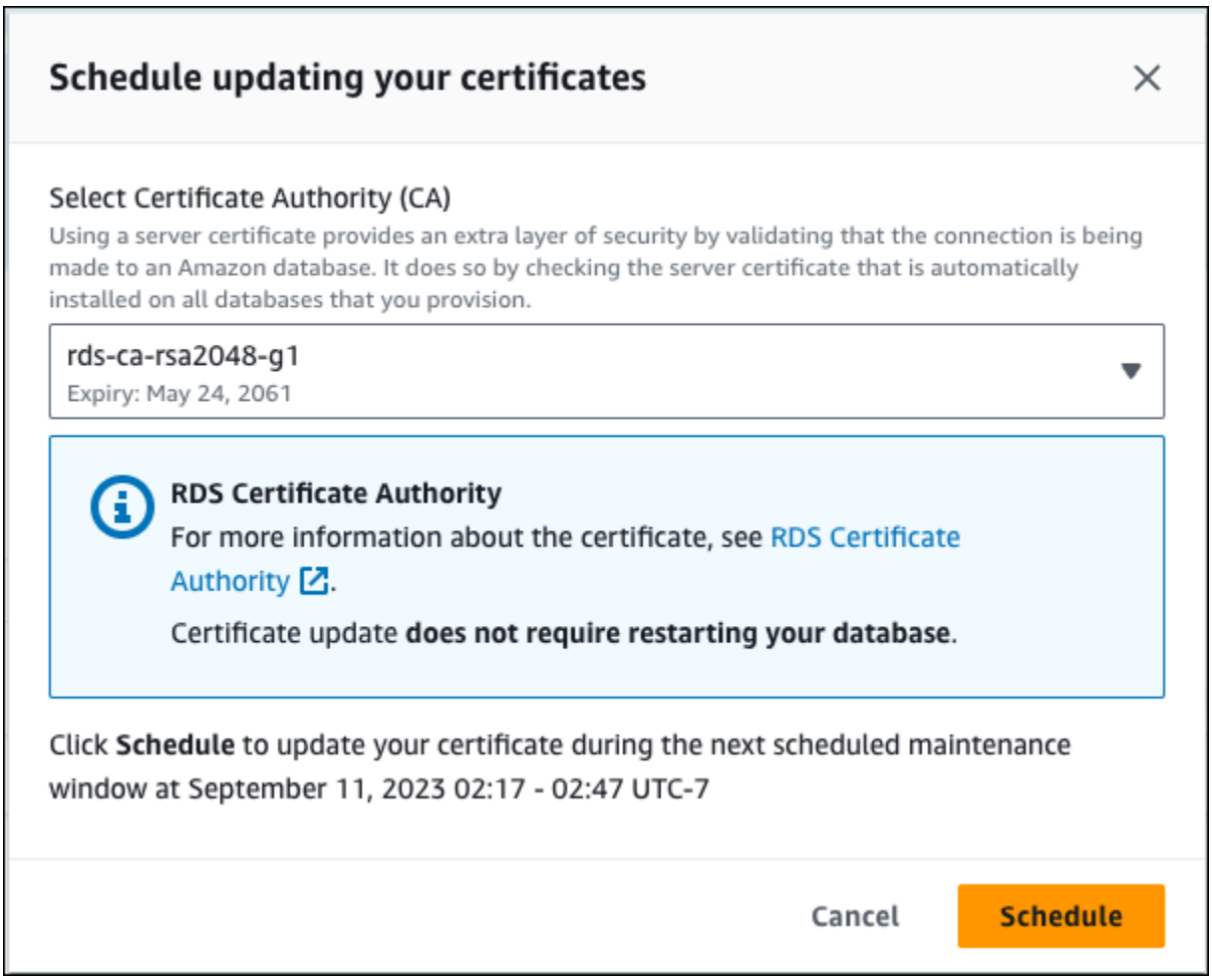
> **Note**
>
> Essa página mostra apenas as instâncias de banco de dados na Região da AWS atual. Se você tiver bancos de dados em mais de uma Região da AWS, confira essa página em cada Região da AWS para ver todas as instâncias de banco de dados com certificados SSL/TLS antigos.

3. Escolha a instância de banco de dados que você deseja atualizar.

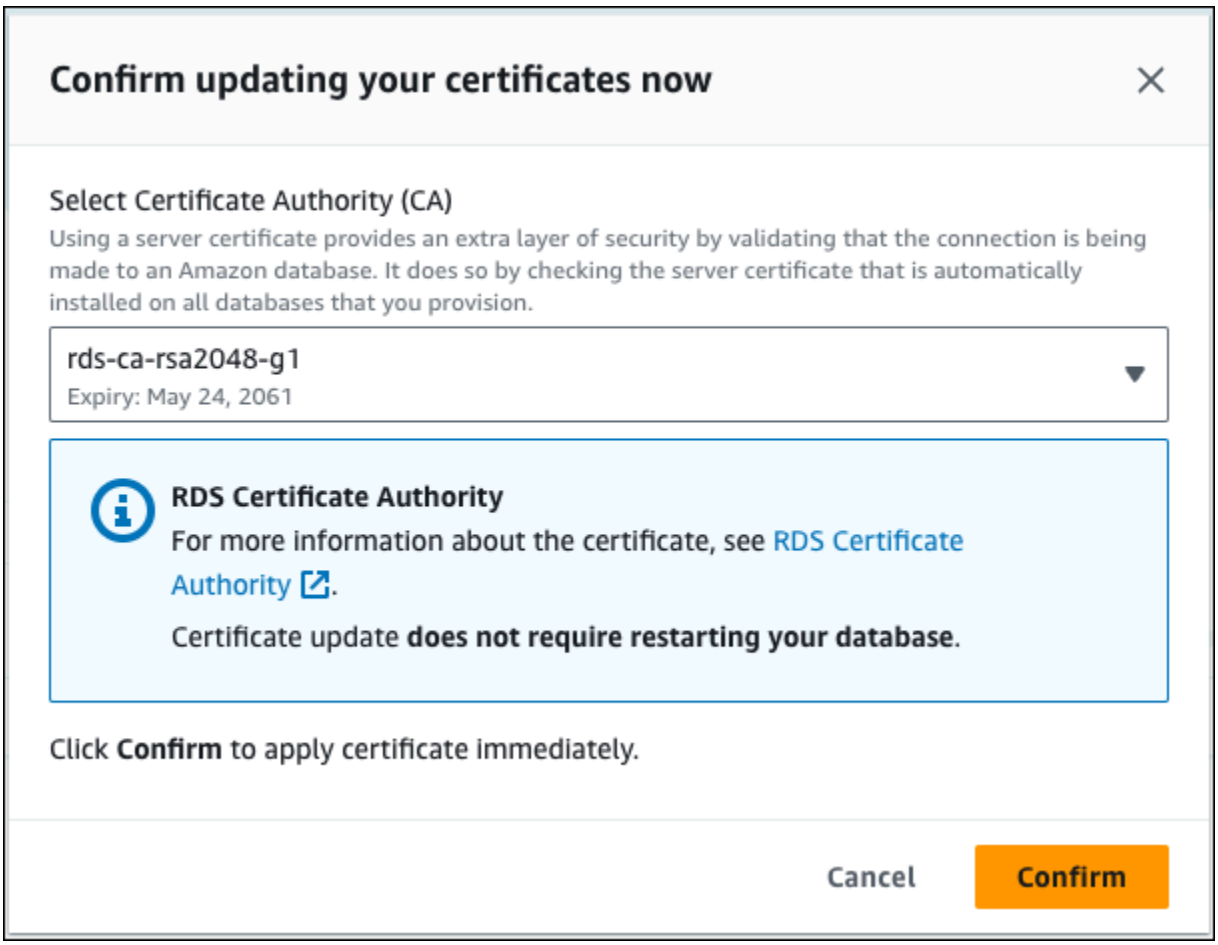
   Você pode programar a alternância de certificado para sua próxima janela de manutenção escolhendo Programar. Aplique a mudança imediatamente escolhendo Aplicar agora.

> **Important**
>
> Se você tiver problemas de conectividade após a expiração do certificado, use a opção Aplicar agora.

4. a. Se você escolher Programar, precisará confirmar a alternância do certificado de CA. Essa solicitação de confirmação também indica a janela agendada para sua atualização.



b. Se você escolher Aplicar agora, precisará confirmar a alternância do certificado de CA.



> **Important**
>
> Antes de programar a rotação do certificado CA no banco de dados, atualize todas as aplicações cliente que usam SSL/TLS e o certificado do servidor para se conectar. Essas atualizações são específicas ao seu mecanismo de banco de dados. Depois de atualizar essas aplicações cliente, você pode confirmar a rotação do certificado CA.

Para continuar, escolha a caixa de seleção e escolha Confirm (Confirmar).

5. Repita as etapas 3 e 4 para cada instância de banco de dados instância que você deseja atualizar.

Alternância automática de certificados do servidor

Se a CA raiz comportar a troca automática de certificados de servidor, o RDS gerenciará automaticamente a troca do certificado do servidor de banco de dados. Como o RDS usa a mesma CA raiz para essa alternância automática, então você não precisa baixar um novo pacote de CA. Consulte Autoridades certificadoras.

A alternância e a validade do certificado do servidor de banco de dados dependem do mecanismo de banco de dados:

- Se o mecanismo de banco de dados comportar a alternância sem reinicialização, o RDS alternará automaticamente o certificado do servidor de banco de dados sem exigir nenhuma ação de sua parte. O RDS tenta alternar o certificado do servidor de banco de dados em sua janela de manutenção preferida na meia-vida do respectivo certificado. O novo certificado do servidor de banco de dados é válido por 12 meses.
- Se seu mecanismo de banco de dados não aceitar alternância sem reinicialização, o Amazon RDS tornará visível uma ação de manutenção pendente `server-certificate-rotation` por meio da API Describe-pending-maintenance-actions, na meia-vida do certificado ou pelo menos 3 meses antes da expiração. É possível aplicar a alternância usando a API apply-pending-maintenance-action. O novo certificado do servidor de banco de dados é válido por 12 meses.

Use o comando describe-db-engine-versions e inspecione o sinalizador `SupportsCertificateRotationWithoutRestart` para identificar se a versão do mecanismo de banco de dados é compatível com a alternância de certificado sem reinicialização. Para ter mais informações, consulte Configurar a CA do banco de dados .

Script de exemplo para importar certificados para o seu armazenamento confiável

Veja os exemplos de scripts do shell que importam o pacote de certificados para um armazenamento de confiança.

Cada script de shell de amostra usa o keytool, que faz parte do Java Development Kit (JDK). Para obter mais informações sobre como instalar o JDK, consulte o Guia de instalação do JDK.

Linux

Veja a seguir um exemplo de script shell que importa o pacote de certificados para um armazenamento confiável em um sistema operacional Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

Veja a seguir um exemplo de script do shell que importa o pacote de certificados em um armazenamento de confiança no macOS.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit
```

```
curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Para saber mais sobre as práticas recomendadas sobre o uso de SSL com o Amazon RDS, consulte Best practices for successful SSL connections to Amazon RDS for Oracle .

Privacidade do tráfego entre redes

As conexões são protegidas entre o Amazon Aurora e as aplicações on-premises e entre o Amazon Aurora e outros recursos da AWS na mesma região da AWS.

Tráfego entre clientes de serviço e on-premises e as aplicações

Você tem duas opções de conectividade entre sua rede privada e a AWS:

- Uma conexão AWS Site-to-Site VPN. Para ter mais informações, consulte O que é o AWS Site-to-Site VPN?
- Uma conexão Direct Connect. Para ter mais informações, consulte O que é o Direct Connect?

Você obtém acesso ao Amazon Aurora pela rede usando operações de API publicadas pela AWS. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Gerenciamento de identidade e acesso no Amazon Aurora

O AWS Identity and Access Management (IAM) é um serviço da AWS service (Serviço da AWS) que ajuda um administrador a controlar com segurança o acesso aos recursos da AWS. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para utilizar os recursos do Amazon RDS. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

Tópicos

- Público
- Autenticar com identidades
- Gerenciamento do acesso usando políticas
- Como o Amazon Aurora funciona com o IAM
- Exemplos de políticas baseadas em identidade do Amazon Aurora
- AWSPolíticas gerenciadas pela para o Amazon RDS
- Atualizações do Amazon RDS para políticas gerenciadas pela AWS
- Prevenção do problema do substituto confuso entre serviços
- Autenticação do banco de dados do IAM
- Solução de problemas de identidade e acesso do Amazon Aurora

Público

A forma de usar o AWS Identity and Access Management (IAM) varia em função do trabalho realizado no Amazon Aurora.

Usuário do serviço: se você usar o serviço Aurora para fazer sua tarefa, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que usar mais recursos do Aurora para fazer seu trabalho, você poderá precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se não for possível acessar um recurso no Aurora, consulte Solução de problemas de identidade e acesso do Amazon Aurora.

Administrador do serviço: se você for o responsável pelos recursos do Aurora em sua empresa, você provavelmente terá acesso total ao Aurora. Seu trabalho é determinar quais recursos do Aurora seus

funcionários devem acessar. Assim, é necessário enviar solicitações ao administrador do para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com o Aurora, consulte Como o Amazon Aurora funciona com o IAM.

Administrador: se você é um administrador, talvez queira saber detalhes sobre como pode escrever políticas para gerenciar o acesso ao Aurora. Para visualizar exemplos de políticas baseadas em identidade do Aurora que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon Aurora.

Autenticar com identidades

A autenticação é a forma como fazer login na AWS usando suas credenciais de identidade. Você precisa se autenticar como o Usuário raiz da conta da AWS, como um usuário do IAM ou assumindo um perfil do IAM.

É possível fazer login como uma identidade federada usando credenciais de uma fonte de identidade, como o Centro de Identidade do AWS IAM (Centro de Identidade do IAM), autenticação única ou credenciais do Google/Facebook. Para ter mais informações sobre como fazer login, consulte How to sign in to your Conta da AWS no Guia do usuário do Início de Sessão da AWS.

Para acesso programático, a AWS oferece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Usuário raiz de conta da AWS

Ao criar uma Conta da AWS, você começa com uma única identidade de login chamada usuário-raiz da Conta da AWS, que tem acesso total a todos os recursos e Serviços da AWS. É altamente recomendável não usar o usuário-raiz para tarefas diárias. Para ver as tarefas que exigem credenciais de usuário-raiz, consulte Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidades para acessar a Serviços da AWS usando credenciais temporárias.

Identidade federada é um usuário do seu diretório empresarial, de um provedor de identidades da web ou do Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem perfis que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, é recomendável usar o Centro de Identidade do AWS IAM. Para obter mais informações, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM.

Usuários e grupos do IAM

Usuário do IAM é uma identidade com permissões específicas a uma única pessoa ou aplicação. Recomendamos usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que os usuários humanos usem a federação com um provedor de identidade para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Você pode se autenticar no cluster de banco de dados usando a autenticação de banco de dados do IAM.

A autenticação do banco de dados do IAM funciona com o Aurora. Para obter mais informações sobre a autenticação no cluster usando o IAM, consulte Autenticação do banco de dados do IAM.

Perfis do IAM

Um perfil do IAM é uma identidade na Conta da AWS que tem permissões específicas. Ela é semelhante a um usuário, mas não está associada a uma pessoa específica. É possível presumir temporariamente um perfil do IAM no Console de gerenciamento da AWS alternando perfis. É possível presumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Permissões temporárias para usuários: um usuário pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidade de terceiros (federação) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Guia do Usuário do Centro de Identidade do AWS IAM.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar um perfil como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Como os perfis do IAM diferem das políticas baseadas em recurso no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam atributos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Sessões de acesso direto: as sessões de acesso direto (FAS) usam as permissões da entidade principal chamando um AWS service (Serviço da AWS), além de solicitar que o AWS service (Serviço da AWS) faça solicitações a serviços de downstream. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
  - Perfil vinculado a serviço: um perfil vinculado a serviço é um tipo de perfil de serviço vinculado a um AWS service (Serviço da AWS). O serviço pode presumir o perfil para executar uma ação em seu nome. Perfis vinculadas ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados a serviço.
- Aplicações em execução no Amazon EC2: é possível usar um perfil do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer

solicitações da AWS CLI ou da API da AWS. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir um perfil da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se você deve usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciamento do acesso usando políticas

Você controla o acesso na AWS criando e anexando políticas às identidades do IAM ou aos recursos da AWS. Uma política é um objeto na AWS que, quando associada a uma identidade ou a um recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade (usuário raiz, usuário ou perfil do IAM) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Um administrador pode usar políticas para especificar quem tem acesso aos recursos da AWS e quais ações essas pessoas podem executar nesses recursos. Cada entidade do IAM (conjunto de permissões ou perfil) começa sem permissões. Em outras palavras, por padrão, os usuários não podem fazer nada, nem mesmo alterar sua própria senha. Para dar permissão a um usuário para fazer algo, um administrador deve anexar uma política de permissões ao usuário. Ou o administrador pode adicionar o usuário a um grupo que tenha as permissões pretendidas. Quando um administrador concede permissões a um grupo, todos os usuários desse grupo recebem essas permissões.

As políticas do IAM definem permissões para uma ação, independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de perfis do Console de gerenciamento da AWS, da AWS CLI ou da API AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos JSON de políticas de permissões que você pode anexar a uma identidade, como um conjunto de permissões ou um perfil. Essas políticas

controlam quais ações cada identidade pode realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único conjunto de permissões ou perfil. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários conjuntos de permissões e perfis em sua conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Para obter informações sobre políticas gerenciadas pela AWS específicas do Amazon Aurora, consulte AWSPolíticas gerenciadas pela para o Amazon RDS.

Outros tipos de política

A AWS oferece compatibilidade com tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (conjunto de permissões ou perfil). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade da entidade e seus limites de permissões. As políticas baseadas em recurso que especificam o conjunto de permissões ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs): SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço para agrupar e gerenciar centralmente várias contas da AWS pertencentes à sua empresa. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas membro, o que inclui cada Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte Como os SCPs Funcionam no Manual do Usuário do AWS Organizations.

- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção entre as políticas baseadas em identidade dos conjuntos de permissões ou do perfil e as políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir ou não uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliação de políticas no Guia do usuário do IAM.

Como o Amazon Aurora funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon Aurora, você precisa entender quais recursos do IAM estão disponíveis para uso com o Aurora.

A tabela a seguir lista os recursos do IAM que você pode usar com o Amazon Aurora:

Recurso do IAM	Compatibilidade do Amazon Aurora
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
Controle de acesso baseado em atributos (ABAC) (tags em políticas)	Sim

Recurso do IAM	Compatibilidade do Amazon Aurora
Credenciais temporárias	Sim
Sessões de acesso direto	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Sim

Para obter uma visão de alto nível de como o Amazon Aurora e outros serviços da AWS funcionam com o IAM, consulte Serviços da AWS compatíveis com o IAM no Guia do usuário do IAM.

Tópicos

- Políticas baseadas em identidade do Aurora
- Políticas baseadas em recursos do Aurora
- Ações de políticas para o Aurora
- Recursos de políticas do Aurora
- Chaves de condição de políticas do Aurora
- Listas de controle de acesso (ACLs) no Aurora
- Controle de acesso baseado em atributos (ABAC) em políticas com tags do Aurora
- Usar credenciais temporárias com o Aurora
- Sessões de acesso direto para o Aurora
- Perfis de serviço do Aurora
- Funções vinculadas a serviço do Aurora

Políticas baseadas em identidade do Aurora

Compatível com políticas baseadas em identidade: sim.

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade do Aurora

Para visualizar exemplos de políticas baseadas em identidade do Aurora, consulte Exemplos de políticas baseadas em identidade do Amazon Aurora.

Políticas baseadas em recursos do Aurora

Compatível com políticas baseadas em recursos: não.

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações de políticas para o Aurora

Compatível com ações de políticas: sim.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de políticas no Aurora usam o seguinte prefixo antes da ação: `rds:`. Por exemplo, para conceder a alguém permissão para descrever instâncias de banco de dados com a operação da API `DescribeDBInstances` do Amazon RDS, inclua a ação `rds:DescribeDBInstances` na política.

As instruções de política devem incluir um elemento `Action` ou `NotAction`. O Aurora define seu próprio conjunto de ações que descrevem as tarefas que você pode executar com esse serviço.

Para especificar várias ações em uma única declaração, separe-as com vírgulas, conforme a seguir.

```
"Action": [
      "rds:action1",
      "rds:action2"
```

Você também pode especificar várias ações utilizando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a ação a seguir:

```
"Action": "rds:Describe*"
```

Para obter uma lista de ações do Aurora, consulte Ações definidas pelo Amazon RDS na Referência de autorização do serviço.

Recursos de políticas do Aurora

Compatível com recursos de políticas: sim.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

O recurso de instância de banco de dados tem o nome do recurso da Amazon (ARN) a seguir.

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

Para obter mais informações sobre o formato de ARNs, consulte Nomes de recursos da Amazon (ARNs) e namespaces de serviços da AWS.

Por exemplo, para especificar a instância de banco de dados `dbtest` em sua instrução, use o ARN a seguir.

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

Para especificar todas as instâncias de banco de dados que pertencem a uma conta específica, use o caractere curinga (*).

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

Algumas operações da API do RDS, como as operações para a criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, use o caractere curinga (*).

```
"Resource": "*"
```

Muitas operações da API do Amazon RDS envolvem vários recursos. Por exemplo, o `CreateDBInstance` cria uma instância de banco de dados. Você pode especificar que um usuário do deve usar um grupo de segurança e um grupo de parâmetros específicos ao criar uma instância de banco de dados. Para especificar vários recursos em uma única declaração, separe os ARNs com vírgulas.

```
"Resource": [
      "resource1",
      "resource2"
```

Para ver uma lista dos tipos de recursos do Aurora e seus ARNs, consulte Tipos de recursos definidos pelo Amazon RDS na Referência de autorização do serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon RDS.

Chaves de condição de políticas do Aurora

Compatível com chaves de condição de política específicas de serviço: sim.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição globais da AWS, consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM.

O Aurora define seu próprio conjunto de chaves de condição e também é compatível com o uso de algumas chaves de condição globais. Para ver todas as chaves de condição globais da AWS, consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM.

Todas as operações da API do RDS oferecem suporte à chave de condição `aws:RequestedRegion`.

Para ver uma lista das chaves de condição do Aurora, consulte Chaves de condição do Amazon RDS na Referência de autorização do serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon RDS.

Listas de controle de acesso (ACLs) no Aurora

Compatível com listas de controle de acesso (ACLs): não.

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Controle de acesso baseado em atributos (ABAC) em políticas com tags do Aurora

Compatível com tags de controle de acesso baseado em atributos (ABAC) em políticas: sim.

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades do IAM e recursos da AWS e, em seguida, projetar políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Para obter mais informações sobre recursos de marcação do Aurora, consulte Especificar condições: usar tags personalizadas. Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Conceder permissão para ações em um recurso com uma tag específica com dois valores diferentes.

Usar credenciais temporárias com o Aurora

Compatível com credenciais temporárias: sim.

As credenciais temporárias dão acesso de curto prazo aos recursos da AWS e são criadas automaticamente quando você usa a federação ou alterna os perfis. A AWS recomenda a você gerar credenciais temporárias dinamicamente, em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Sessões de acesso direto para o Aurora

Compatível com sessões de acesso direto: sim.

As sessões de acesso direto (FAS) usam as permissões da entidade principal chamando um AWS service (Serviço da AWS), bem como o AWS service (Serviço da AWS) solicitante, para fazer solicitações a serviços subsequentes. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Perfis de serviço do Aurora

Compatível com perfis de serviço: sim.

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> A alteração das permissões de uma função de serviço pode interromper a funcionalidade do Aurora. Edite perfis de serviço somente quando o Aurora fornecer orientação para isso.

Funções vinculadas a serviço do Aurora

Compatível com perfis vinculados a serviços: sim.

Um perfil vinculado a serviço é um tipo de perfil de serviço vinculado a um AWS service (Serviço da AWS). O serviço pode presumir o perfil de executar uma ação em seu nome. Perfis vinculados ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como usar funções vinculadas o serviço do Aurora, consulte Usar funções vinculadas ao serviço do Amazon Aurora.

Exemplos de políticas baseadas em identidade do Amazon Aurora

Por padrão, os conjuntos de permissões e perfis não têm permissão para criar nem modificar recursos do Aurora. Eles também não podem executar tarefas usando o Console de gerenciamento da AWS, a AWS CLI ou uma API da AWS. Um administrador deve criar políticas do IAM que concedam aos conjuntos de permissões e perfis permissão para executar operações de API específicas nos recursos especificados de que precisam. Depois, o administrador deve anexar essas políticas aos conjuntos de permissões e perfis que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM utilizando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Tópicos

- Práticas recomendadas de política
- Usar o console do Aurora
- Permissões necessárias para usar o console
- Permitir que os usuários visualizem suas próprias permissões
- Políticas de permissão para criar, modificar e excluir recursos no Aurora
- Políticas de exemplo: usar chaves de condição
- Especificar condições: usar tags personalizadas
- Conceder permissão para marcar recursos do Aurora durante a criação

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon RDS em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas gerenciadas pela AWS e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as políticas gerenciadas pela AWS, que concedem permissões para muitos casos de uso comuns. Elas estão disponíveis em seus Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS que são específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Também pode usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um AWS service (Serviço da AWS) específico, como o CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Aurora

Para acessar o console do Amazon Aurora, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon Aurora em sua Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Não é necessário conceder permissões mínimas do console para usuários que fazem chamadas somente à AWS CLI ou à API do AWS. Em vez disso, permita o acesso somente às ações que corresponderem a operação da API que você estiver tentando executar.

Para garantir que essas entidades ainda possam usar o console do Aurora, anexe também a seguinte política gerenciada pela AWS às entidades.

```
AmazonRDSReadOnlyAccess
```

Para obter mais informações, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Permissões necessárias para usar o console

Para um usuário trabalhar com o console, esse usuário deve ter um conjunto de permissões mínimo. Essas permissões permitem que o usuário descreva os recursos do Amazon Aurora para a conta da AWS e forneça outras informações relacionadas, inclusive informações de segurança e rede do Amazon EC2.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda consigam usar o console, associe também a política gerenciada `AmazonRDSReadOnlyAccess` ao usuário, conforme descrito em Gerenciamento do acesso usando políticas.

Não é necessário conceder permissões mínimas do console para usuários que fazem chamadas somente à AWS CLI ou à API do Amazon RDS.

A seguinte política concede acesso completo a todos os recursos do Amazon Aurora para a conta raiz da AWS:

```
AmazonRDSFullAccess
```

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a API da AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Políticas de permissão para criar, modificar e excluir recursos no Aurora

As seções a seguir apresentam exemplos de políticas de permissão que concedem e restringem o acesso aos recursos:

Permitir que um usuário crie instâncias de Bancos de Dados em uma conta da AWS

Veja a seguir um exemplo de política que permite à conta com o ID `123456789012` criar instâncias de banco de dados para a sua conta da AWS. A política exige que o nome da nova instância de banco de dados comece com `test`. A nova instância de banco de dados também deve usar o mecanismo de banco de dados MySQL e a classe de instância de banco de dados `db.t2.micro`. Além disso, a nova instância de banco de dados deve usar um grupo de opções e um grupo de parâmetros de banco de dados que começa com `default` e deve usar o grupo de sub-redes `default`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
```

```
    ]
}
```

A política inclui uma única instrução que especifica as seguintes permissões para o usuário do :

- A política permite que a conta crie uma instância de banco de dados usando a operação de API CreateDBInstance (isso também se aplica ao comando create-db-instance da AWS CLI e ao Console de gerenciamento da AWS).
- O elemento `Resource` especifica que o usuário pode realizar ações em ou com recursos. Você especifica recursos usando um nome de recurso da Amazon (ARN). O ARN inclui o nome do serviço ao qual o recurso pertence (`rds`), a região da AWS (`*` indica qualquer região neste exemplo), o número de conta da AWS (`123456789012` é o número de conta neste exemplo) e o tipo de recurso. Para obter mais informações sobre como criar ARNs, consulte Nomes de recurso da Amazon (ARNs) no Amazon RDS.

  O elemento `Resource` neste exemplo especifica as restrições da política a seguir em recursos para o usuário:

  - O identificador de instância de banco de dados para a nova instância de banco de dados deve começar com `test` (por exemplo, `testCustomerData1`, `test-region2-data`).
  - O grupo de opções para a nova instância de banco de dados deve começar com `default`.
  - O grupo de parâmetros de banco de dados para a nova instância de banco de dados deve começar com `default`.
  - O grupo de sub-redes para a nova instância de banco de dados deve ser o grupo de sub-redes `default`.
- O elemento `Condition` especifica que o mecanismo de banco de dados deve ser MySQL e a classe da instância de banco de dados deve ser `db.t2.micro`. O elemento `Condition` especifica as condições quando uma política deve entrar em vigor. Você pode adicionar permissões ou restrições usando o elemento `Condition`. Para obter mais informações sobre como especificar condições, consulte Chaves de condição de políticas do Aurora. Este exemplo especifica condições `rds:DatabaseEngine` e `rds:DatabaseClass`. Para obter informações sobre valores de condição válidos para o `rds:DatabaseEngine`, consulte a lista no parâmetro `Engine` em CreateDBInstance. Para obter informações sobre os valores de condição válidos para `rds:DatabaseClass`, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.

A política não especifica o elemento `Principal` porque, em uma política baseada em identidade, não se especifica o principal que obtém as permissões. Quando você anexar uma política um usuário, o usuário será a entidade principal implícita. Quando você anexa uma política de permissão a um perfil do IAM, o principal identificado na política de confiança do perfil obtém as permissões.

Para obter uma lista de ações do Aurora, consulte Ações definidas pelo Amazon RDS na Referência de autorização do serviço.

Permitir que um usuário execute qualquer ação de descrição em qualquer recurso do RDS

A seguinte política de permissões concede permissões a um usuário para executar todas as ações que começam com `Describe`. Essas ações mostram informações sobre um recurso do RDS, como uma instância de banco de dados. O caractere curinga (*) no elemento `Resource` indica que as ações são permitidas para todos os recursos do Amazon Aurora que pertencem à conta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

Permitir que um usuário crie uma instância de banco de dados que usa o grupo de parâmetros de banco de dados e o grupo de sub-redes especificados.

A seguinte política de permissões concede permissões para permitir que um usuário crie apenas uma instância de banco de dados que deve usar o grupo de parâmetros de banco de dados `mydbpg` e o grupo de sub-rede de banco de dados `mydbsubnetgroup`.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

Conceder permissão para ações em um recurso com uma tag específica com dois valores diferentes

Você pode usar condições em sua política baseada em identidade para controlar o acesso aos recursos do Aurora com base em tags. A política a seguir concede permissão para realizar a operação de API `CreateDBSnapshot` em instâncias de banco de dados com a etiqueta `stage` definida como `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
```

```
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

A política a seguir concede permissão para realizar a operação de API `ModifyDBInstance` em instâncias de banco de dados com a etiqueta `stage` definida como `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
```

```
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Impedir que um usuário exclua uma instância de banco de dados

A seguinte política de permissões concede permissões para impedir que um usuário exclua uma instância de banco de dados específica. Por exemplo, você pode querer negar a capacidade de excluir suas instâncias de banco de dados de produção para qualquer usuário que não seja um administrador.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Negar todo o acesso a um recurso

É possível negar acesso explicitamente a um recurso. As políticas de negação têm precedência sobre as políticas de permissão. A política a seguir nega explicitamente a um usuário a capacidade de gerenciar um recurso:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

Políticas de exemplo: usar chaves de condição

Os seguintes exemplos mostram como você pode usar chaves de condição em políticas de permissões do IAM do Amazon Aurora.

Exemplo 1: conceder permissão para criar uma instância de banco de dados que usa um mecanismo de banco de dados específico e não é MultiAZ

A seguinte política usa uma chave de condição do RDS e permite que um usuário crie apenas instâncias de banco de dados que usam o mecanismo de banco de dados MySQL e não use o MultiAZ. O elemento `Condition` indica a exigência de que o mecanismo de banco de dados seja MySQL.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
```

```
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Exemplo 2: negar explicitamente a permissão para criar instâncias de bancos de dados para determinadas classes de instância de banco de dados e criar instâncias de bancos de dados que usam IOPS provisionadas

A seguinte política nega explicitamente a permissão para criar instâncias de bancos de dados que usam as classes de instância de banco de dados `r3.8xlarge` e `m4.10xlarge`, que são as classes de instância de banco de dados maiores e mais caras. Essa política também impede que os usuários criem instâncias de banco de dados que usam IOPS provisionadas, que resultam em custos adicionais.

A negação explícita da permissão substitui quaisquer outras permissões concedidas. Isso garante que as identidades não obtenham acidentalmente permissão que você nunca deseja conceder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
```

```
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
            }
         }
      }
   ]
}
```

Exemplo 3: limitar o conjunto de chaves de tag e valores que podem ser usados para identificar um recurso

A política a seguir usa uma chave de condição do RDS e permite a adição de uma tag com a chave `stage` a ser adicionada a um recurso com os valores `test`, `qa` e `production`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagEdits",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "arn:aws:rds:us-east-1:123456789012:db:db-123456",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/stage": [
            "test",
            "qa",
            "production"
          ]
        }
```

```
      }
    }
  ]
}
```

Especificar condições: usar tags personalizadas

O Amazon Aurora oferece suporte para especificar condições em uma política do IAM usando tags personalizadas.

Por exemplo, suponha que você adicione uma tag chamada `environment` às suas instâncias de banco de dados com valores como `beta`, `staging`, `production` e assim por diante. Se fizer isso, você poderá criar uma política que restrinja determinados usuários a instâncias de banco de dados com base no valor da tag `environment`.

> **Note**
>
> Os identificadores de tags personalizados diferenciam maiúsculas de minúsculas.

A tabela a seguir lista os identificadores de tags do RDS que você pode usar em um elemento `Condition`.

Identificador de tag do RDS	Aplica-se a
`db-tag`	Instâncias de bancos de dados, incluindo réplicas de leitura
`snapshot-tag`	DB snapshots
`ri-tag`	Instâncias de bancos de dados reservadas
`og-tag`	Grupos de opções de banco de dados
`pg-tag`	Grupos de parâmetros do banco de dados
`subgrp-tag`	Grupos de sub-redes de banco de dados

Identificador de tag do RDS	Aplica-se a
`es-tag`	Assinaturas de eventos
`cluster-tag`	clusters de banco de dados
`cluster-pg-tag`	Grupos de parâmetros de cluster de banco de dados
`cluster-snapshot-tag`	Snapshots de cluster de banco de dados

A sintaxe de uma condição de tag personalizada é a seguinte:

`"Condition":{"StringEquals":{"rds:`*`rds-tag-identifier`*`/`*`tag-name`*`": ["`*`value`*`"]} }`

Por exemplo, o seguinte elemento `Condition` se aplica a instâncias de banco de dados com uma tag `environment` e um valor de tag de `production`.

`"Condition":{"StringEquals":{"rds:db-tag/`**`environment`**`": ["`**`production`**`"]} }`

Para obter informações sobre como criar tags, consulte Marcar recursos do Amazon Aurora e do Amazon RDS.

Important

Se você gerenciar o acesso aos recursos do RDS usando tags, recomendamos proteger o acesso às tags para os seus recursos do RDS. Você pode gerenciar o acesso a tags, criando políticas para as ações `AddTagsToResource` e `RemoveTagsFromResource`. Por exemplo, a seguinte política nega aos usuários a capacidade de adicionar ou remover tags para todos os recursos. Você pode então criar políticas para permitir que usuários específicos adicionem ou removam tags.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
```

```
            "Sid":"DenyTagUpdates",
            "Effect":"Deny",
            "Action":[
               "rds:AddTagsToResource",
               "rds:RemoveTagsFromResource"
            ],
            "Resource":"*"
         }
      ]
}
```

Para obter uma lista de ações do Aurora, consulte Ações definidas pelo Amazon RDS na Referência de autorização do serviço.

Políticas de exemplo: usar tags personalizadas

Os seguintes exemplos mostram como você pode usar tags personalizadas em políticas de permissões do IAM do Amazon Aurora. Para obter mais informações sobre como adicionar tags a um recurso do Amazon Aurora, consulte Nomes de recurso da Amazon (ARNs) no Amazon RDS.

> **Note**
>
> Todos os exemplos usam a região us-west-2 e contêm IDs de conta fictícios.

Exemplo 1: conceder permissão para ações em um recurso com uma tag específica com dois valores diferentes

A política a seguir concede permissão para realizar a operação de API `CreateDBSnapshot` em instâncias de banco de dados com a etiqueta `stage` definida como `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
```

```
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

A política a seguir concede permissão para realizar a operação de API `ModifyDBInstance` em instâncias de banco de dados com a etiqueta `stage` definida como `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
          "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
```

```
            "arn:aws:rds:*:123456789012:og:*"
            ]
      },
      {
        "Sid":"AllowDevTestToModifyInstance",
        "Effect":"Allow",
        "Action":[
           "rds:ModifyDBInstance"
           ],
        "Resource":"arn:aws:rds:*:123456789012:db:*",
        "Condition":{
           "StringEquals":{
              "rds:db-tag/stage":[
                 "development",
                 "test"
                 ]
              }
           }
      }
   ]
}
```

Exemplo 2: negar explicitamente a permissão para criar uma instância de banco de dados que usa grupos de parâmetros de banco de dados especificados

A seguinte política nega explicitamente a permissão para criar uma instância de banco de dados que usa grupos de parâmetros de banco de dados com valores de tag específicos. Você poderá aplicar essa política se precisar que um grupo de parâmetros de banco de dados específico criado pelo cliente sempre seja usado ao criar instâncias de bancos de dados. As políticas que utilizam `Deny` são mais frequentemente usadas para restringir o acesso que foi concedido por uma política mais ampla.

A negação explícita da permissão substitui quaisquer outras permissões concedidas. Isso garante que as identidades não obtenham acidentalmente permissão que você nunca deseja conceder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

Exemplo 3: conceder permissão para executar ações em uma instância de banco de dados com um nome de instância prefixado com um nome de usuário

A seguinte política permite chamar qualquer API (exceto para `AddTagsToResource` ou `RemoveTagsFromResource`) em uma instância de banco de dados que tem um nome prefixado com o nome do usuário e que tem uma tag `stage` igual a `devo` ou sem tag `stage`.

A linha `Resource` na política identifica um recurso pelo seu Nome de Recurso Amazon (ARN). Para obter mais informações sobre como usar ARNs com recursos do Amazon Aurora, consulte Nomes de recurso da Amazon (ARNs) no Amazon RDS.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
```

```
            }
         }
      }
   ]
}
```

Conceder permissão para marcar recursos do Aurora durante a criação

Algumas operações de API do RDS permitem atribuir tags ao criar recursos. É possível usar tags de recursos para implementar o controle baseado em atributo (ABAC). Para ter mais informações, consulte What is ABAC for AWS? e Controlar o acesso a recursos da AWS usando tags.

Para permitir que os usuários marquem recursos na criação, eles devem ter permissões para usar a ação que cria o recurso, como `rds:CreateDBCluster`. Se as tags forem especificadas no ato da criação, o RDS concederá uma autorização adicional na ação `rds:AddTagsToResource` para verificar se os usuários têm permissões para criar tags. Portanto, os usuários também precisam ter permissões para usar a ação `rds:AddTagsToResource`.

Na definição de política do IAM para a ação `rds:AddTagsToResource`, você pode usar a chave de condição `aws:RequestTag` para exigir tags em uma solicitação para marcar um recurso.

Por exemplo, a seguinte política permite que os usuários criem instâncias de banco de dados e apliquem tags durante a criação, mas somente com chaves de tag específicas (`environment` ou `project`):

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
```

```
                "rds:AddTagsToResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": ["production", "development"],
                    "aws:RequestTag/project": ["dataanalytics", "webapp"]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": ["environment", "project"]
                }
            }
        }
    ]
}
```

Essa política nega qualquer solicitação de criação de instância de banco de dados que inclua tags diferentes das tags `environment` ou `project` ou que não especifique nenhuma dessas tags. Além disso, os usuários devem especificar valores para as tags que correspondam aos valores permitidos na política.

A seguinte política permite que os usuários criem clusters de banco de dados e apliquem qualquer tag durante a criação, exceto a tag `environment=prod`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:AddTagsToResource"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/environment": "prod"
                }
            }
        }
    ]
}
```

Ações de API do RDS que permitem atribuir tags durante a criação

As ações de API do RDS a seguir permitem atribuir tags durante a criação de um recurso. Para estas ações, você pode atribuir tags ao criar o recurso:

- `CreateBlueGreenDeployment`
- `CreateCustomDBEngineVersion`
- `CreateDBCluster`
- `CreateDBClusterEndpoint`
- `CreateDBClusterParameterGroup`
- `CreateDBClusterSnapshot`
- `CreateDBInstance`
- `CreateDBInstanceReadReplica`
- `CreateDBParameterGroup`
- `CreateDBProxy`
- `CreateDBProxyEndpoint`
- `CreateDBSecurityGroup`
- `CreateDBShardGroup`
- `CreateDBSnapshot`
- `CreateDBSubnetGroup`
- `CreateEventSubscription`
- `CreateGlobalCluster`
- `CreateIntegration`
- `CreateOptionGroup`

- `CreateTenantDatabase`
- `CopyDBClusterParameterGroup`
- `CopyDBClusterSnapshot`
- `CopyDBParameterGroup`
- `CopyDBSnapshot`
- `CopyOptionGroup`
- `RestoreDBClusterFromS3`
- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceFromS3`
- `RestoreDBInstanceToPointInTime`
- `PurchaseReservedDBInstancesOffering`

Se você usar a AWS CLI ou a API para criar um recurso com tags, o parâmetro `Tags` será usado para aplicar tags aos recursos durante a criação.

Para essas ações de API, se a atribuição de tags falhar, o recurso não será criado e a solicitação falhará e apresentará um erro. Isso garante que os recursos sejam criados com tags ou não sejam criados, o que evita a criação de recursos sem as tags pretendidas.

AWSPolíticas gerenciadas pela para o Amazon RDS

Para adicionar permissões a conjuntos de permissões e perfis, é mais fácil usar políticas gerenciadas pela AWS do que elaborar políticas por conta própria. É necessário tempo e experiência para criar políticas gerenciadas pelo cliente do IAM que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, você pode usar nossas políticas gerenciadas pela AWS. Essas políticas abrangem casos de uso comuns e estão disponíveis na sua Conta da AWS. Para ter mais informações sobre as políticas gerenciadas da AWS, consulte Políticas gerenciadas da AWS no Guia do usuário do IAM.

Os Serviços da AWS mantêm e atualizam políticas gerenciadas pela AWS. Não é possível alterar as permissões em políticas gerenciadas pela AWS. Os serviços ocasionalmente acrescentam permissões adicionais a uma política gerenciada pela AWS para oferecer compatibilidade com novos atributos. Esse tipo de atualização afeta todas as identidades (conjuntos de permissões e perfis) às quais a política está anexada. É mais provável que os serviços atualizem uma política gerenciada pela AWS quando um novo atributo for iniciado ou novas operações se tornarem disponíveis. Os serviços não removem permissões de uma política gerenciada pela AWS. Portanto, as atualizações de políticas não suspendem suas permissões atuais.

Além disso, a AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política `ReadOnlyAccess` gerenciada pela AWS concede acesso somente leitura a todos os recursos e Serviços da AWS. Quando um serviço executa um novo recurso, a AWS adiciona permissões somente leitura para novas operações e recursos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.

Tópicos

- AWSPolítica gerenciada pela : AmazonRDSReadOnlyAccess
- AWSPolítica gerenciada pela : AmazonRDSFullAccess
- AWSPolítica gerenciada pela : AmazonRDSDataFullAccess
- AWSPolítica gerenciada pela : AmazonRDSEnhancedMonitoringRole
- AWSPolítica gerenciada pela : AmazonRDSPerformanceInsightsReadOnly
- Política gerenciada pela AWS: AmazonRDSPerformanceInsightsFullAccess
- AWSPolítica gerenciada pela : AmazonRDSDirectoryServiceAccess
- AWSPolítica gerenciada pela : AmazonRDSServiceRolePolicy
- Política gerenciada da AWS: AmazonRDSPreviewServiceRolePolicy

- Política gerenciada da AWS: AmazonRDSBetaServiceRolePolicy

AWSPolítica gerenciada pela : AmazonRDSReadOnlyAccess

Essa política permite o acesso somente leitura ao Amazon RDS por meio do Console de gerenciamento da AWS.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `rds`: permite que as entidades principais descrevam os recursos do Amazon RDS e listem as etiquetas dos respectivos recursos.
- `cloudwatch`: permite que as entidades principais obtenham estatísticas de métricas do Amazon CloudWatch.
- `ec2`: permite que as entidades principais descrevam zonas de disponibilidade e recursos de rede.
- `logs`: permite que as entidades principais descrevam fluxos de log do CloudWatch Logs de grupos de logs e obtenham eventos de log do CloudWatch Logs.
- `devops-guru`: permite que as entidades principais descrevam os recursos que têm cobertura do Amazon DevOps Guru, que é especificada pelos nomes das pilhas ou pelas tags de recursos do CloudFormation.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSReadOnlyAccess no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSFullAccess

Essa política concede acesso total ao Amazon RDS por meio do Console de gerenciamento da AWS.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `rds`: concede às entidades principais acesso total ao Amazon RDS.
- `application-autoscaling`: permite que as entidades principais descrevam e gerenciem destinos e políticas de escalabilidade do Application Auto Scaling.

- `cloudwatch`: permite que as entidades principais obtenham estáticas de métricas do CloudWatch e gerenciem os respectivos alarmes.
- `ec2`: permite que as entidades principais descrevam zonas de disponibilidade e recursos de rede.
- `logs`: permite que as entidades principais descrevam fluxos de log do CloudWatch Logs de grupos de logs e obtenham eventos de log do CloudWatch Logs.
- `outposts`: permite que as entidades principais obtenham tipos de instância AWS Outposts.
- `pi`: permite que as entidades principais obtenham métricas do Performance Insights.
- `sns`: permite que as entidades principais acessem assinaturas e tópicos do Amazon Simple Notification Service (Amazon SNS) e publiquem mensagens do Amazon SNS.
- `devops-guru`: permite que as entidades principais descrevam os recursos que têm cobertura do Amazon DevOps Guru, que é especificada pelos nomes das pilhas ou pelas tags de recursos do CloudFormation.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSFullAccess no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSDataFullAccess

Essa política permite acesso total ao uso da API DATA e ao editor de consultas em clusters do Aurora Serverless em uma Conta da AWS específica. Essa política permite que a Conta da AWS obtenha o valor de um segredo do AWS Secrets Manager.

É possível anexar a política `AmazonRDSDataFullAccess` às suas identidades do IAM.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `dbqms`: permite que as entidades principais acessem, criem, excluam, descrevam e atualizem consultas. O Database Query Metadata Service (`dbqms`) é um serviço somente interno. Ele fornece suas consultas recentes e salvas para o editor de consultas no Console de gerenciamento da AWS para vários Serviços da AWS, inclusive o Amazon RDS.
- `rds-data`: permite que as entidades principais executem instruções SQL em bancos de dados do Aurora Serverless.
- `secretsmanager` – permite que as entidades principais obtenham o valor de um segredo do AWS Secrets Manager.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSDataFullAccess no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSEnhancedMonitoringRole

Essa política fornece acesso ao Amazon CloudWatch Logs for Amazon RDS Enhanced Monitoring.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `logs`: permite que as entidades principais criem grupos de logs e políticas de retenção do CloudWatch Logs, bem como criem e descrevam fluxos de log do CloudWatch Logs de grupos de logs. Ela também permite que as entidades principais insiram e obtenham eventos de log do CloudWatch Logs.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSEnhancedMonitoringRole no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSPerformanceInsightsReadOnly

Essa política fornece acesso somente leitura ao Insights de Performance do Amazon RDS para instâncias de banco de dados do Amazon RDS e de clusters de banco de dados do Amazon Aurora.

Essa política agora inclui `Sid` (ID da instrução) como identificador para a declaração de política.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `rds`: permite que as entidades principais descrevam instâncias de banco de dados do Amazon RDS e de clusters de banco de dados Amazon Aurora
- `pi`: permite que as entidades principais façam chamadas para a API do Insights de Performance do Amazon RDS e acessem as métricas do Insights de Performance.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSPerformanceInsightsReadOnly no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonRDSPerformanceInsightsFullAccess

Essa política fornece acesso total ao Insights de Performance do Amazon RDS para instâncias de banco de dados do Amazon RDS e clusters de banco de dados do Amazon Aurora.

Essa política agora inclui `Sid` (ID da instrução) como identificador para a declaração de política.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `rds`: permite que as entidades principais descrevam instâncias de banco de dados do Amazon RDS e de clusters de banco de dados Amazon Aurora
- `pi`: permite que as entidades principais façam chamadas para a API do Insights de Performance do Amazon RDS e criem, visualizem e excluam relatórios de análise de performance.
- `cloudwatch`: permite que as entidades principais listem métricas do Amazon CloudWatch e obtenham dados de métricas e estatística.

Para receber mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSPerformanceInsightsFullAccess no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSDirectoryServiceAccess

Essa política permite que o Amazon RDS faça chamadas ao Directory Service.

Detalhes relacionados às permissões

Esta política inclui a seguinte permissão:

- `ds`: permite que as entidades principais descrevam diretórios do Directory Service e controlem a autorização para diretórios do Directory Service.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSDirectoryServiceAccess no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada pela : AmazonRDSServiceRolePolicy

Não é possível anexar a política `AmazonRDSServiceRolePolicy` às suas entidades do IAM. Essa política é anexada a uma função vinculada ao serviço que permite ao Amazon RDS realizar ações

em seu nome. Para ter mais informações, consulte Permissões de função vinculada ao serviço do Amazon Aurora.

Política gerenciada da AWS: AmazonRDSPreviewServiceRolePolicy

Não é possível anexar `AmazonRDSPreviewServiceRolePolicy` às suas entidades do IAM. Essa política é anexada a um perfil vinculado ao serviço que permite ao Amazon RDS chamar serviços da AWS em nome de seus recursos de banco de dados do RDS. Para obter mais informações, consulte Perfil vinculado ao serviço da Pré-visualização do Amazon RDS.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `ec2`: permite que as entidades principais descrevam Zonas de Disponibilidade e recursos de rede.
- `secretsmanager` – permite que as entidades principais obtenham o valor de um segredo do AWS Secrets Manager.
- `cloudwatch`, `logs`: permite que o Amazon RDS faça upload de métricas e logs da instância de banco de dados no CloudWatch por meio do agente do CloudWatch.

Para ter mais informações sobre essa política, bem como o documento de política JSON, consulte AmazonRDSPreviewServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonRDSBetaServiceRolePolicy

Não é possível anexar `AmazonRDSBetaServiceRolePolicy` às suas entidades do IAM. Essa política é anexada a um perfil vinculado ao serviço que permite ao Amazon RDS chamar serviços da AWS em nome de seus recursos de banco de dados do RDS. Para obter mais informações, consulte Permissões de perfis vinculados ao serviço para o Amazon RDS Beta.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `ec2`: permite que o Amazon RDS execute operações de backup na instância de banco de dados que fornece recursos de restauração para um ponto no tempo.
- `secretsmanager`: permite ao Amazon RDS gerenciar segredos específicos da instância de banco de dados criados pelo Amazon RDS.

- `cloudwatch`, `logs`: permite que o Amazon RDS faça upload de métricas e logs da instância de banco de dados no CloudWatch por meio do agente do CloudWatch.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSBetaServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Atualizações do Amazon RDS para políticas gerenciadas pela AWS

Visualize detalhes sobre atualizações em políticas gerenciadas pela AWS para o Amazon RDS desde que esse serviço começou a monitorar essas alterações. Para receber alertas automáticos sobre mudanças nesta página, assine o feed RSS na página Histórico de documentos do Amazon RDS.

Alteração	Descrição	Data
Política gerenciada da AWS: AmazonRDSPreviewServiceRolePolicy: atualizar para uma política existente.	O Amazon RDS removeu a permissão `sns:Publish` da `AmazonRDSPreviewServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDSPreview` . Para ter mais informações, consulte Política gerenciada da AWS: AmazonRDSPreviewServiceRolePolicy.	7 de agosto de 2024
Política gerenciada da AWS: AmazonRDSBetaServiceRolePolicy: atualizar para uma política existente.	O Amazon RDS removeu a permissão `sns:Publish` da `AmazonRDSBetaServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDSBeta` . Para ter mais informações, consulte Política gerenciada da AWS: AmazonRDSBetaServiceRolePolicy.	7 de agosto de 2024
AWSPolítica gerenciada pela : AmazonRDSServiceRolePolicy: atualizar para uma política existente.	O Amazon RDS removeu a permissão `sns:Publish` da `AmazonRDSServiceRolePolicy`	2 de julho de 2024

Alteração	Descrição	Data
	do perfil vinculado ao serviço `AWSServiceRoleForRDS` . Para ter mais informações, consulte AWSPolítica gerenciada pela : AmazonRDSServiceRolePolicy.	
AWSPolíticas gerenciadas pela para o Amazon RDS: atualizar para uma política existente.	O Amazon RDS adicionou uma nova permissão ao `AmazonRDSCustomServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDSCustom` para permitir que o RDS Custom para SQL Server modifique o tipo de instância de host do banco de dados subjacente. O RDS também adicionou a permissão `ec2:DescribeInstanceTypes` para receber informações sobre o tipo de instância para o host do banco de dados. Para ter mais informações, consulte AWSPolíticas gerenciadas pela para o Amazon RDS.	8 de abril de 2024

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS – Nova política	O Amazon RDS adicionou uma nova política gerenciada chamada `AmazonRDSCustomInstanceProfileRolePolicy` para permitir que o RDS Custom execute ações de automação e tarefas de gerenciamento de banco de dados por meio de um perfil de instância do EC2. Para ter mais informações, consulte AWSPolíticas gerenciadas pela para o Amazon RDS.	27 de fevereiro de 2024
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou novos IDs de declarações à `AmazonRDSServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDS` . Para ter mais informações, consulte Permissões de função vinculada ao serviço do Amazon Aurora.	19 de janeiro de 2024

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS: atualizações em políticas existentes.	As políticas `AmazonRDSPerformanceInsightsFullAccess` gerenciadas `AmazonRDSPerformanceInsightsReadOnly` e as políticas agora incluem `Sid` (ID da instrução) como identificador na declaração de política. Para obter mais informações, consulte AWSPolítica gerenciada pela : AmazonRDSPerformanceInsightsReadOnly e Política gerenciada pela AWS: AmazonRDSPerformanceInsightsFullAccess.	23 de outubro de 2023
AWSPolíticas gerenciadas pela para o Amazon RDS: atualizar para uma política existente.	O Amazon RDS adicionou novas permissões à política gerenciada `AmazonRDSFullAccess`. As permissões autorizam que você gere, visualize e exclua o relatório de análise de performance por um período. Para receber mais informações sobre como configurar políticas de acesso para o Insights de Performance, consulte Configurar políticas de acesso para o Performance Insights.	17 de agosto de 2023

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS: nova política e atualização da política existente.	O Amazon RDS adicionou novas permissões à política gerenciada `AmazonRDSPerformanceInsightsReadOnly` e uma nova política gerenciada chamada `AmazonRDSPerformanceInsightsFullAccess`. Essas permissões autorizam que você analise o Insights de Performance por um período, visualize os resultados da análise com as recomendações e exclua os relatórios. Para receber mais informações sobre como configurar políticas de acesso para o Insights de Performance, consulte Configurar políticas de acesso para o Performance Insights.	16 de agosto de 2023

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS: atualizar para uma política existente	O Amazon RDS adicionou um novo namespace do Amazon CloudWatch `ListMetrics` a `AmazonRDSFullAccess` e a `AmazonRDSReadOnlyAccess` . Esse namespace é necessário para que o Amazon RDS liste métricas específicas de uso de recursos. Para ter mais informações, consulte Visão geral do gerenciamento de permissões de acesso aos recursos do CloudWatch no Guia do usuário do Amazon CloudWatch.	4 de abril de 2023

Alteração	Descrição	Data
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou novas permissões à `AmazonRDSServiceRolePolicy` da função vinculada ao serviço `AWSServiceRoleForRDS` para integração com o AWS Secrets Manager. O RDS requer integração com o Secrets Manager para gerenciar senhas do usuário principal no Secrets Manager. O segredo usa uma convenção de nomenclatura reservada e restringe as atualizações do cliente. Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.	22 de dezembro de 2022

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS: atualizações em políticas existentes.	O Amazon RDS adicionou uma nova permissão a `AmazonRDSFullAccess` e às políticas gerenciadas `AmazonRDSReadOnlyAccess` para possibilitar que você ative o Amazon DevOps Guru no console do RDS. Essa permissão é necessária para conferir se o DevOps Guru está ativado. Para ter mais informações, consulte Configurar políticas de acesso do IAM para DevOps Guru para RDS.	19 de dezembro de 2022
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou um novo namespace do Amazon CloudWatch ao `AmazonRDSPreviewServiceRolePolicy` para `PutMetricData`. Esse namespace é necessário para que o Amazon RDS publique métricas de uso de recursos. Para ter mais informações, consulte Usar chaves de condição para limitar o acesso a namespaces do CloudWatch no Guia do usuário do Amazon CloudWatch.	7 de junho de 2022

Alteração	Descrição	Data
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou um novo namespace do Amazon CloudWatch ao `AmazonRDSBetaServiceRolePolicy` para `PutMetricData`. Esse namespace é necessário para que o Amazon RDS publique métricas de uso de recursos. Para ter mais informações, consulte Usar chaves de condição para limitar o acesso a namespaces do CloudWatch no Guia do usuário do Amazon CloudWatch.	7 de junho de 2022
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou um novo namespace do Amazon CloudWatch ao `AWSServiceRoleForRDS` para `PutMetricData`. Esse namespace é necessário para que o Amazon RDS publique métricas de uso de recursos. Para ter mais informações, consulte Usar chaves de condição para limitar o acesso a namespaces do CloudWatch no Guia do usuário do Amazon CloudWatch.	22 de abril de 2022

Alteração	Descrição	Data
AWSPolíticas gerenciadas pela para o Amazon RDS – Nova política	O Amazon RDS adicionou uma nova política gerenciada denominada `AmazonRDSPerformanceInsightsReadOnly`, para permitir que o Amazon RDS chame serviços da AWS em nome de suas instâncias de banco de dados. Para receber mais informações sobre como configurar políticas de acesso para o Insights de Performance, consulte Configurar políticas de acesso para o Performance Insights.	10 de março de 2022

Alteração	Descrição	Data
Permissões de função vinculada ao serviço do Amazon Aurora: atualizar para uma política existente	O Amazon RDS adicionou novos namespaces do Amazon CloudWatch ao `AWSServiceRoleForRDS` para `PutMetricData` . Esses namespaces são necessários para que o Amazon DocumentDB (compatível com MongoDB) e o Amazon Neptune publiquem métricas do CloudWatch. Para ter mais informações, consulte Usar chaves de condição para limitar o acesso a namespaces do CloudWatch no Guia do usuário do Amazon CloudWatch.	4 de março de 2022
O Amazon RDS passou a monitorar alterações	O Amazon RDS passou a monitorar alterações nas políticas gerenciadas pela AWS.	26 de outubro de 2021

Prevenção do problema do substituto confuso entre serviços

O problema “confused deputy” é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Na AWS, a personificação entre serviços pode resultar no problema do ‘confused deputy’.

A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta. Para obter mais informações, consulte O problema confused deputy no Guia do usuário IAM.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas de recursos para limitar as permissões que o Amazon RDS concede a outro serviço para um recurso específico.

Em alguns casos, o valor `aws:SourceArn` não contém o ID da conta, por exemplo, quando você usa o nome do recurso da Amazon (ARN) para um bucket do Simple Storage Service (Amazon S3). Nesses casos, certifique-se de usar as duas chaves de contexto de condição global para limitar as permissões. Em alguns casos, você usa chaves de contexto de condição global e o valor `aws:SourceArn` contém o ID da conta. Nesses casos, verifique se o valor `aws:SourceAccount` e a conta no `aws:SourceArn` usa o mesmo ID de conta quando eles são usados na mesma instrução de política. Use se quiser que apenas um recurso seja associado ao acesso entre serviços `aws:SourceArn`. Use `aws:SourceAccount` se você quiser permitir que qualquer recurso nessa conta da AWS específica seja associado ao uso entre serviços.

Verifique se o valor de `aws:SourceArn` é um ARN para um tipo de recurso do Amazon RDS. Para obter mais informações, consulte Nomes de recurso da Amazon (ARNs) no Amazon RDS.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Em alguns casos, talvez você não saiba o ARN completo do recurso ou pode estar especificando vários recursos. Nesses casos, use a chave de condição de contexto global com curingas `aws:SourceArn` (`*`) para as partes desconhecidas do ARN. Um exemplo é `arn:aws:rds:*:123456789012:*`.

O exemplo a seguir mostra como é possível usar as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` no Amazon RDS, a fim de evitar o problema do substituto confuso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Para obter mais exemplos de políticas que usam as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount`, consulte as seguintes seções:

- Conceder permissões para publicar notificações em um tópico do Amazon SNS
- Configurar o acesso a um bucket do Amazon S3 (Importação do PostgreSQL)
- Configurar o acesso a um bucket do Amazon S3 (Exportação do PostgreSQL)

Autenticação do banco de dados do IAM

Você pode se autenticar o cluster de banco de dados usando a autenticação de banco de dados do AWS Identity and Access Management (IAM). A autenticação do banco de dados do IAM funciona com o Aurora MySQL e o Aurora PostgreSQL. Com esse método de autenticação, você não precisa usar uma senha ao conectar-se a um de instância de banco de dados. Em vez disso, você usa um token de autenticação.

Um token de autenticação é uma string exclusiva de caracteres que o Amazon Aurora gera mediante solicitação. Os tokens de autenticação são gerados usando o Signature da AWS versão 4. Cada token tem uma vida útil de 15 minutos. Você não precisa armazenar as credenciais de usuário no banco de dados, porque a autenticação é gerenciada externamente usando o IAM. Você também pode usar a autenticação de banco de dados padrão. O token é usado apenas para autenticação e não afetará a sessão depois que ela for estabelecida.

A autenticação do banco de dados do IAM oferece os seguintes benefícios:

- O tráfego de rede de e para o banco de dados é criptografado usando Secure Socket Layer (SSL) ou Transport Layer Security (TLS). Para obter mais informações sobre como usar SSL/TLS com o Amazon Aurora, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.
- Você pode usar o IAM para gerenciar centralmente o acesso aos recursos de banco de dados, em vez de gerenciar o acesso individualmente em cada cluster de banco de dados.
- Para aplicações em execução no Amazon EC2, você pode usar as credenciais específicas da instância do EC2 para acessar o banco de dados em vez de uma senha para maior segurança.

Em geral, considere usar a autenticação de banco de dados do IAM quando suas aplicações criam menos de 200 conexões por segundo e você não deseja gerenciar nomes de usuário e senhas diretamente no código da aplicação.

O driver JDBC da Amazon Web Services (AWS) comporta a autenticação do banco de dados do IAM. Para ter mais informações, consulte AWS IAM Authentication Plugin no Amazon Web Services (AWS) JDBC Driver GitHub repository.

O driver Python da Amazon Web Services (AWS) comporta a autenticação do banco de dados do IAM. Para ter mais informações, consulte AWS IAM Authentication Plugin no Amazon Web Services (AWS) Python Driver GitHub repository.

Navegue pelos tópicos a seguir para aprender o processo de configuração do IAM para autenticação de banco de dados:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM
- Conectar-se ao cluster de banco de dados usando a autenticação do IAM

Disponibilidade de regiões e versões

A disponibilidade e a compatibilidade de recursos variam entre versões específicas de cada mecanismo de banco de dados do Aurora e entre Regiões da AWS. Para obter mais informações sobre a disponibilidade de versões e regiões com Aurora e autenticação de banco de dados do IAM, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a autenticação de banco de dados do IAM.

Para o Aurora MySQL, todas as classes de instância de banco de dados compatíveis oferecem suporte à autenticação de banco de dados do IAM, exceto db.t2.small e db.t3.small. Para obter informações sobre as classes de instância de banco de dados compatíveis, consulte Mecanismos de banco de dados compatíveis para classes de instância de banco de dados.

Suporte para CLI e SDK

A autenticação de banco de dados do IAM está disponível para a AWS CLI e para os seguintes AWS SDKs específicos à linguagem:

- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWS SDK para Java
- AWS SDK para JavaScript
- AWS SDK para PHP
- AWS SDK para Python (Boto3)
- AWS SDK para Ruby

Limitações para a autenticação de banco de dados do IAM

Ao usar a autenticação do banco de dados do IAM, as seguintes limitações se aplicam:

- Atualmente, a autenticação do banco de dados do IAM não oferece suporte a todas as chaves de contexto de condição global.

  Para obter mais informações sobre chaves de contexto de condição global, consulte Chaves de contexto de condição global AWS no Guia do usuário do IAM.
- No PostgreSQL, se o perfil do IAM (`rds_iam`) for adicionado a um usuário (por exemplo, o usuário principal do RDS), a autenticação do IAM terá precedência sobre a autenticação por senha, então o usuário precisará fazer login como um usuário do IAM.
- Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.
- O CloudWatch e o CloudTrail não registram em log a autenticação do IAM. Esses serviços não rastreiam chamadas de API `generate-db-auth-token` que autorizam o perfil do IAM a habilitar a conexão com o banco de dados.
- A autenticação de banco de dados do IAM requer recursos de computação no cluster de banco de dados. É necessário ter entre 300 MiB e 1.000 MiB extras de memória no banco de dados para ter uma conectividade confiável. Para ver a memória necessária para a workload, compare a coluna RES para processos do RDS na lista de processos do monitoramento avançado antes e depois de habilitar a autenticação de banco de dados do IAM. Consulte Como visualizar métricas do SO no console do RDS.

  Se você estiver usando uma instância de classe com capacidade de expansão, evite ficar sem memória reduzindo a memória usada por outros parâmetros, como buffers e cache, na mesma quantidade.
- Em relação ao Aurora MySQL, não é possível usar autenticação baseada em senha para um usuário do banco de dados configurado com a autenticação do IAM.
- Não é possível usar a autenticação de banco de dados do IAM com o RDS no Outposts em nenhum mecanismo.

Recomendações para autenticação de banco de dados do IAM

Recomendamos o seguinte durante o uso da autenticação do banco de dados do IAM:

- Use a autenticação de banco de dados do IAM quando sua aplicação exigir menos de 200 novas conexões de autenticação de banco de dados do IAM por segundo.

  Os mecanismos de banco de dados que funcionam com o Amazon Aurora não impõem quaisquer limites para as tentativas de autenticação por segundo. No entanto, quando você usa a autenticação de banco de dados do IAM, sua aplicação deve gerar um token de autenticação. Sua aplicação então usa esse token para se conectar ao cluster de banco de dados. Se você exceder o limite de novas conexões máximas por segundo, a sobrecarga extra da autenticação de banco de dados IAM poderá causar a limitação da conexão.

  Considere usar o agrupamento de conexões em suas aplicações para mitigar a criação constante de conexões. Isso pode reduzir a sobrecarga da autenticação de banco de dados do IAM e permitir que as aplicações reutilizem as conexões existentes. Como alternativa, considere usar o RDS Proxy para esses casos. O RDS Proxy tem custos adicionais. Consulte Preços do RDS Proxy.
- O tamanho de um token de autenticação de banco de dados do IAM depende de muitas coisas, incluindo o número de etiquetas do IAM, políticas de serviço do IAM, comprimentos de ARN, bem como outras propriedades do IAM e do banco de dados. O tamanho mínimo desse token geralmente é de cerca de 1 KB, mas pode ser maior. Como esse token é usado como senha na string de conexão com o banco de dados por meio da autenticação do IAM, você deve garantir que o driver de banco de dados (por exemplo, ODBC) e/ou quaisquer ferramentas não limitem nem trunquem esse token devido ao respectivo tamanho. Um token truncado fará com que a validação da autenticação feita pelo banco de dados e pelo IAM falhe.
- Se você estiver usando credenciais temporárias ao criar um token de autenticação do banco de dados do IAM, as credenciais temporárias ainda deverão ser válidas ao usar o token de autenticação do banco de dados do IAM para fazer uma solicitação de conexão.

Chaves de contexto de condição globais da AWS incompatíveis

A autenticação do banco de dados do IAM não é compatível com o seguinte subconjunto de chaves de contexto de condição global da AWS.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`

- `aws:VpcSourceIp`

Para obter mais informações, consulte Chaves de contexto de condição global da AWS no Guia do usuário do IAM.

Habilitar e desabilitar a autenticação de banco de dados do IAM

Por padrão, a autenticação de banco de dados do IAM está desabilitada nos clusters de banco de dados. É possível habilitar ou desabilitar a autenticação de banco de dados do IAM usando o Console de gerenciamento da AWS, a AWS CLI ou a API.

É possível habilitar a autenticação de banco de dados do IAM ao executar uma das seguintes ações:

- Para criar um novo cluster de banco de dados com autenticação de banco de dados do IAM habilitada, consulte Criar um cluster de bancos de dados do Amazon Aurora.
- Para modificar um cluster de banco de dados para habilitar a autenticação de banco de dados do IAM, consulte Modificar um cluster de bancos de dados Amazon Aurora.
- Para restaurar um cluster de banco de dados de um snapshot com a autenticação de banco de dados do IAM habilitada, consulte Restauração de um snapshot de um cluster de banco de dados.
- Para restaurar um cluster de banco de dados em um momento específico com a autenticação de banco de dados do IAM habilitada, consulte Restaurar um cluster de banco de dados para um horário especificado.

Console

Cada fluxo de trabalho de criação ou modificação tem uma seção Database authentication (Autenticação de banco de dados), onde é possível habilitar ou desabilitar a autenticação de banco de dados do IAM. Nessa seção, escolha Password and IAM database authentication (Senha e autenticação do banco de dados do IAM) para habilitar a autenticação do banco de dados do IAM.

Para habilitar ou desabilitar a autenticação do banco de dados do IAM para um cluster de banco de dados existente

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha o cluster de banco de dados que você deseja modificar.

> **Note**
>
> Só será possível habilitar a autenticação do IAM se todas as instâncias de banco de dados no cluster de banco de dados forem compatíveis com o IAM. Verifique os requisitos de compatibilidade em Disponibilidade de regiões e versões.

4. Selecione Modify.
5. Na seção Autenticação do banco de dados, escolha Autenticação do banco de dados do IAM para habilitar a autenticação do banco de dados do IAM. Escolha Autenticação de senha ou Senha e autenticação Kerberos para desabilitar a autenticação do IAM.
6. Você também pode optar por habilitar a publicação de logs de autenticação de banco de dados do IAM no CloudWatch Logs. Em Exportações de log, escolha a opção Log iam-db-auth-error. A publicação de logs no CloudWatch Logs consome armazenamento e gera cobranças por esse armazenamento. Se não precisar mais de algum log do CloudWatch, lembre-se de excluí-lo.
7. Escolha Continue.
8. Para aplicar as alterações imediatamente, escolha Immediately (Imediatamente) na seção Scheduling of modifications (Programação de modificações).
9. Escolha Modify cluster (Modificar cluster) para salvar suas alterações.

AWS CLI

Para criar um novo cluster de banco de dados com a autenticação do IAM usando a AWS CLI, use o comando `create-db-cluster`. Especifique a opção `--enable-iam-database-authentication`.

Para atualizar um cluster de banco de dados existente a fim de ter ou não autenticação do IAM, use o comando da AWS CLI `modify-db-cluster`. Especifique a opção `--enable-iam-database-authentication` ou `--no-enable-iam-database-authentication`, conforme apropriado.

> **Note**
>
> Só será possível habilitar a autenticação do IAM se todas as instâncias de banco de dados no cluster de banco de dados forem compatíveis com o IAM. Verifique os requisitos de compatibilidade em Disponibilidade de regiões e versões.

Por padrão, o Aurora modifica a instância de banco de dados durante a próxima janela de manutenção. Se você quiser habilitar a autenticação de banco de dados do IAM o mais rápido possível, use o parâmetro `--apply-immediately`.

Se você está restaurando um cluster de banco de dados, use um dos comandos da AWS CLI a seguir:

- `restore-db-cluster-to-point-in-time`
- `restore-db-cluster-from-db-snapshot`

A autenticação de banco de dados do IAM assumirá como padrão aquela do snapshot de origem. Para alterar essa configuração, defina a opção `--enable-iam-database-authentication` ou `--no-enable-iam-database-authentication`, conforme apropriado.

API do RDS

Para criar uma nova instância de banco de dados com a autenticação do IAM usando a API, use a operação da API `CreateDBCluster`. Defina o parâmetro `EnableIAMDatabaseAuthentication` como `true`.

Para atualizar um cluster de banco de dados existente a fim de ter ou não autenticação do IAM, use a operação da API `ModifyDBCluster`. Defina o parâmetro `EnableIAMDatabaseAuthentication` como `true` para habilitar a autenticação do IAM, ou `false` para desabilitá-la.

> **Note**
>
> Só será possível habilitar a autenticação do IAM se todas as instâncias de banco de dados no cluster de banco de dados forem compatíveis com o IAM. Verifique os requisitos de compatibilidade em Disponibilidade de regiões e versões.

Se você está restaurando um de instância de banco de dados, use uma das operações da API a seguir:

- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`

A autenticação de banco de dados do IAM assumirá como padrão aquela do snapshot de origem. Para alterar essa configuração, defina o parâmetro `EnableIAMDatabaseAuthentication` como `true` para habilitar a autenticação do IAM, ou `false` para desabilitá-la.

Criar e usar uma política do IAM para acesso do banco de dados do IAM

Para permitir que um usuário ou um perfil se conecte ao cluster de banco de dados, você deve criar uma política do IAM. Depois disso, associe a política a um conjunto de permissões ou a um perfil.

> **Note**
>
> Para saber mais sobre as políticas do IAM, consulte Gerenciamento de identidade e acesso no Amazon Aurora.

O exemplo de política a seguir permite que um usuário se conecte a um cluster de banco de dados usando a autenticação de banco de dados do IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:cluster-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

Important

Um usuário com permissões de administrador pode acessar clusters de banco de dados sem permissões explícitas em uma política do IAM. Se você quiser restringir o acesso do administrador a clusters de banco de dados, é possível criar um perfil do IAM com as permissões adequadas e menos privilegiadas e atribuí-lo ao administrador.

Note

Não confunda o prefixo `rds-db:` com outros prefixos de operações da API do RDS que começam com `rds:`. Você usa o prefixo `rds-db:` e a ação `rds-db:connect` somente para a autenticação de banco de dados do IAM. Eles não são válidos em nenhum outro contexto.

Os exemplos de política incluem uma única instrução com os seguintes elementos:

- `Effect`: especifica `Allow` para conceder acesso ao cluster de banco de dados. Se você não permitir explicitamente o acesso, o acesso será negado por padrão.
- `Action`: especifica `rds-db:connect` para permitir conexões com o cluster de banco de dados.
- `Resource`: especifica um nome do recurso da Amazon (ARN) que descreva uma conta de banco de dados em um cluster. de banco de dados. O formato do ARN é o seguinte.

```
arn:aws:rds-db:region:account-id:dbuser:DbClusterResourceId/db-user-name
```

Neste formato, substitua o seguinte:

- *`region`* é a região da AWS para a de Bancos de Dados cluster. No exemplo de política, a região da AWS é `us-east-2`.
- *`account-id`* é o número da conta da AWS para de Bancos de Dadoscluster. No exemplo de política, o número da conta é `1234567890`. O usuário deve estar na mesma conta que a conta do cluster de banco de dados.

  Para realizar o acesso entre contas, crie um perfil do IAM com a política mostrada acima na conta do cluster de banco de dados e permita que sua outra conta assuma o perfil.

- *DbClusterResourceId* é o identificador do cluster de banco de dados. Esse identificador é exclusivo para uma região da AWS e nunca muda. Na exemplo de política, o identificador é `cluster-ABCDEFGHIJKL01234`.

  Para encontrar um ID de recurso de cluster de banco de dados no Console de gerenciamento da AWS do Amazon Aurora, escolha o cluster de banco de dados para ver os respectivos detalhes. Em seguida, escolha a guia Configuration (Configuração). O Resource ID (ID de recurso) é exibido na seção Configuration (Configuração).

  Como alternativa, use o comando da AWS CLI para listar os identificadores e os IDs de recurso de de Bancos de Dados cluster na região atual da AWS, conforme mostrado a seguir.

  ```
  aws rds describe-db-clusters --query "DBClusters[*].
  [DBClusterIdentifier,DbClusterResourceId]"
  ```

  > **Note**
  >
  > Se você estiver se conectando a um banco de dados por meio do RDS Proxy, especifique o ID do recurso de proxy, como `prx-ABCDEFGHIJKL01234`. Para obter informações sobre como usar a autenticação de banco de dados do IAM com RDS Proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

- *db-user-name* é o nome da conta de banco de dados para associar à autenticação do IAM. No exemplo de política, a conta de banco de dados é `db_user`.

Você pode criar outros ARNs que sejam compatíveis com vários padrões de acesso. A política a seguir permite o acesso a duas contas de banco de dados diferentes em um cluster de banco de dados.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
```

```
        "Action": [
            "rds-db:connect"
        ],
        "Resource": [
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/jane_doe",
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/mary_roe"
        ]
      }
   ]
}
```

A política a seguir usa o caractere "*" para comparar de Bancos de Dados clusters e todas as contas de banco de dados para uma conta daAWS e uma região específicas da AWS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:*/*"
            ]
        }
    ]
}
```

A política a seguir compara de Bancos de Dados clusters de uma conta da AWS e uma região da AWS específicas. Contudo, a política concede acesso somente aos clusters de banco de dados que têm uma conta de banco de dados `jane_doe`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

O usuário ou perfil tem acesso apenas aos bancos de dados que o usuário do banco de dados tem. Por exemplo, suponha que seu cluster de banco de dados tenha um banco de dados chamado dev e outro chamado test. Se a usuária do banco de dados `jane_doe` tiver acesso apenas a dev, os usuários ou perfis que acessarem esse cluster de banco de dados com a usuária `jane_doe` também terão acesso apenas a dev. Essa restrição de acesso também é válida para outros objetos de banco de dados, como tabelas, visualizações e assim por diante.

Um administrador deve criar políticas do IAM que concedam às entidades permissões para executar operações de API específicas nos recursos especificados de que precisam. Depois, o administrador deve anexar essas políticas aos conjuntos de permissões e perfis que exigem essas permissões. Para obter exemplos de políticas, consulte Exemplos de políticas baseadas em identidade do Amazon Aurora.

Anexar uma política do IAM a um conjunto de permissões ou perfil

Depois de criar uma política do IAM para permitir a autenticação de banco de dados, você precisa anexar a política a um conjunto de permissões ou a um perfil. Para obter um tutorial sobre esse tópico, consulte Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

À medida que avança pelo tutorial, você pode usar um dos exemplos de política mostrados nessa seção como um ponto de partida e adequá-lo às suas necessidades. No fim do tutorial, você terá um conjunto de permissões com uma política anexada que pode usar a ação `rds-db:connect`.

> **Note**
>
> Você pode mapear vários conjuntos de permissões ou perfis para a mesma conta de usuário do banco de dados. Por exemplo, suponha que a sua política do IAM especificou o seguinte recurso do ARN.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Se você anexar a política a Jane, Bob e Diego, todos eles poderão se conectar ao cluster de banco de dados em questão usando a conta de banco de dados `jane_doe`.

Criar uma conta de banco de dados usando autenticação do IAM

Com a autenticação de banco de dados do IAM, você não precisa atribuir senhas de banco de dados à conta de usuário que você criar. Se você remover um usuário que está mapeado a uma conta de banco de dados, também deverá remover a conta de banco de dados com a instrução `DROP USER`.

> **Note**
>
> O nome de usuário usado para autenticação do IAM deve corresponder ao caso do nome de usuário no banco de dados.

Tópicos

- Usar autenticação do IAM com o Aurora MySQL
- Usar a autenticação do IAM com o AuroraPostgreSQL

Usar autenticação do IAM com o Aurora MySQL

Com o Aurora MySQL, a autenticação é processada por `AWSAuthenticationPlugin`: um plug-in fornecido pela AWS que funciona perfeitamente com o IAM para autenticar seus usuários. Conecte-se ao cluster de banco de dados como usuário principal ou um usuário diferente que possa criar usuários e conceder privilégios. Depois de se conectar, emita a instrução `CREATE USER` conforme mostrado no exemplo a seguir.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

A cláusula `IDENTIFIED WITH` permite que o Aurora MySQL usem o `AWSAuthenticationPlugin` para autenticar a conta de banco de dados (`jane_doe`). A `AS 'RDS'` cláusula refere-se ao método de autenticação. Verifique se o nome do usuário do banco de dados especificado é o mesmo que um recurso na política do IAM para acesso ao banco de dados do IAM. Para obter mais informações, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM.

> **Note**
>
> Se você ver a mensagem a seguir, significa que o plugin fornecido pela AWS não está disponível para de Bancos de Dados atual cluster.
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> Para solucionar esse erro, verifique se você está usando uma configuração compatível e se habilitou a autenticação de banco de dados do IAM em seu cluster de banco de dados. Para obter mais informações, consulte Disponibilidade de regiões e versões e Habilitar e desabilitar a autenticação de banco de dados do IAM.

Apos criar uma conta usando `AWSAuthenticationPlugin`, você a gerencia do mesmo modo que as outras contas de banco de dados. Por exemplo, você pode modificar os privilégios da conta com os atributos `GRANT` e `REVOKE`, ou modificar os vários atributos da conta com a instrução `ALTER USER`.

O tráfego de rede do banco de dados é criptografado utilizando SSL/TLS ao usar o IAM. Para permitir conexões SSL, modifique a conta do usuário com o comando a seguir.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

Usar a autenticação do IAM com o AuroraPostgreSQL

Para usar a autenticação do IAM com o Aurora PostgreSQL, conecte-se ao cluster de banco de dados como usuário principal ou um usuário diferente que possa criar usuários e conceder privilégios. Depois de conectar-se, crie usuários de banco de dados e, depois, conceda a eles a função `rds_iam` conforme mostrado no exemplo a seguir.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

Verifique se o nome do usuário do banco de dados especificado é o mesmo que um recurso na política do IAM para acesso ao banco de dados do IAM. Para obter mais informações, consulte Criar e usar uma política do IAM para acesso do banco de dados do IAM. Você deve conceder o perfil `rds_iam` para usar a autenticação do IAM. Também é possível usar associações aninhadas ou concessões indiretas do perfil.

Observe que um usuário de banco de dados PostgreSQL pode usar a autenticação do IAM ou Kerberos, mas não ambas, portanto esse usuário também não pode ter a função `rds_ad`. Isso também se aplica a assinaturas aninhadas. Para obter mais informações, consulte Etapa 7: Criar usuários do PostgreSQL para suas entidades principais do Kerberos.

Conectar-se ao cluster de banco de dados usando a autenticação do IAM

Com a autenticação de banco de dados do IAM, você usa um token de autenticação ao se conectar ao cluster de banco de dados. Um token de autenticação é uma string de caracteres que você usa em vez de uma senha. Depois que você gerar um token de autenticação, ele será válido por 15 minutos antes de expirar. Se você tentar se conectar usando um token expirado, a solicitação de conexão será negada.

Todo token de autenticação deve ser acompanhado por uma assinatura válida, usando o Signature da AWS versão 4. (Para ter mais informações, consulte Processo de assinatura do Signature versão 4 na Referência geral da AWS.) A AWS CLI e um SDK da AWS, como AWS SDK para Java ou AWS SDK para Python (Boto3), podem assinar automaticamente cada token que você criar.

Você pode usar um token de autenticação quando se conectar ao Amazon Aurora de outro serviço da AWS, como o AWS Lambda. Ao usar um token, você não precisa inserir uma senha no seu código. Como alternativa, você pode usar o SDK da AWS para criar e assinar programaticamente um token de autenticação.

Depois que tiver um token de autenticação do IAM assinado, você poderá se conectar a um cluster de bancos de dados Aurora. Veja a seguir como fazer isso usando uma ferramenta de linha de comando ou um SDK da AWS, como o AWS SDK para Java ou AWS SDK para Python (Boto3).

Para ter mais informações, consulte as seguintes postagens no blog:

- Use IAM authentication to connect with SQL Workbench/J to Aurora MySQL or Amazon RDS para MySQL
- Using IAM authentication to connect with pgAdmin Amazon Aurora PostgreSQL or Amazon RDS para PostgreSQL

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

Tópicos

- Conectar-se ao cluster de banco de dados usando a autenticação do IAM com os drivers da AWS.
- Conectando-se ao seu cluster de banco de dados usando a autenticação do IAM na linha de comando: AWS CLI e cliente mysql
- Conectar o cluster de banco de dados usando a autenticação do IAM na linha de comando: AWS CLI e cliente psql
- Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para .NET
- Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Go
- Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Java
- Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Python (Boto3)

Conectar-se ao cluster de banco de dados usando a autenticação do IAM com os drivers da AWS.

O pacote de drivers da AWS foram projetados para comportar tempos mais rápidos de transição e de failover, além de autenticação com o AWS Secrets Manager, o AWS Identity and Access

Management (IAM) e identidades federadas. Os drivers da AWS dependem do monitoramento do status do cluster de banco de dados e do conhecimento da topologia do cluster para determinar o novo gravador. Essa abordagem reduz os tempos de transição e de failover para segundos de um dígito, em comparação com dezenas de segundos para drivers de código aberto.

Para ter mais informações sobre os drivers da AWS, consulte o driver de linguagem correspondente para o cluster de banco de dados do Aurora MySQL ou do Aurora PostgreSQL.

Conectando-se ao seu cluster de banco de dados usando a autenticação do IAM na linha de comando: AWS CLI e cliente mysql

Você pode se conectar de uma linha de comando a um cluster de bancos de dados Aurora com a AWS CLI e a ferramenta da linha de comando `mysql`, conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

> **Note**
>
> Para obter informações sobre como se conectar ao banco de dados usando o SQL Workbench/J com autenticação do IAM, consulte a publicação do blog Use IAM authentication to connect with SQL Workbench/J to Aurora MySQL or Amazon RDS para MySQL.

Tópicos

- Gerar um token de autenticação do IAM
- Conexão ao cluster de banco de dados

Gerar um token de autenticação do IAM

O exemplo a seguir mostra como obter um token de autenticação assinado usando a AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

No exemplo, os parâmetros são os seguintes:

- `--hostname`: o nome do host do cluster de banco de dados que você deseja acessar
- `--port`: o número da porta usada para se conectar ao cluster de banco de dados
- `--region`: a região da AWS na qual o cluster do banco de dados está em execução.
- `--username`: a conta de banco de dados que você deseja acessar

Os primeiros caracteres do token são parecidos com os seguintes.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

Conexão ao cluster de banco de dados

O formato geral para se conectar é mostrado a seguir.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

Os parâmetros são os seguintes:

- `--host`: o nome do host do cluster de banco de dados que você deseja acessar
- `--port`: o número da porta usada para se conectar ao cluster de banco de dados
- `--ssl-ca`: o caminho completo para o arquivo de certificado SSL que contém a chave pública

Para ter mais informações, consulte Conexões do TLS com clusters de banco de dados do Aurora MySQL.

Para baixar um certificado SSL, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

- `--enable-cleartext-plugin`: um valor que especifica que o `AWSAuthenticationPlugin` deve ser usado para essa conexão

  Se você estiver usando um cliente MariaDB, a opção`--enable-cleartext-plugin` não será necessária.
- `--user`: a conta de banco de dados que você deseja acessar
- `--password`: um token de autenticação do IAM assinado

Um token de autenticação é composto de várias centenas de caracteres. Ele pode ser incômodo para a linha de comando. Um modo de contornar isso é salvar o token em uma variável de ambiente, e usar essa variável quando você se conectar. O exemplo a seguir mostra um modo de executar essa solução alternativa. No exemplo, *`/sample_dir/`* corresponde ao caminho completo do arquivo de certificado SSL contendo a chave pública.

```
RDSHOST="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

Quando você se conecta usando o `AWSAuthenticationPlugin`, a conexão é protegida usando SSL. Para verificar isso, digite o seguinte no prompt de comando `mysql>`.

```
show status like 'Ssl%';
```

As seguintes linhas na saída mostram mais detalhes.

```
+---------------+-------------+
| Variable_name | Value
                                                                    |
```

```
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                    |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                    |
| ...           | ...
+-----------------------------+
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar o cluster de banco de dados usando a autenticação do IAM na linha de comando: AWS CLI e cliente psql

Conecte-se pela linha de comando a um cluster de bancos de dados Aurora PostgreSQL com a AWS CLI e a ferramenta da linha de comando psql conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

> **Note**
>
> Para obter informações sobre como se conectar ao seu banco de dados usando pgAdmin com autenticação do IAM, consulte a publicação do blog Using IAM authentication to connect with pgAdmin Amazon Aurora PostgreSQL or Amazon RDS para PostgreSQL.

Tópicos

- Gerar um token de autenticação do IAM

- Conectar-se a um cluster do Aurora PostgreSQL

Gerar um token de autenticação do IAM

Um token de autenticação é composto de várias centenas de caracteres. Dessa maneira, ele pode ficar estranho na linha de comando. Um modo de contornar isso é salvar o token em uma variável de ambiente, e usar essa variável quando você se conectar. O exemplo a seguir mostra como usar a AWS CLI para obter um token de autenticação assinado usando o comando `generate-db-auth-token` e armazená-lo em uma variável de ambiente `PGPASSWORD`.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

No exemplo, os parâmetros para o comando `generate-db-auth-token` são os seguintes:

- `--hostname`: o nome do host do cluster (endpoint do cluster) de banco de dados que você deseja acessar
- `--port`: o número da porta usada para se conectar ao cluster de banco de dados
- `--region`: a região da AWS na qual o cluster do banco de dados está em execução.
- `--username`: a conta de banco de dados que você deseja acessar

Os primeiros caracteres do token gerado são parecidos com os seguintes.

```
mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

Conectar-se a um cluster do Aurora PostgreSQL

O formato geral para usar psql na conexão é mostrado a seguir.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

Os parâmetros são os seguintes:

- `host`: o nome do host do cluster (endpoint do cluster) de banco de dados que você deseja acessar
- `port`: o número da porta usada para se conectar ao cluster de banco de dados
- `sslmode`: o modo SSL a ser usado

  Quando você usa `sslmode=verify-full`, a conexão SSL verifica o endpoint do cluster de banco de dados em relação ao endpoint no certificado SSL.
- `sslrootcert`: o caminho completo para o arquivo de certificado SSL que contém a chave pública

  Para ter mais informações, consulte Como proteger dados do Aurora PostgreSQL com SSL/TLS.

  Para baixar um certificado SSL, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.
- `dbname`: o banco de dados que você deseja acessar
- `user`: a conta de banco de dados que você deseja acessar
- `password`: um token de autenticação do IAM assinado

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

O exemplo a seguir mostra o uso do psql para conexão. No exemplo, psql usa a variável de ambiente `RDSHOST` para o host e a variável de ambiente `PGPASSWORD` para o token gerado. Além disso, */sample_dir/* corresponde ao caminho completo do arquivo de certificado SSL contendo a chave pública.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

```
psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para .NET

Você pode se conectar a um cluster de banco de dados Aurora MySQL ou Aurora PostgreSQL com a AWS SDK para .NET, conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

Exemplos

O exemplo de código a seguir mostra como gerar um token de autenticação e usá-lo para se conectar a um cluster de banco de dados.

Para executar esse exemplo de código, você precisa do AWS SDK para .NET, encontrado no site AWS. Os pacotes `AWSSDK.CORE` e `AWSSDK.RDS` são necessários. Para se conectar a um cluster de banco de dados, use o conector de banco de dados .NET para o mecanismo de banco de dados, como MySqlConnector para MariaDB ou MySQL ou Npgsql para PostgreSQL.

Esse código se conecta a um cluster de banco de dados Aurora MySQL. Modifique os valores das seguintes variáveis, conforme necessário:

- `server`: o endpoint do cluster de banco de dados que você deseja acessar
- `user`: a conta de banco de dados que você deseja acessar
- `database`: o banco de dados que você deseja acessar
- `port`: o número da porta usada para se conectar ao cluster de banco de dados
- `SslMode`: o modo SSL a ser usado

Quando você usa `SslMode=Required`, a conexão SSL verifica o endpoint do cluster de banco de dados em relação ao endpoint no certificado SSL.

- `SslCa`: o caminho completo para o certificado SSL do Amazon Aurora

  Para baixar um certificado, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new
 MySqlConnection($"server=mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);

      // Execute a query
```

```
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
  }
}
```

Esse código se conecta a um cluster de banco de dados Aurora PostgreSQL.

Modifique os valores das seguintes variáveis, conforme necessário:

- `Server`: o endpoint do cluster de banco de dados que você deseja acessar
- `User ID`: a conta de banco de dados que você deseja acessar
- `Database`: o banco de dados que você deseja acessar
- `Port`: o número da porta usada para se conectar ao cluster de banco de dados
- `SSL Mode`: o modo SSL a ser usado

  Quando você usa `SSL Mode=Required`, a conexão SSL verifica o endpoint do cluster de banco de dados em relação ao endpoint no certificado SSL.
- `Root Certificate`: o caminho completo para o certificado SSL do Amazon Aurora

  Para baixar um certificado, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

```
using System;
using Npgsql;
```

```
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;User Id=jane_doe;Password={pwd};Database=mydb;SSL
 Mode=Require;Root Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Go

Você pode se conectar a um cluster de banco de dados Aurora MySQL ou Aurora PostgreSQL com a AWS SDK para Go, conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

Exemplos

Para executar esses exemplos de código, você precisa do AWS SDK para Go, encontrado no site AWS.

Modifique os valores das seguintes variáveis, conforme necessário:

- `dbName`: o banco de dados que você deseja acessar
- `dbUser`: a conta de banco de dados que você deseja acessar
- `dbHost`: o endpoint do cluster de banco de dados que você deseja acessar

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

- `dbPort`: o número da porta usada para se conectar ao cluster de banco de dados
- `region`: a região da AWS na qual o cluster do banco de dados está em execução.

Além disso, verifique se as bibliotecas importadas no código de exemplo existem no sistema.

> **Important**
>
> Os exemplos nesta seção usam o seguinte código para fornecer credenciais que acessam um banco de dados a partir de um ambiente local:
> `creds := credentials.NewEnvCredentials()`
> Se estiver acessando um banco de dados de um serviço da AWS, como o Amazon EC2 ou Amazon ECS, você poderá substituir o código pelo seguinte código:
> `sess := session.Must(session.NewSession())`

```
creds := sess.Config.Credentials
```

Se você fizer essa alteração, certifique-se de adicionar a seguinte importação:

```
"github.com/aws/aws-sdk-go/aws/session"
```

Tópicos

- Conectar-se usando a autenticação do IAM e o AWS SDK para Go V2
- Conectar-se usando a autenticação do IAM e o AWS SDK para Go V1.

Conectar-se usando a autenticação do IAM e o AWS SDK para Go V2

Conecte-se a um cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Go V2.

O exemplo de código a seguir mostra como gerar um token de autenticação e usá-lo para se conectar a um cluster de banco de dados.

Esse código se conecta a um cluster de banco de dados Aurora MySQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"
```

```
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Esse código se conecta a um cluster de banco de dados Aurora PostgreSQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
```

```
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar-se usando a autenticação do IAM e o AWS SDK para Go V1.

Conecte-se a uma instância de banco de dados usando a autenticação do IAM e o AWS SDK para Go V1

O exemplo de código a seguir mostra como gerar um token de autenticação e usá-lo para se conectar a um cluster de banco de dados.

Esse código se conecta a um cluster de banco de dados Aurora MySQL.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Esse código se conecta a um cluster de banco de dados Aurora PostgreSQL.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Java

Você pode se conectar a um cluster de banco de dados Aurora MySQL ou Aurora PostgreSQL com a AWS SDK para Java, conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM
- Configurar o AWS SDK for Java

Consulte exemplos de como usar o SDK para Java 2.x em Amazon RDS examples using SDK for Java 2.x. Você também pode usar o AWS Advanced JDBC Wrapper. Para isso, consulte a documentação do AWS Advanced JDBC Wrapper.

Tópicos

- Gerar um token de autenticação do IAM
- Criar manualmente um token de autenticação do IAM
- Conexão ao cluster de banco de dados

Gerar um token de autenticação do IAM

Se estiver escrevendo programas usando o AWS SDK para Java, você pode obter um token de autenticação assinado usando a classe `RdsIamAuthTokenGenerator`. O uso dessa classe exige que você forneça as credenciais da AWS. Para fazer isso, crie uma instância da classe `DefaultAWSCredentialsProviderChain`. `DefaultAWSCredentialsProviderChain` usa a primeira chave de acesso da AWS e a chave secreta encontradas na cadeia de fornecedores de credencial padrão. Para ter mais informações sobre chaves de acesso da AWS, consulte Gerenciar chaves de acesso para usuários.

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

Após criar uma instância de `RdsIamAuthTokenGenerator`, você pode chamar o método `getAuthToken` para obter um token assinado. Forneça a região da AWS, o nome do host, o número da porta e o nome do usuário. O exemplo de código a seguir ilustra como fazer isso.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

   static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());
```

```
     return authToken;
    }

}
```

Criar manualmente um token de autenticação do IAM

No Java, o modo mais fácil de gerar um token de autenticação é usar o `RdsIamAuthTokenGenerator`. Essa classe cria um token de autenticação para você e assina-o usando o Signature da AWS versão 4. Para ter mais informações, consulte Processo de assinatura do Signature versão 4 na Referência geral da AWS.

No entanto, você também pode construir e assinar um token de autenticação manualmente, conforme mostrado no exemplo de código a seguir.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
```

```
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
       DefaultAWSCredentialsProviderChain creds = new
DefaultAWSCredentialsProviderChain();
    String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
    String awsSecretKey = creds.getCredentials().getAWSSecretKey();
       String expiryMinutes = "900";

       System.out.println("Step 1:  Create a canonical request:");
       String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
       System.out.println(canonicalString);
       System.out.println();

       System.out.println("Step 2:  Create a string to sign:");
       String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
       System.out.println(stringToSign);
       System.out.println();

       System.out.println("Step 3:  Calculate the signature:");
       String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
       System.out.println(signature);
       System.out.println();

       System.out.println("Step 4:  Add the signing info to the request");

       System.out.println(appendSignature(signature));
       System.out.println();
```

```
    }

    //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
 should be in format YYYYMMDDTHHMMSSZ
    public static String createCanonicalString(String user, String accessKey, String
 date, String dateTime, String region, String expiryPeriod, String hostName, String
 port) throws Exception {
        canonicalQueryParameters.put("Action", action);
        canonicalQueryParameters.put("DBUser", user);
        canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
        canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
        canonicalQueryParameters.put("X-Amz-Date", dateTime);
        canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
        canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
 canonicalQueryParameters.remove(currentQueryParameter);
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
 currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }
```

```
    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                          SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String
serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                        SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
```

```
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

Conexão ao cluster de banco de dados

O exemplo de código a seguir mostra como gerar um token de autenticação e usá-lo para se conectar a um cluster que esteja executando o Aurora MySQL.

Para executar esse exemplo de código, você precisa do AWS SDK para Java, encontrado no site AWS. Além disso, você precisa do seguinte:

- MySQL Connector/J. Este exemplo de código foi testado com `mysql-connector-java-5.1.33-bin.jar`.
- Um certificado intermediário do Amazon Aurora específico de uma região da AWS. (Para ter mais informações, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.) No tempo de execução, o carregador de classe procura o certificado no mesmo diretório do exemplo de código Java para que possa encontrá-lo.
- Modifique os valores das seguintes variáveis, conforme necessário:
  - `RDS_INSTANCE_HOSTNAME`: o nome do host do cluster de banco de dados que você deseja acessar.

- `RDS_INSTANCE_PORT`: o número da porta usado na conexão com o cluster de banco de dados PostgreSQL.
- `REGION_NAME`: a região da AWS na qual o cluster do banco de dados está em execução.
- `DB_USER`: a conta de banco de dados que você deseja acessar.
- `SSL_CERTIFICATE`: um certificado SSL para o Amazon Aurora que é específico de uma região da AWS.

  Para baixar o certificado para a sua região da AWS, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados. Coloque o certificado do SSL no mesmo diretório que esse arquivo do programa Java para que o carregador de classe possa encontrar o certificado no tempo de execução.

Esse exemplo de código obtém as credenciais da AWS a partir dacadeia de fornecedores de credencial padrão.

> **Note**
>
> Especifique uma senha para `DEFAULT_KEY_STORE_PASSWORD` diferente do prompt mostrado aqui como prática recomendada de segurança.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
```

```
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
```

```
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

.hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
and password:
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
verification during the IAM Database Authentication to
     * the db instance.
     * @return
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
```

```
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
 Exception {
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
 KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Conectar-se ao cluster de banco de dados usando a autenticação do IAM e o AWS SDK para Python (Boto3)

Você pode se conectar a um cluster de banco de dados Aurora MySQL ou Aurora PostgreSQL com a AWS SDK para Python (Boto3), conforme descrito a seguir.

Pré-requisitos

Veja a seguir os pré-requisitos para se conectar ao cluster de banco de dados usando a autenticação do IAM:

- Habilitar e desabilitar a autenticação de banco de dados do IAM
- Criar e usar uma política do IAM para acesso do banco de dados do IAM
- Criar uma conta de banco de dados usando autenticação do IAM

Além disso, verifique se as bibliotecas importadas no código de exemplo existem no sistema.

Exemplos

Os exemplos de código usam perfis para credenciais compartilhadas. Para obter informações sobre a especificação de credenciais, consulte Credenciais na documentação AWS SDK para Python (Boto3).

O exemplo de código a seguir mostra como gerar um token de autenticação e usá-lo para se conectar a um cluster de banco de dados.

Para executar esse exemplo de código, você precisa do AWS SDK para Python (Boto3), encontrado no site AWS.

Modifique os valores das seguintes variáveis, conforme necessário:

- `ENDPOINT`: o endpoint do cluster de banco de dados que você deseja acessar
- `PORT`: o número da porta usada para se conectar ao cluster de banco de dados
- `USER`: a conta de banco de dados que você deseja acessar
- `REGION`: a região da AWS na qual o cluster do banco de dados está em execução.
- `DBNAME`: o banco de dados que você deseja acessar
- `SSLCERTIFICATE`: o caminho completo para o certificado SSL do Amazon Aurora

Para `ssl_ca`, defina um certificado SSL. Para baixar um certificado SSL, consulte Usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.

> **Note**
>
> Não é possível usar um registro DNS personalizado do Route 53 em vez do endpoint do cluster de banco de dados para gerar o token de autenticação.

Esse código se conecta a um cluster de banco de dados Aurora MySQL.

Antes de executar esse código, instale o driver PyMySQL seguindo as instruções no Python Package Index.

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
```

```
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Esse código se conecta a um cluster de banco de dados Aurora PostgreSQL.

Antes de executar esse código, instale `psycopg2`, seguindo as instruções na documentação de Psycopg.

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Se você quiser se conectar a um cluster de banco de dados por meio de um proxy, consulte Conectar-se a um banco de dados usando autenticação do IAM.

Solucionar problemas referentes à autenticação de banco de dados do IAM

A seguir, é possível encontrar ideias de solução de problemas para alguns problemas comuns de autenticação de banco de dados do IAM e informações sobre logs e métricas do CloudWatch para autenticação de banco de dados do IAM.

Exportar logs de erros de autenticação de banco de dados do IAM para o CloudWatch Logs

Os logs de erros de autenticação de banco de dados do IAM são armazenados no host do banco de dados e podem ser exportados para sua conta do CloudWatch Logs. Use os logs e os métodos de correção nesta página para solucionar problemas de autenticação de banco de dados do IAM.

Você pode habilitar as exportações de log para o CloudWatch Logs por meio do console, da AWS CLI e da API do RDS. Para obter instruções sobre o console, consulte Publicação de logs de banco de dados no Amazon CloudWatch Logs.

Para exportar logs de erros de autenticação de banco de dados do IAM para o CloudWatch Logs ao criar um cluster de banco de dados por meio da AWS CLI, use o seguinte comando:

```
aws rds create-db-cluster --db-cluster-identifier mydbinstance \
--region us-east-1 \
--engine postgres \
--engine-version 16 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

Para exportar logs de erros de autenticação de banco de dados do IAM para o CloudWatch Logs ao modificar um cluster de banco de dados por meio da AWS CLI, use o seguinte comando:

```
aws rds modify-db-cluster --db-instance-identifier mydbcluster \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

Para verificar se o cluster de banco de dados está exportando logs de autenticação de banco de dados do IAM para o CloudWatch Logs, confira se o parâmetro `EnabledCloudwatchLogsExports` está definido como `iam-db-auth-error` na saída do comando `describe-db-instances`.

```
aws rds describe-db-cluster --region us-east-1 --db-cluster-identifier mydbcluster
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

Métricas do CloudWatch de autenticação de banco de dados do IAM

O Amazon Aurora fornece métricas quase em tempo real sobre a autenticação de banco de dados do IAM para sua conta do Amazon CloudWatch. A seguinte tabela lista as métricas de autenticação de banco de dados do IAM disponíveis usando o CloudWatch:

Métrica	Descrição
`IamDbAuthConnectionRequests`	Número total de solicitações de conexão feitas com a autenticação de banco de dados do IAM.
`IamDbAuthConnectionSuccess`	Número total de solicitações de autenticação de banco de dados do IAM bem-sucedidas.
`IamDbAuthConnectionFailure`	Número total de solicitações de autenticação de banco de dados do IAM malsucedidas.
`IamDbAuthConnectionFailureInvalidToken`	Número total de solicitações de autenticação de banco de dados do IAM malsucedidas devido a token inválido.
`IamDbAuthConnectionFailureInsufficientPermissions`	Número total de solicitações de autenticação de banco de dados do IAM malsucedidas devido a políticas ou permissões incorretas.
`IamDbAuthConnectionFailureThrottling`	Número total de solicitações de autenticação de banco de dados do IAM malsucedidas devido ao controle de utilização da autentica ção de banco de dados do IAM.

Métrica	Descrição
`IamDbAuthConnectionFailureServerError`	Número total de solicitações de autenticação de banco de dados do IAM malsucedidas devido a um erro interno do servidor no recurso de autenticação de banco de dados do IAM.

Problemas e soluções comuns de

Você pode encontrar os problemas a seguir ao usar a autenticação de banco de dados do IAM. Use as etapas de correção na tabela para resolver os problemas:

Erro	Métrica(s)	Causa	Solução

`[ERROR] Failed to authenticate the connection request for user db_user because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInvalidToken`
		O token de autenticação de banco de dados do IAM na solicitação de conexão não é um token SigV4a válido ou não está formatado corretamente.	Verifique a estratégia de geração de tokens em sua aplicação. Em alguns casos, observe se você está passando o token com uma formatação válida. Truncar o token (ou formatar incorretamente a string) tornará o token inválido.

`[ERROR] Failed to authenticate the connection request for user db_user because the token age is longer than 15 minutes. (Status`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInvalidToken`
		O token de autenticação de banco de dados do IAM expirou. Os tokens são válidos somente por 15 minutos.	Verifique a lógica de armazenamento em cache e/ou a reutilização de tokens em sua aplicação. Você não deve reutilizar tokens com mais de 15 minutos.

Erro	Métrica(s)	Causa	Solução
`Code: 400, Error Code:ExpiredToken)`			

`[ERROR] Failed to authorize the connection request for user` *`db_user`* `because the IAM policy assumed by the caller 'arn:aws:sts::123456789012:assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInsufficientPermissions`
		Esse erro pode ocorrer devido aos seguintes motivos:
		• A política do IAM assumida pela aplicação não autoriza a ação `rds-db:connect` .
		• Você está assumindo o perfil/política incorretos para *`db_user`* para se conectar ao banco de dados.
		• Você está assumindo a política correta para *`db_user`*, mas não está se conectando ao banco de dados correto.
			Verifique o perfil e/ou a política do IAM que você está assumindo em sua aplicação. Você deve assumir a mesma política para gerar o token e para se conectar ao banco de dados.

Erro	Métrica(s)	Causa	Solução

`[ERROR] Failed to authorize the connection request for user db_user due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Você está fazendo muitas solicitações de conexão ao banco de dados em um curto espaço de tempo. O limite de controle de utilização da autenticação de banco de dados do IAM é duzentas conexões por segundo.	Reduza a taxa de estabelecimento de novas conexões por meio da autenticação do IAM. Considere implementar o agrupamento de conexões usando o RDS Proxy para reutilizar as conexões estabelecidas em sua aplicação.

`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Houve um erro interno ao autorizar a conexão do banco de dados com a autenticação de banco de dados do IAM.	Entre em contato com https://aws.amazon.com/premiumsupport/ para investigar o problema.

Solução de problemas de identidade e acesso do Amazon Aurora

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o Aurora e o IAM.

Tópicos

- Não estou autorizado a realizar uma ação no Aurora
- Não estou autorizado a executar iam:PassRole
- Quero permitir que pessoas fora de minha conta da AWS acessem meus recursos do Aurora

Não estou autorizado a realizar uma ação no Aurora

Se o Console de gerenciamento da AWS informar que você não está autorizado a executar uma ação, será necessário entrar em contato com o administrador para obter assistência. Seu administrador é a pessoa que forneceu a você suas credenciais de início de sessão.

O exemplo de erro a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um *`widget`*, mas não tem as permissões `rds:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `rds:`*`GetWidget`*.

Não estou autorizado a executar iam:PassRole

Se você receber uma mensagem de erro informando que você não está autorizado a executar a ação `iam:PassRole`, entre em contato com o administrador para obter assistência. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão. Peça a essa pessoa para atualizar suas políticas para permitir que você passe uma função para o Aurora.

Alguns serviços da AWS permitem que você passe uma função existente para o serviço, em vez de criar um novo perfil de serviço ou perfil vinculado ao serviço. Para fazer isso, um usuário deve ter permissões para passar a função para o serviço.

O erro de exemplo a seguir ocorre quando uma usuária chamada `marymajor` tenta usar o console para executar uma ação no Aurora. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar a função para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Neste caso, Mary pede ao administrador para atualizar suas políticas para permitir que ela execute a ação `iam:PassRole`.

Quero permitir que pessoas fora de minha conta da AWS acessem meus recursos do Aurora

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte o seguinte:

- Para saber se o Aurora oferece suporte a esses recursos, consulte Como o Amazon Aurora funciona com o IAM.
- Para saber como conceder acesso a seus recursos em todas as contas da AWS pertencentes a você, consulte Fornecer acesso a um usuário do IAM em outra conta da AWS pertencente a você no Guia de usuário do IAM.
- Para saber como conceder acesso aos recursos para contas da AWS de terceiros, consulte Fornecer acesso a contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Registrar em log e monitorar no Amazon Aurora

O monitoramento é uma parte importante da manutenção da confiabilidade, da disponibilidade e da performance do Amazon Aurora e de suas soluções da AWS. É necessário coletar dados de monitoramento de todas as partes de sua solução da AWS para depurar uma falha de vários pontos com maior facilidade, caso ocorra. A AWS oferece várias ferramentas para monitorar seus recursos do Amazon Aurora e responder a possíveis incidentes:

Alarmes do Amazon CloudWatch

Com o uso de alarmes do Amazon CloudWatch, você observa uma única métrica durante um período especificado. Se a métrica ultrapassar um limite especificado, uma notificação será

enviada para um tópico do Amazon SNS ou para uma política do AWS Auto Scaling. Os alarmes do CloudWatch não invocam ações só porque estão em um determinado estado. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos.

AWS CloudTrailLogs do

O CloudTrail fornece um registro de ações executadas por um usuário, perfil ou serviço da AWS no Amazon Aurora. O CloudTrail captura todas as chamadas de API para o Amazon Aurora como eventos, inclusive as chamadas do console e de chamadas do código para operações de API do Amazon RDS. Usando as informações coletadas pelo CloudTrail, é possível determinar a solicitação que foi feita ao Amazon Aurora, o endereço IP da solicitação, quem fez a solicitação, quando ela foi feita e outros detalhes. Para ter mais informações, consulte Monitorar chamadas de API do Amazon Aurorano AWS CloudTrail .

Monitoramento avançado

O Amazon Aurora fornece métricas em tempo real para o sistema operacional (SO) no qual o cluster de banco de dados é executado. Você pode visualizar as métricas do cluster de banco de dados usando o console ou consumir o resultado do JSON de monitoramento avançado do Amazon CloudWatch Logs em um sistema de monitoramento de sua escolha. Para ter mais informações, consulte Monitorar métricas do SO com o monitoramento avançado .

Amazon RDS Performance Insights

O Insights de Performance expande os recursos de monitoramento existentes do Amazon Aurora para ilustrar a performance do banco de dados e ajudar a analisar todos os problemas que o afetam. Com o painel do Performance Insights, você pode visualizar a carga do banco de dados e filtrá-la por esperas, instruções SQL, hosts ou usuários. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora .

Logs de banco de dados

Você pode visualizar, baixar e observar os logs de banco de dados usando o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS. Para ter mais informações, consulte Monitorar arquivos de log do Amazon Aurora .

Recomendações do Amazon Aurora

O Amazon Aurora oferece recomendações automatizadas para recursos de banco de dados. Essas recomendações fornecem orientações de práticas recomendadas analisando a configuração, o uso e os dados de performance do cluster de banco de dados. Para ter mais informações, consulte Recomendações do Amazon Aurora .

Notificação de eventos do Amazon Aurora

OAmazon Aurora usa o Amazon Simple Notification Service (Amazon SNS) para fornecer uma notificação quando um evento do Amazon Aurora ocorre. Essas notificações podem estar em qualquer formato de notificação compatível com o Amazon SNS para uma região da AWS, como um e-mail, uma mensagem de texto ou uma chamada para um endpoint HTTP. Para ter mais informações, consulte Trabalhar com a notificação de eventos do Amazon RDS .

AWS Trusted Advisor

O Trusted Advisor conta com as práticas recomendadas aprendidas com o atendimento a centenas de milhões de clientes da AWS. O Trusted Advisor inspeciona seu ambiente da AWS e faz recomendações quando há oportunidades para economizar dinheiro, melhorar a performance e a disponibilidade do sistema e ajuda a corrigir falhas de segurança. Todos os clientes da AWS têm acesso a cinco verificações do Trusted Advisor. Os clientes com um plano de suporte Business ou Enterprise podem ver todas as verificações do Trusted Advisor.

O Trusted Advisor tem as seguintes verificações relacionadas ao Amazon Aurora:

- Instâncias ociosas de banco de dados do Amazon Aurora
- Risco de acesso de grupo de segurança do Amazon Aurora
- Backups do Amazon Aurora
- Multi-az do Amazon Aurora
- Acessibilidade da instância de bancos de dados Aurora

Para obter mais informações sobre essas verificações, consulte Práticas recomendadas do Trusted Advisor (verificações).

Streams de atividades do banco de dados

Os fluxos de atividades de banco de dados protegem os bancos de dados de ameaças internas controlando o acesso de DBA aos fluxos de atividades de banco de dados. Assim, a coleta, transmissão, armazenamento e subsequente processamento do fluxo de atividade de banco de dados está além do acesso dos administradores de banco de dados que gerenciam o banco de dados. Os fluxos de atividades de banco de dados podem ajudar a fornecer proteções para o banco de dados e atender aos requisitos regulatórios e de conformidade. Para obter mais informações, consulteMonitorar o Amazon Aurora com o recurso Database Activity Streams .

Para ter mais informações sobre monitoramento do Aurora, consulte Monitorar métricas em um cluster do Amazon Aurora.

Validação de conformidade do Amazon Aurora

Auditores de terceiros avaliam a segurança e a compatibilidade do Amazon Aurora como parte de vários programas de conformidade da AWS. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para obter uma lista dos serviços da AWS no escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo por programa de conformidade. Para obter informações gerais, consulte Programas de conformidade da AWS.

Você pode baixar relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Baixar os relatórios no AWS Artifact.

Sua responsabilidade com relação à conformidade ao usar o Amazon Aurora é determinada pela confidencialidade dos dados, pelos objetivos de compatibilidade da organização e pelos regulamentos e leis aplicáveis. A AWS fornece os seguintes recursos para ajudar com a compatibilidade:

- Guias de início rápido de segurança e conformidade: esses guias de implantação abordam as considerações de arquitetura e fornecem etapas para a implantação de ambientes de linha de base concentrados em conformidade e segurança na AWS.
- Architecting for HIPAA Security and Compliance on Amazon Web Services (Arquitetura para segurança e conformidade com HIPAA na Amazon Web Services): esse artigo técnico descreve como as empresas podem usar a AWS para criar aplicações em conformidade com os padrões HIPAA.
- Recursos de conformidade da AWS: esta coleção de manuais e guias pode ser aplicada ao seu setor e local.
- AWS Config: este produto da AWS avalia até que ponto suas configurações de recursos atendem adequadamente às práticas internas e às diretrizes e regulamentações do setor.
- AWS Security Hub CSPM: este AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança na AWS. O Security Hub CSPM usa controles de segurança para avaliar os recursos da AWS e conferir a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para acessar uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub CSPM.

Resiliência no Amazon Aurora

A infraestrutura global da AWS é criada com base em regiões da AWSe zonas de disponibilidade da AWS. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, as quais são conectadas com baixa latência, alto throughput e redes altamente redundantes. Com as zonas de disponibilidade, você pode projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte Infraestrutura global da AWS.

Além da infraestrutura global da AWS, o Auroraoferece recursos para ajudar a oferecer suporte às suas necessidades de resiliência e backup de dados.

Backup e restauração

O Aurora faz backup do volume de cluster automaticamente e mantém dados de restauração pela duração do período de retenção de backup. Os backups do Aurora são contínuos e incrementais para que você possa restaurar rapidamente em qualquer ponto do período de retenção de backup. Quando os dados do backup estão sendo gravados, não há nenhum impacto sobre a performance ou interrupção de serviço do banco de dados. Você pode especificar um período de retenção de backup, de 1 a 35 dias, ao criar ou modificar um cluster de banco de dados.

Se você quiser manter um backup além do período de retenção do backup, também será possível fazer um snapshot dos dados no seu volume de cluster. O Aurora retém dados de restauração incrementais durante todo o período de retenção de backup. Assim, você precisa criar um snapshot apenas para os dados que deseja reter além do período de retenção do backup. Crie um novo cluster de banco de dados a partir do snapshot.

Você pode recuperar seus dados criando um novo cluster de bancos de dados Aurora a partir dos dados de backup retidos pelo Aurora ou a partir de um snapshot de cluster de banco de dados que você salvou. É possível criar rapidamente uma nova cópia de um cluster de banco de dados a partir dos dados de backup a qualquer momento no período de retenção do backup. Devido à natureza contínua e incremental dos backups do Aurora durante o período de retenção do backup, você não precisa fazer snapshots frequentes de seus dados para melhorar os tempos de restauração.

Para obter mais informações, consulte Como fazer o backup e a restauração de um cluster de banco de dados do Amazon Aurora.

Replicação

As réplicas do Aurora são endpoints independentes em um cluster de banco de dados Aurora, cuja melhor utilidade é escalar operações de leitura e aumentar a disponibilidade. Até 15 réplicas do Aurora podem ser distribuídas entre as zonas de disponibilidade abrangidas por um cluster de banco de dados em uma região da AWS. O volume do cluster de banco de dados é composto por várias cópias dos dados do cluster de banco de dados. Contudo, os dados no volume do cluster são representados como um volume lógico, único, para a instância do banco de dados primário e para réplicas do Aurora no cluster de banco de dados. Se a instância do banco de dados primário falhar, uma réplica do Aurora é promovida para ser a instância primária do banco de dados.

O Aurora também oferece suporte a opções de replicação específicas do Aurora MySQL e do Aurora PostgreSQL.

Para obter mais informações, consulte Replicação com o Amazon Aurora.

Failover

O Aurora armazena cópias dos dados em um cluster de banco de dados em várias zonas de disponibilidade em uma única região da AWS. Esse armazenamento ocorre independentemente de as instâncias de banco de dados no cluster de banco de dados abrangerem várias zonas de disponibilidade. Quando você cria réplicas do Aurora nas zonas de disponibilidade, o Aurora as provisiona e as mantém automaticamente, de maneira síncrona. A instância do banco de dados primário é replicada de maneira síncrona em zonas de disponibilidade para réplicas do Aurora a fim de fornecer a redundância de dados, eliminar os congelamentos de E/S e minimizar os picos de latência durante backups do sistema. Executar um cluster de banco de dados com alta disponibilidade pode aumentar a disponibilidade durante a manutenção planejada do sistema e ajudar a proteger os bancos de dados contra falhas e interrupções da zona de disponibilidade.

Para mais informações, consulte Alta disponibilidade do Amazon Aurora.

Segurança da infraestrutura no Amazon Aurora

Como um serviço gerenciado, o Amazon Relational Database Service é protegido pela segurança de rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança da infraestrutura, consulte Proteção de Infraestrutura em Pilar de Segurança: AWS Estrutura bem arquitetada.

Você usa chamadas de API publicadas da AWS para acessar o Amazon RDS por meio da rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, o Aurora oferece recursos para ajudar a oferecer suporte à segurança da infraestrutura.

Grupos de segurança

Os grupos de segurança controlam o acesso que o tráfego tem dentro e fora de um cluster de banco de dados. Por padrão, o acesso à rede é desativado para um cluster de banco de dados. É possível especificar regras em um grupo de segurança que permitem o acesso de um intervalo de endereço IP, de uma porta ou de grupo de segurança. Depois que as regras de entrada são configuradas, as mesmas regras se aplicam a todos os clusters de banco de dados associados a esse grupo de segurança.

Para obter mais informações, consulte Controlar acesso com grupos de segurança.

Public accessibility

Quando você inicia uma instância de banco de dados dentro de uma nuvem privada virtual (VPC) com base no serviço da Amazon VPC, pode ativar ou desativar a acessibilidade pública para essa instância de banco de dados. Para designar se a instância de banco de dados que você cria tem um nome DNS que é determinado como um endereço IP público, use o parâmetro Public accessibility. Usando esse parâmetro, você pode designar se há acesso público à instância do banco de dados. Você pode modificar uma instância de banco de dados para ativar ou desativar a acessibilidade pública modificando o parâmetro Public accessibility (Acessibilidade pública).

Para obter mais informações, consulte Ocultar um cluster de banco de dados em uma VPC da Internet.

> **Note**
>
> Se sua instância de banco de dados estiver em uma VPC, mas não estiver acessível publicamente, também será possível usar uma conexão AWS Site-to-Site VPN ou uma conexão do Direct Connect para acessá-la de uma rede privada. Para obter mais informações, consulte Privacidade do tráfego entre redes.

API do Amazon RDS e endpoints da VPC de interface (AWS PrivateLink)

É possível estabelecer uma conexão privada entre a VPC e os endpoints da API do Amazon RDS criando um VPC endpoint de interface. Os endpoints de interface são desenvolvidos pelo AWS PrivateLink.

O AWS PrivateLink permite que você acesse as operações de API do Amazon RDS sem um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do Direct Connect. As instâncias de banco de dados na VPC não precisam de endereços IP públicos para se comunicarem com endpoints de API do Amazon RDS para executar, modificar ou encerrar instâncias e clusters de banco de dados. As instâncias de banco de dados também não precisam de endereços IP públicos para usar qualquer uma das operações de API do RDS disponíveis. O tráfego entre seu VPC e Amazon RDS não deixa a rede da Amazon.

Cada endpoint de interface é representado por uma ou mais interfaces de rede elástica nas sub-redes. Para obter mais informações sobre interfaces de rede elástica, consulte Interfaces de rede elástica no Guia do usuário do Amazon EC2.

Para obter mais informações sobre limites de VPC , consulte VPC endpoints de interface (AWS PrivateLink) no Guia do usuário da Amazon VPC. Para obter informações sobre as operações da API do RDS, consulte Referência da API do Amazon RDS.

Você não precisa de um endpoint da VPC de interface para se conectar a um cluster de banco de dados. Para obter mais informações, consulte Cenários para acessar um cluster de banco de dados em uma VPC.

Considerações sobre VPC endpoints

Antes de configurar um VPC endpoint de interface para endpoints da API do Amazon RDS, revise Propriedades e limitações do endpoint de interface no Guia do usuário da Amazon VPC.

Todas as operações de API do RDS relevantes para o gerenciamento de recursos do Amazon Aurora estão disponíveis na VPC usando o AWS PrivateLink.

As políticas de endpoint da VPC têm suporte para endpoints da API do RDS. Por padrão, o acesso total às operações de API do RDS é permitido através do endpoint. Para obter mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Disponibilidade

No momento, a API do Amazon RDS é compatível com os limites da VPC nas seguintes regiões da AWS:

- Leste dos EUA (Ohio)
- Leste dos EUA (Norte da Virgínia)
- Oeste dos EUA (N. da Califórnia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Asia Pacific (Mumbai)
- Ásia-Pacífico (Nova Zelândia)
- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Taipei)
- Ásia-Pacífico (Tailândia)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Oeste do Canadá (Calgary)
- China (Pequim)
- China (Ningxia)
- Europa (Frankfurt)
- Europa (Zurique)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Paris)
- Europa (Estocolmo)

- Europa (Milão)
- Israel (Tel Aviv)
- México (Centro)
- Middle East (Bahrain)
- South America (São Paulo)
- AWS GovCloud (Leste dos EUA)
- AWS GovCloud (Oeste dos EUA)

Criar um VPC endpoint de interface para a API Amazon RDS

Você pode criar um endpoint da VPC para o serviço de APIs do Amazon RDS usando o console da Amazon VPC ou a AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC.

Crie um VPC endpoint para a API Amazon RDS usando o nome de serviço `com.amazonaws.region.rds`.

Ao excluir regiões da AWS na China, se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API ao Amazon RDS com o endpoint da VPC usando seu nome DNS padrão para a região da AWS, por exemplo, `rds.us-east-1.amazonaws.com`. Para as regiões China (Pequim) e China (Ningxia) da AWS, é possível fazer solicitações de API com o endpoint da VPC usando `rds-api.cn-north-1.amazonaws.com.cn` e `rds-api.cn-northwest-1.amazonaws.com.cn`, respectivamente.

Para obter mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

Criar uma política de VPC endpoint para a API Amazon RDS

É possível anexar uma política de endpoint ao VPC endpoint que controla o acesso à API Amazon RDS Essa política especifica as seguintes informações:

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos sobre os quais as ações podem ser realizadas.

Para obter mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Exemplo: política de VPC endpoint para ações da API Amazon RDS

Veja a seguir um exemplo de uma política de endpoint da API Amazon RDS. Quando anexada a um endpoint, essa política concede acesso às ações indicadas da API Amazon RDS para todos os principais em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Exemplo: política de endpoint da VPC que nega todo o acesso de uma conta da AWS especificada

A política de endpoint da VPC a seguir nega à conta da AWS 123456789012 todo o acesso aos recursos que usam o limite. A política permite todas as ações de outras contas.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
```

```
    ]
}
```

Práticas recomendadas de segurança do Amazon Aurora

Use contas do AWS Identity and Access Management (IAM) para controlar o acesso a operações da API do Amazon RDS, especialmente operações que criam, modificam ou excluem recursos do Amazon Aurora. Esses recursos incluem clusters de banco de dados, grupos de segurança e grupos de parâmetros. Além disso, use o IAM para controlar ações que executam ações administrativas comuns, como fazer backup e restaurar clusters de banco de dados.

- Crie um usuário individual para cada pessoa que gerencia recursos do Amazon Aurora, incluindo você. Não use as credenciais-raiz da AWS para gerenciar recursos do Amazon Aurora.
- Conceda a cada usuário o conjunto mínimo de permissões necessárias para realizar suas funções.
- Use grupos do IAM para gerenciar efetivamente permissões para vários usuários.
- Mude suas credenciais do IAM regularmente.
- Configure o AWS Secrets Manager para alternar automaticamente os segredos para o Amazon Aurora. Para ter mais informações, consulte Alternar os segredos do AWS Secrets Manager no Guia do usuário do AWS Secrets Manager. Também é possível recuperar a credencial do AWS Secrets Manager forma programática. Para ter mais informações, consulte Recuperar o valor do segredo no Guia do usuário do AWS Secrets Manager.

Para ter mais informações sobre a segurança do Amazon Aurora, consulte Segurança no Amazon Aurora. Para ter mais informações sobre o IAM, consulte AWS Identity and Access Management. Para obter informações sobre as práticas recomendadas do IAM, acesse Melhores práticas do IAM.

O AWS Security Hub CSPM utiliza controles de segurança para avaliar configurações de recursos e padrões de segurança que ajudam você a cumprir várias frameworks de conformidade. Para ter mais informações sobre como usar o Security Hub CSPM para avaliar os recursos do RDS, consulte Controles do Amazon Relational Database Service no Guia do usuário do AWS Security Hub.

É possível monitorar o uso do RDS em relação às práticas recomendadas de segurança com o Security Hub CSPM. Para ter mais informações, consulte O que é o AWS Security Hub CSPM?.

Use o Console de gerenciamento da AWS, a AWS CLI ou a API do RDS para alterar a senha do usuário mestre. Se você usar outra ferramenta, como um cliente SQL, para alterar a senha do usuário mestre, isso poderá resultar na revogação de privilégios ao usuário involuntariamente.

O Amazon GuardDuty é um serviço contínuo de monitoramento de segurança que analisa e processa uma série de fontes de dados, incluindo a atividade de login do Amazon RDS. Ele usa feeds de inteligência sobre ameaças e machine learning para identificar atividades inesperadas e possivelmente não autorizadas e mal-intencionadas no ambiente da AWS.

Quando a Proteção RDS do Amazon GuardDuty detecta uma ameaça em potencial ou uma tentativa de login anômala que indica uma ameaça ao banco de dados, o GuardDuty gera uma nova descoberta com detalhes sobre o banco de dados possivelmente comprometido. Para ter mais informações, consulte Monitorar ameaças com o Amazon GuardDuty RDS Protection para Amazon Aurora .

Controlar acesso com grupos de segurança

Os grupos de segurança de VPC controlam o acesso que o tráfego tem dentro e fora de um cluster de banco de dados. Por padrão, o acesso à rede é desativado para um cluster de banco de dados. É possível especificar regras em um grupo de segurança que permitem o acesso de um intervalo de endereço IP, de uma porta ou de grupo de segurança. Depois que as regras de entrada são configuradas, as mesmas regras se aplicam a todos os clusters de banco de dados associados a esse grupo de segurança. Você pode especificar até 20 regras no grupo de segurança.

Visão geral dos grupos de segurança de VPC

Cada regra de grupo de segurança de VPC possibilita que uma origem específica acesse um cluster de banco de dados em uma VPC que esteja associada a esse grupo de segurança de VPC. A origem pode ser uma gama de endereços (por exemplo, 203.0.113.0/24) ou outro grupo de segurança da VPC. Ao especificar um grupo de segurança de VPC como origem, você permite o tráfego recebido de todas as instâncias (geralmente servidores de aplicações) que usam o grupo de segurança de VPC de origem. Os grupos de segurança de VPC podem ter regras que controlam o tráfego de entrada e saída. No entanto, as regras de tráfego de saída normalmente não se aplicam a clusters de banco de dados. As regras de tráfego de saída se aplicarão apenas se o cluster de banco de dados atuar como um cliente. É necessário usar a API do Amazon EC2 ou a opção Security Group (Grupo de segurança) no console da VPC para criar grupos de segurança de VPC.

Quando você cria regras para o grupo de segurança de VPC que permitem acessar os clusters na sua VPC, você deve especificar uma porta para cada intervalo de endereços ao qual a regra permite o acesso. Por exemplo, se quiser ativar o acesso via Secure Shell (SSH) para instâncias na VPC, crie uma regra que permitirá o acesso à porta TCP 22 para o intervalo de endereços especificado.

É possível configurar vários grupos de segurança de VPC que permitem o acesso a diferentes portas para diferentes instâncias na sua VPC. Por exemplo, você pode criar um grupo de segurança da VPC que permite acessar a porta TCP 80 para servidores web na VPC. Você pode criar outro grupo de segurança de VPC que permita acessar a porta TCP 3306 para instâncias de bancos de dados do Aurora MySQL na VPC.

> **Note**
>
> Em um cluster de bancos de dados Aurora, o grupo de segurança da VPC associado a esse cluster também é associado a todas as instâncias de banco de dados que ele contém. Se você alterar o grupo de segurança da VPC do cluster de banco de dados ou de uma instância de banco de dados, a alteração será aplicada automaticamente a todas as instâncias de banco de dados nesse cluster.

Para ter mais informações sobre grupos de segurança da VPC, consulte Grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

> **Note**
>
> Se o cluster de banco de dados estiver em uma VPC, mas não acessível publicamente, também será possível usar um Site-to-Site VPN ou uma conexão do Direct Connect para acessá-la de uma rede privada. Para ter mais informações, consulte Privacidade do tráfego entre redes .

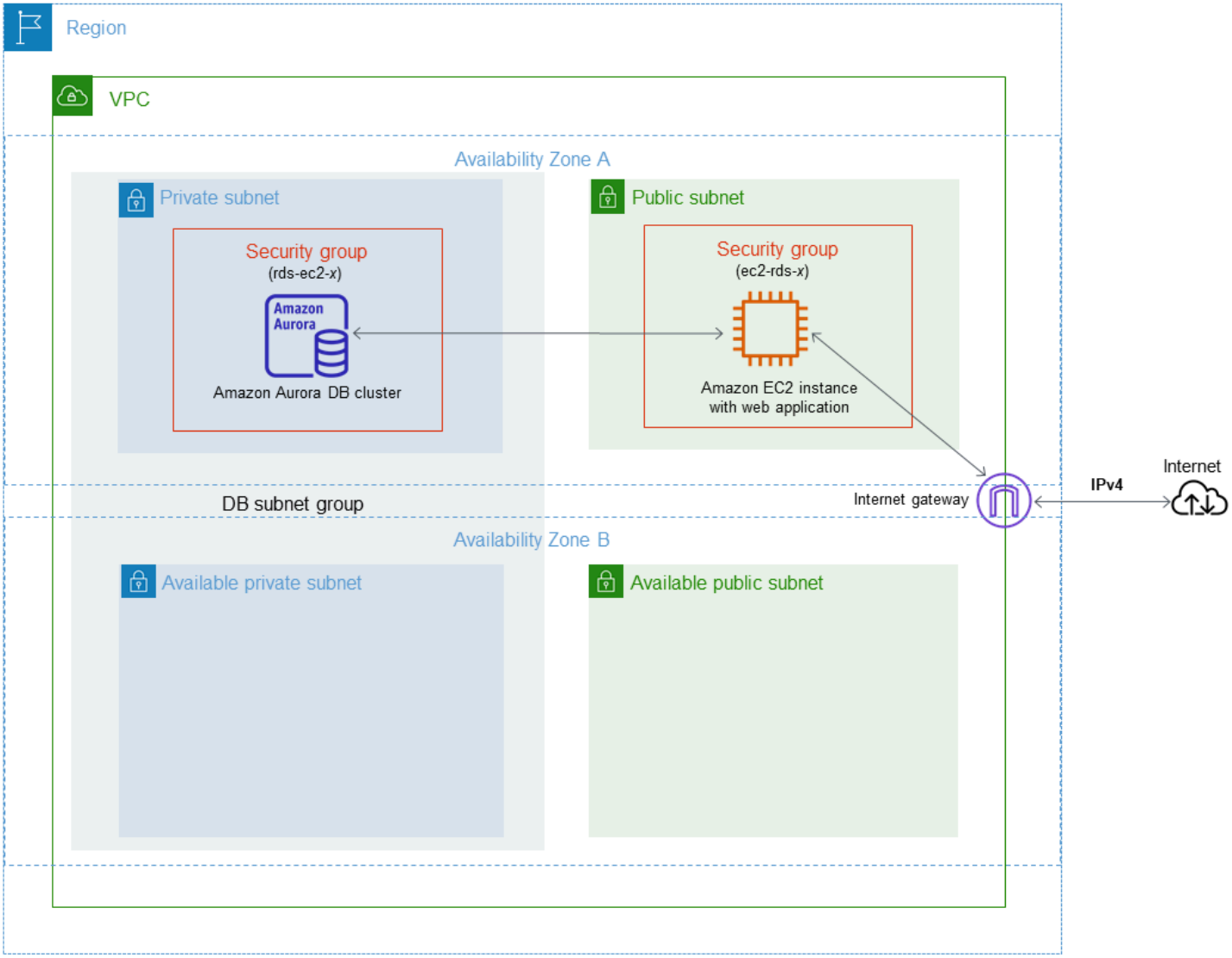
Cenário de grupos de segurança

Um uso comum de um cluster de banco de dados em uma VPC é compartilhar dados com um servidor de aplicações executado em uma instância do Amazon EC2 na mesma VPC, acessado por uma aplicação cliente fora da VPC. Para este cenário, use as páginas do RDS e da VPC no Console de gerenciamento da AWS ou nas operações de API do RDS e do EC2 para criar as instâncias e os grupos de segurança necessários:

1. Crie um grupo de segurança de VPC (por exemplo, `sg-0123ec2example`) e defina as regras de entrada que utilizam os endereços IP da aplicação cliente como a origem. Esse grupo de segurança permite que sua aplicação cliente se conecte a instâncias do EC2 em uma VPC que usa esse grupo de segurança.

2. Crie uma instância do EC2 para a aplicação e adicione a instância do EC2 ao grupo de segurança de VPC (`sg-0123ec2example`) que você criou na etapa anterior.
3. Crie um segundo grupo de segurança de VPC (por exemplo, `sg-6789rdsexample`) e crie uma nova regra especificando o grupo de segurança de VPC que você criou na etapa 1 (`sg-0123ec2example`) como origem.
4. Crie um cluster de banco de dados e adicione a o cluster de banco de dados ao grupo de segurança de VPC (`sg-6789rdsexample`) que você criou na etapa anterior. Quando você criar o cluster de banco de dados, use o mesmo número de porta especificado para a regra do grupo de segurança de VPC (`sg-6789rdsexample`) que você criou na etapa 3.

O diagrama a seguir mostra esse cenário.



Para obter instruções detalhadas sobre como configurar uma VPC para esse cenário, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4). Para ter mais informações sobre como usar uma VPC, consulte Amazon VPC e Amazon Aurora.

Criar um grupo de segurança de VPC

Você pode criar um grupo de segurança de VPC para uma instância de banco de dados usando o console da VPC. Para obter informações sobre como criar um grupo de segurança, consulte Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança e Grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

Associação de um grupo de segurança a um cluster de banco de dados

Você pode associar um grupo de segurança a um cluster de banco de dados usando Modify cluster (Modificar cluster) no console do RDS, a API `ModifyDBCluster` do Amazon RDS ou o comando `modify-db-cluster` da AWS CLI.

O exemplo da CLI a seguir associa um grupo de VPC específico e remove grupos de segurança de banco de dados do cluster de banco de dados.

```
aws rds modify-db-cluster --db-cluster-identifier dbName --vpc-security-group-ids sg-ID
```

Para ter informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Privilégios da conta de usuário mestre

Ao criar um cluster de banco de dados, o usuário principal padrão usado obtém determinados privilégios para esse cluster de banco de dados. Não é possível alterar o nome de usuário principal depois que o cluster de banco de dados é criado.

> **Important**
>
> É altamente recomendável não usar o usuário mestre diretamente nas aplicações. Em vez disso, siga as práticas recomendadas de usar um usuário do banco de dados criado com os privilégios mínimos obrigatórios para a aplicação.

> **Note**
>
> Se você excluir acidentalmente as permissões do usuário principal, poderá restaurá-las modificando o cluster de banco de dados e definindo uma nova senha de usuário principal. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

A tabela a seguir mostra os privilégios e as funções de banco de dados que o usuário mestre obtém para cada um dos mecanismos do banco de dados.

Mecanismo do banco de dados	Privilégio do sistema	Função do banco de dados
Aurora MySQL	Versão 2: `ALTER,ALTER ROUTINE, CREATE,CREATE ROUTINE,CREATE TEMPORARY TABLES,CREATE USER,CREATE VIEW,DELETE,DROP, EVENT,EXECUTE,GRANT OPTION,INDEX,INSERT, LOAD FROM S3,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATI ON CLIENT , REPLICATION SLAVE ,SELECT, SELECT INTO S3,SHOW DATABASES , SHOW VIEW,TRIGGER, UPDATE`	—
	Versão 3: `ALTER,APPLICATION_PASSWORD_ADMIN , ALTER ROUTINE,CONNECTION_ADMIN , CREATE,CREATE ROLE,CREATE ROUTINE,CREATE TEMPORARY TABLES, CREATE USER,CREATE VIEW, DELETE,DROP,DROP ROLE, EVENT,EXECUTE,INDEX, INSERT,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATION CLIENT , REPLICATI`	`rds_superuser_role` Para obter mais informações sobre o rds_superuser_role, consulte Modelo de privilégios baseados em funções.

Mecanism do banco de dados	Privilégio do sistema	Função do banco de dados
	`ON SLAVE ,ROLE_ADMIN , SET_USER_ID ,SELECT,SHOW DATABASES ,SHOW VIEW,TRIGGER, UPDATE,XA_RECOVER_ADMIN` A partir do Aurora MySQL versão 3.04.0, o usuário principal também recebe o privilégio `SHOW_ROUTINE` . A partir do Aurora MySQL versão 3.09.0, o usuário principal também recebe os privilégios `FLUSH_OPTIMIZER_COSTS` , `FLUSH_STATUS` , `FLUSH_TABLES` e `FLUSH_USER_RESOURCES` .	
Aurora PostgreSQL	`LOGIN,NOSUPERUSER , INHERIT,CREATEDB, CREATEROLE ,NOREPLICATION ,VALID UNTIL 'infinity'`	`RDS_SUPERUSER` Para obter mais informações sobre RDS_SUPERUSER, consulte Noções básicas de perfis e permissões do PostgreSQL.

Usar funções vinculadas ao serviço do Amazon Aurora

O Amazon Aurora usa funções vinculadas ao serviço do AWS Identity and Access Management (IAM). O perfil vinculado ao serviço é um tipo exclusivo de perfil do IAM vinculado diretamente ao Amazon Aurora. Os perfis vinculados a serviços são predefinidos pelo Amazon Aurora e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita o uso do Amazon Aurora porque você não precisa adicionar as permissões necessárias manualmente. O Amazon Aurora define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon Aurora pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Você pode excluir as funções somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do Amazon Aurora, pois você não pode remover por engano as permissões para acessar os recursos.

Para ter informações sobre outros serviços compatíveis com perfis vinculados ao serviço, consulte Serviços da AWS compatíveis com o IAM e procure serviços que tenham Sim na coluna Perfis vinculados ao serviço. Escolha um Sim com um link para visualizar a documentação da função vinculada a serviço desse serviço.

Permissões de função vinculada ao serviço do Amazon Aurora

O Amazon Aurora utiliza a função vinculada a serviço chamada AWSServiceRoleForRDS para permitir que o Amazon RDS chame serviços da AWS em nome dos seus clusters de banco de dados.

A função vinculada ao serviço AWSServiceRoleForRDS confia nos seguintes serviços para assumir a função:

- `rds.amazonaws.com`

Essa função vinculada a serviços tem uma política de permissões anexada a ela, chamada `AmazonRDSServiceRolePolicy`, que concede permissões para operar na conta.

Para ter mais informações sobre essa política, incluindo o documento de política JSON, consulte AmazonRDSServiceRolePolicy no Guia de referência de políticas gerenciadas pela AWS.

> **Note**
>
> É necessário configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Se encontrar a seguinte mensagem de erro:
> Impossível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.
> Certifique-se de que você tem as seguintes permissões ativadas:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Para ter mais informações, consulte Service-linked role permissions (Permissões de nível vinculado a serviços) no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon Aurora

Você não precisa criar manualmente uma função vinculada a serviço. Ao criar um cluster de banco de dados, o Amazon Aurora cria o perfil vinculado ao serviço para você novamente.

> **Important**
>
> Se você já usava o serviço Amazon Aurora antes de 1.º de dezembro de 2017, quando ele começou a comportar perfis vinculados ao serviço, o Amazon Aurora já criou o perfil AWSServiceRoleForRDS em sua conta. Para saber mais, consulte Uma nova função apareceu na minha conta da AWS.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Ao criar um cluster de banco de dados, o Amazon Aurora cria o perfil vinculado ao serviço para você novamente.

Editar uma função vinculada ao serviço para o Amazon Aurora

O Amazon Aurora não permite que você edite a função vinculada ao serviço AWSServiceRoleForRDS. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, será possível editar a descrição do perfil usando o IAM. Para ter mais informações, consulte Editar uma função vinculada a serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço para o Amazon Aurora

Se você não precisar mais usar um recurso ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. Contudo, você deve excluir todos os seus clusters de banco de dados antes de poder excluir a função vinculada ao serviço.

Limpar uma função vinculada ao serviço

Antes de você poder usar o IAM para excluir uma função vinculada ao serviço, você deve primeiro confirmar que a função não tem sessões ativas e remover quaisquer recursos usados pela função.

Para verificar se a função vinculada ao serviço tem uma sessão ativa no console do IAM

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, escolha Roles (Perfis). A seguir, escolha o nome (não a caixa de seleção) da função AWSServiceRoleForRDS.
3. Na página Resumo do perfil escolhido, selecione a guia Último acesso.
4. Na guia Último acesso, analise a atividade recente para o perfil vinculado ao serviço.

> **Note**
>
> Se não tiver certeza se o Amazon Aurora está usando a função AWSServiceRoleForRDS, você pode tentar excluir a função. Se o serviço estiver usando a função, a exclusão falhará e você poderá visualizar as regiões da AWS em que a função está sendo usada. Se a função está sendo usada, você deve aguardar a sessão

final antes de excluir a função. Você não pode revogar a sessão para uma função vinculada a serviço.

Para remover a função AWSServiceRoleForRDS, primeiro é necessário excluir todas os clusters de banco de dados.

Exclusão de todos os clusters

Siga um destes procedimentos para excluir um cluster. Repita o procedimento para cada um dos clusters.

Para excluir um cluster (console)

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Na lista Databases (Bancos de dados), escolha o cluster que você deseja excluir.
3. Em Cluster Actions (Ações de cluster), escolha Delete (Excluir).
4. Escolha Delete (Excluir).

Para excluir um cluster (CLI)

Consulte `delete-db-cluster` no AWS CLI Command Reference.

Para excluir um cluster (API)

Consulte `DeleteDBCluster` no Amazon RDS API Reference.

Também é possível usar o console do IAM, a CLI do IAM ou a API do IAM para excluir a função AWSServiceRoleForRDS vinculada a serviço. Para ter mais informações, consulte Excluir uma função vinculada ao serviço no Guia do usuário do IAM.

Permissões de perfis vinculados ao serviço para o Amazon RDS Beta

O Amazon Aurora usa o perfil vinculado ao serviço chamado `AWSServiceRoleForRDSBeta` para permitir que o Amazon Aurora chame serviços da AWS em nome dos recursos de banco de dados do RDS.

O perfil vinculado ao serviço AWSServiceRoleForRDSBeta confia nos seguintes serviços para assumir o perfil:

- `rds.amazonaws.com`

Essa função vinculada a serviços tem uma política de permissões anexada a ela, chamada `AmazonRDSBetaServiceRolePolicy`, que concede permissões para operar na conta. Para obter mais informações, consulte Política gerenciada da AWS: AmazonRDSBetaServiceRolePolicy.

> **Note**
>
> É necessário configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Se encontrar a seguinte mensagem de erro:
> Impossível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.
> Certifique-se de que você tem as seguintes permissões ativadas:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Para ter mais informações, consulte Service-linked role permissions (Permissões de nível vinculado a serviços) no Guia do usuário do IAM.

Perfil vinculado ao serviço da Pré-visualização do Amazon RDS

O Amazon Aurora usa o perfil vinculado ao serviço chamado `AWSServiceRoleForRDSPreview` para permitir que o Amazon Aurora chame serviços da AWS em nome dos recursos de banco de dados do RDS.

O perfil vinculado ao serviço AWSServiceRoleForRDSPreview confia nos seguintes serviços para assumir o perfil:

- `rds.amazonaws.com`

Essa função vinculada a serviços tem uma política de permissões anexada a ela, chamada `AmazonRDSPreviewServiceRolePolicy`, que concede permissões para operar na conta. Para obter mais informações, consulte Política gerenciada da AWS: AmazonRDSPreviewServiceRolePolicy.

> **Note**
>
> É necessário configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Se encontrar a seguinte mensagem de erro:
> Impossível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.
> Certifique-se de que você tem as seguintes permissões ativadas:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Para ter mais informações, consulte Service-linked role permissions (Permissões de nível vinculado a serviços) no Guia do usuário do IAM.

Amazon VPC e Amazon Aurora

A Amazon Virtual Private Cloud (Amazon VPC) possibilita a execução de recursos da AWS, como clusters de bancos de dados do Aurora, em uma nuvem privada virtual (VPC).

Ao usar uma VPC, você tem controle sobre o ambiente de rede virtual. É possível escolher seu próprio intervalo de endereços IP, criar sub-redes e configurar o roteamento e listas de controle de acesso. Não há custos adicionais para executar o cluster de banco de dados em uma VPC.

As contas têm uma VPC padrão. Todos os novos clusters de banco de dados são criados na VPC padrão, a menos que você especifique o contrário.

Tópicos

- Trabalhar com um cluster de banco de dados em uma VPC
- Cenários para acessar um cluster de banco de dados em uma VPC
- Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4)
- Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla)

Veja a seguir uma discussão sobre a funcionalidade da VPC relevante para clusters de banco de dados do Amazon Aurora. Para obter mais informações sobre uma Amazon VPC, consulte o Guia de conceitos básicos da Amazon VPC e Guia do usuário da Amazon VPC.

Trabalhar com um cluster de banco de dados em uma VPC

Seu cluster deve estar em uma nuvem privada virtual (VPC). Uma VPC é uma rede virtual logicamente isolada de outras redes virtuais na Nuvem AWS. A Amazon VPC permite que você execute recursos da AWS, como um cluster de banco de dados do Amazon Aurora ou instância do Amazon EC2, em uma VPC. A VPC pode ser uma VPC padrão que vem com sua conta ou aquele que você criou. Todas as VPCs estão associadas à sua conta da AWS.

Sua VPC padrão possui três sub-redes que você pode usar para isolar recursos dentro da VPC. A VPC padrão também possui um gateway da Internet que pode ser usado para fornecer acesso a recursos na VPC de fora da VPC.

Para obter uma lista de cenários envolvendo clusters de banco de dados do Amazon Aurora em uma VPC , consulte Cenários para acessar um cluster de banco de dados em uma VPC.

Tópicos

- Trabalhar com um cluster de banco de dados em uma VPC
- Controle de criptografia da VPC
- Trabalhar com grupos de sub-redes de banco de dados
- Sub-redes compartilhadas
- Endereçamento IP do Amazon Aurora
- Ocultar um cluster de banco de dados em uma VPC da Internet
- Criar um cluster de banco de dados em uma VPC

Nos tutoriais a seguir, você pode aprender a criar uma VPC a ser usar para um cenário comum do Amazon Aurora:

- Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4)
- Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla)

Trabalhar com um cluster de banco de dados em uma VPC

Veja a seguir algumas dicas sobre como trabalhar com um cluster de banco de dados em uma VPC:

- A VPC deve ter pelo menos duas sub-redes. Essas sub-redes devem estar em duas zonas de disponibilidade diferentes na Região da AWS onde você deseja implantar o cluster de banco de dados. Uma sub-rede é um segmento do intervalo de endereços IP de uma VPC que você pode especificar e que permite agrupar clusters com base nas suas necessidades operacionais e de segurança.
- Se quiser que seu cluster de banco de dados na VPC seja publicamente acessível, você deverá habilitar os atributos DNS hostnames (Nomes de host de DNS) e DNS resolution (Resolução de DNS).
- Sua VPC deve ter um grupo de sub-redes de banco de dados que você criou. Crie um grupo de sub-redes de banco de dados especificando as sub-redes criadas. O Amazon Aurora escolhe uma sub-rede e um endereço IP dentro dessa sub-rede para associar à instância de banco de dados primária no cluster de banco de dados. A instância de banco de dados primária usa a zona de disponibilidade que contém a sub-rede.
- Sua VPC deve ter um grupo de segurança de VPC que permita o acesso ao cluster de banco de dados.

Para ter mais informações, consulte Cenários para acessar um cluster de banco de dados em uma VPC.

- Os blocos CIDR em cada uma das suas sub-redes devem ser suficientemente grandes para acomodar endereços IP sobressalentes para o Amazon Aurora usar durante atividades de manutenção, incluindo failover e escalabilidade de computação. Por exemplo, um intervalo como 10.0.0.0/24 e 10.0.1.0/24 normalmente é grande o suficiente.
- Uma VPC pode ter um atributo instance tenancy (locação de instâncias) com o valor default (padrão) ou dedicated (dedicado). Todas as VPC padrão têm o atributo de locação de instâncias definido como padrão, e uma VPC padrão pode oferecer suporte a qualquer classe de instância de banco de dados.

  Se você optar por ter seu cluster de banco de dados em uma VPC dedicada em que o atributo de locação de instâncias esteja definido como dedicado, a classe de seu cluster de banco de dados deverá ser um dos tipos aprovados de instância dedicada do Amazon EC2. Por exemplo, a instância dedicada r5.large do EC2 corresponde à classe de instância de banco de dados db.r5.large. Para obter informações sobre a locação de instâncias em uma VPC, consulte Instâncias dedicadas no Guia do usuário do Amazon Elastic Compute Cloud.

  Consulte mais informações sobre os tipos de instância que podem estar em uma instância dedicada em Instâncias dedicadas do Amazon EC2, na página de preços do Amazon EC2.

  > **Note**
  >
  > Quando você define o atributo de locação de instâncias como dedicado para um cluster de banco de dados, ele não garante que esse cluster terá execução em um host dedicado.

Controle de criptografia da VPC

Os controles de criptografia da VPC permitem que você aplique a criptografia em trânsito para todo o tráfego de rede em suas VPCs. Use o controle de criptografia para atender aos requisitos de conformidade regulatória, garantindo que somente hardware baseado em Nitro habilitado para criptografia possa ser provisionado em VPCs designadas. O controle de criptografia também detecta problemas de compatibilidade no momento da solicitação da API, e não durante o provisionamento. Suas workloads existentes continuam funcionando e somente novas solicitações incompatíveis são bloqueadas.

Defina seus controles de criptografia de VPC definindo o modo de controle de VPC como:

- desabilitado (padrão)
- monitorar o
- imposta

Para conferir o modo de controle atual da sua VPC, use o comando Console de gerenciamento da AWS da CLI ou da API DescribeVpcs.

Se sua VPC impõe criptografia, você só pode provisionar clusters de banco de dados com base em Nitro que aceitem criptografia em trânsito nessa VPC. Para obter mais informações, consulte, Tipos de classe de instância de banco de dados. Para acessar informações sobre instâncias Nitro, consulte Instâncias baseadas no AWS Sistema Nitro no Guia do usuário do Amazon EC2.

> **Note**
>
> Se você tentar provisionar clusters de banco de dados incompatíveis em uma VPC com criptografia imposta, o Aurora exibirá uma exceção `VpcEncryptionControlViolationException`.

Para Aurora Serverless para MySQL e PostreSQL, o controle de criptografia requer a versão da plataforma 3 ou posterior.

Trabalhar com grupos de sub-redes de banco de dados

Sub-redes são segmentos do intervalo de endereços IP de uma VPC que você designa para agrupar seus recursos com base em necessidades operacionais e de segurança. Um grupo de sub-redes de banco de dados é uma coleção de sub-redes (geralmente privadas) que você cria em uma VPC e designa para seus clusters de banco de dados. Ao usar um grupo de sub-redes de banco de dados, você pode especificar uma VPC específica ao criar clusters de banco de dados usando a AWS CLI ou a API. Se você usar o console, poderá escolher somente a VPC e os grupos de sub-redes que deseja usar.

Cada grupo de sub-redes de banco de dados deve ter sub-redes em pelo menos duas zonas de disponibilidade em determinada Região da AWS. Ao criar um cluster de banco de dados em uma VPC, escolha um grupo de sub-redes de banco de dados. Do grupo de sub-rede de banco de dados, o Amazon Aurora escolhe uma sub-rede e um endereço IP dentro dessa sub-rede para associar à

instância de banco de dados primáriaà instância de banco de dados no cluster de banco de dados. O banco de dados usa a zona de disponibilidade que contém a sub-rede. O Aurora sempre atribui um endereço IP de uma sub-rede que tem espaço de endereço IP livre.

As sub-redes em um grupo de sub-redes de banco de dados são públicas ou privadas. As sub-redes são públicas ou privadas, dependendo da configuração definida para as listas de controle de acesso à rede (ACLs de rede) e tabelas de roteamento. Para que um cluster de banco de dados seja acessível ao público, todas as sub-redes em seu grupo de sub-redes de banco de dados devem ser públicas. Se uma sub-rede associada a um cluster de banco de dados acessível ao público mudar de pública para privada, ela poderá afetar a disponibilidade do cluster de banco de dados.

Para criar um grupo de sub-redes de banco de dados que seja compatível com o modo de pilha dupla, verifique se cada sub-rede adicionada ao grupo de sub-redes de banco de dados tem um bloco CIDR do Internet Protocol versão 6 (IPv6) associado a ele. Para ter mais informações, consulte Endereçamento IP do Amazon Aurora e Migrar para IPv6 no Guia do usuário da Amazon VPC.

Quando o Amazon Aurora cria um cluster de banco de dados em uma VPC, ele atribui uma interface de rede ao seu cluster de banco de dados usando um endereço IP do seu grupo de sub-redes de banco de dados. No entanto, é altamente recomendável que você use o Sistema de Nomes de Domínio (DNS) para se conectar ao seu cluster de banco de dados. Recomendamos isso porque o endereço IP subjacente muda durante o failover.

> **Note**
>
> Para cada cluster de banco de dados executado em uma VPC, é necessário reservar pelo menos um endereço em cada sub-rede no grupo de sub-redes de banco de dados para uso pelo Amazon Aurora para ações de recuperação.

Sub-redes compartilhadas

Você pode criar um cluster de banco de dados em uma VPC compartilhada.

Algumas considerações que você deve ter em mente ao usar VPCs compartilhadas:

- Você pode mover um cluster de banco de dados de uma sub-rede de VPC compartilhada para uma sub-rede de VPC não compartilhada e vice-versa.
- Os participantes de uma VPC compartilhada devem criar um grupo de segurança na VPC para permitir que criem um cluster de banco de dados.

- Proprietários e participantes de uma VPC compartilhada podem acessar o banco de dados usando consultas SQL. No entanto, somente o criador de um recurso pode fazer chamadas de API no recurso.

Endereçamento IP do Amazon Aurora

Os endereços IP habilitam recursos na sua VPC para se comunicar com outros e com recursos na Internet. O Amazon Aurora comporta protocolos de endereçamento IPv4 e IPv6. Por padrão, o Amazon Aurora e a Amazon VPC usam o protocolo de endereçamento IPv4. Você não pode desativar esse comportamento. Ao criar uma VPC, especifique um bloco CIDR IPv4 (um intervalo de endereços IPv4 privados). Você também pode atribuir um bloco CIDR IPv6 à VPC e às sub-redes, bem como endereços IPv6 desse bloco a clusters de banco de dados em sua sub-rede.

A compatibilidade com o protocolo IPv6 expande o número de endereços IP compatíveis. Ao usar o protocolo IPv6, você tem a garantia de que terá endereços disponíveis suficientes para o crescimento futuro da Internet. Os recursos do RDS novos e existentes podem usar endereços IPv4 e IPv6 na VPC. Configurar, proteger e converter o tráfego de rede entre os dois protocolos usados em diferentes partes de uma aplicação pode causar sobrecarga operacional. Você pode padronizar o protocolo IPv6 para recursos do Amazon RDS a fim de simplificar sua configuração de rede.

Tópicos

- Endereços IPv4
- Endereços IPv6
- Modo de pilha dupla

Endereços IPv4

Quando você cria uma VPC, é necessário especificar um intervalo de endereços IPv4 para a VPC em forma de um bloco CIDR, como `10.0.0.0/16`. m grupo de sub-redes de banco de dados define o intervalo de endereços IP nesse bloco CIDR que um cluster de banco de dados pode usar. Esses endereços IP podem ser públicos ou privados.

Um endereço IPv4 privado é um endereço IP que não é acessível pela Internet. Você pode usar endereços IPv4 privados para comunicação entre o cluster de banco de dados e outros recursos, como instâncias do Amazon EC2, na mesma VPC. Cada cluster de banco de dados tem um endereço IP privado para comunicação na VPC.

Um endereço IP público é um endereço IPv4 acessível pela Internet. Você pode usar endereços públicos para comunicação entre o cluster de banco de dados e os recursos na Internet, como um cliente SQL. Você controla se o cluster de banco de dados recebe um endereço IP público.

O Amazon RDS usa endereços IPv4 elásticos públicos do grupo de endereços IPv4 públicos do EC2 para instâncias de banco de dados acessíveis ao público. Esses endereços IP são visíveis em sua conta da AWS ao usar a `describe-addresses` CLI e a API ou ao visualizar a seção IPs elásticos (EIPs) no Console de gerenciamento da AWS. Cada endereço IP gerenciado pelo RDS é marcado com um atributo `service_managed` definido como `"rds"`.

Embora esses IPs estejam visíveis em sua conta, eles continuam sendo totalmente gerenciados pelo Amazon RDS e não podem ser modificados ou liberados. O Amazon RDS libera IPs de volta ao grupo de endereços IPv4 públicos quando eles não estão mais em uso.

O CloudTrail registra em log chamadas de API relacionadas a EIPs do RDS, como `AllocateAddress`. Essas chamadas de API são invocadas pela entidade principal de serviço `rds.amazonaws.com`.

> **Note**
>
> Os IPs alocados pelo Amazon RDS não são contabilizados nos limites de EIPs da sua conta.

Para ver um tutorial que mostra como criar uma VPC somente com endereços IPv4 privados que você pode usar para um cenário comum do Amazon Aurora, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

Endereços IPv6

Como opção, você pode associar um bloco CIDR IPv6 a sua VPC e sub-redes e atribuir endereços IPv6 desse bloco a recursos em sua VPC. Todo endereço IPv6 é globalmente exclusivo.

O bloco CIDR IPv6 da VPC é automaticamente atribuído do grupo de endereços IPv6 da Amazon. Você não pode escolher o intervalo.

Ao se conectar a um endereço IPv6, verifique se as seguintes condições são atendidas:

- O cliente está configurado para que o tráfego do cliente para o banco de dados via IPv6 seja permitido.

- Os grupos de segurança do RDS usados pela instância de banco de dados estão configurados corretamente para que o tráfego do cliente para o banco de dados via IPv6 seja permitido.
- A pilha do sistema operacional do cliente permite tráfego no endereço IPv6, e os drivers e bibliotecas do sistema operacional estão configurados para escolher o endpoint de instância de banco de dados padrão correto (IPv4 ou IPv6).

Para ter mais informações sobre IPv6, consulte Endereçamento IP no Guia do usuário da Amazon VPC.

Modo de pilha dupla

Um cluster de banco de dados é executado no modo de pilha dupla quando consegue se comunicar pelos protocolos de endereçamento IPv4 e IPv6. Os recursos podem então se comunicar com o cluster de banco de dados usando IPv4, IPv6 ou ambos os protocolos. As instâncias de banco de dados privadas do modo de pilha dupla têm endpoints IPv6 aos quais o RDS concede somente acesso à VPC, garantindo que seus endpoints IPv6 permaneçam privados. As instâncias de banco de dados no modo de pilha dupla pública oferecem endpoints IPv4 e IPv6 endpoints que podem ser acessados pela internet.

Tópicos

- Modo de pilha dupla e grupos de sub-redes de banco de dados
- Trabalhar com instâncias de banco de dados de modo de pilha de dupla
- Modificar clusters de banco de dados somente IPv4 para usar o modo de pilha dupla
- Disponibilidade de clusters de banco de dados de rede de pilha dupla
- Limitações para clusters de banco de dados de rede de pilha dupla

Para ver um tutorial que mostra como criar uma VPC com endereços IPv4 e IPv6 que você pode usar para um cenário comum do Amazon Aurora, consulte Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla).

Modo de pilha dupla e grupos de sub-redes de banco de dados

Para usar o modo de pilha dupla, verifique se cada sub-rede no grupo de sub-redes de banco de dados que você associa ao cluster de banco de dados tem um bloco CIDR IPv6 associado a ela. Você pode criar um grupo de sub-redes de banco de dados ou modificar um existente para atender a esse requisito. Depois que um cluster de banco de dados entra no modo de pilha dupla, os clientes

podem se conectar normalmente. Os firewalls de segurança do cliente e os grupos de segurança de instâncias de banco de dados do RDS devem ser configurados com precisão para permitir tráfego por IPv6. Para se conectar, os clientes usam o endpoint da instância primária do cluster de banco de dados. As aplicações cliente podem especificar qual protocolo é o preferencial ao se conectar a um banco de dados. No modo de pilha dupla, o cluster de banco de dados detecta o protocolo de rede de preferência do cliente, IPv4 ou IPv6, e o usa para a conexão.

Se um grupo de sub-redes de banco de dados deixar de ser compatível com o modo de pilha dupla devido à exclusão de sub-rede ou desassociação do CIDR, existe o risco de um estado de rede incompatível para instâncias de banco de dados associadas ao grupo de sub-redes de banco de dados. Além disso, não é possível usar o grupo de sub-redes de banco de dados ao criar um novo cluster de banco de dados do modo de pilha dupla.

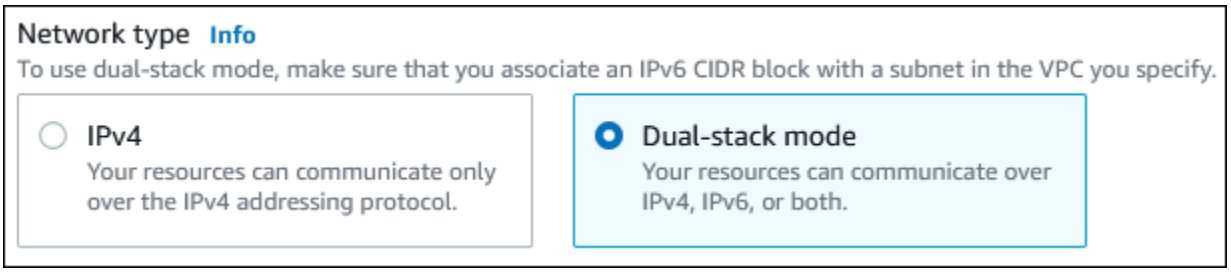
Para determinar se um grupo de sub-redes de banco de dados é compatível com o modo de pilha dupla usando o Console de gerenciamento da AWS, visualize o Network type (Tipo de rede) na página de detalhes do grupo de sub-redes de banco de dados. Para determinar se um grupo de sub-redes de banco de dados comporta o modo de pilha dupla usando a AWS CLI, execute o comando describe-db-subnet-groups e visualize `SupportedNetworkTypes` na saída.

As réplicas de leitura são tratadas como instâncias de banco de dados independentes e podem ter um tipo de rede diferente da instância de banco de dados primária. Se você alterar o tipo de rede da instância de banco de dados primária de uma réplica de leitura, esta não será afetada. Você pode restaurar uma instância de banco de dados para qualquer tipo de rede compatível.

Trabalhar com instâncias de banco de dados de modo de pilha de dupla

Ao criar ou modificar um cluster de banco de dados, você pode especificar o modo de pilha dupla para permitir que os recursos se comuniquem com o cluster de banco de dados por IPv4, IPv6 ou ambos.

Ao usar o Console de gerenciamento da AWS para criar ou modificar uma instância de banco de dados, você pode especificar o modo de pilha dupla na seção Network type (Tipo de rede). A imagem a seguir mostra a seção Network type (Tipo de rede) no console.



Ao usar a AWS CLI para criar ou modificar um cluster de banco de dados, defina a opção `--network-type` como `DUAL` para usar o modo de pilha dupla. Ao usar a API do RDS para criar ou modificar um cluster de banco de dados, defina o parâmetro `NetworkType` como `DUAL` para usar o modo de pilha dupla. Quando você estiver modificando o tipo de rede de uma instância de banco de dados, é possível que ocorra tempo de inatividade. Se o modo de pilha dupla não for compatível com a versão do mecanismo de banco de dados especificado nem com o grupo de sub-redes de banco de dados, o erro `NetworkTypeNotSupported` será retornado.

Para ter mais informações sobre como criar um cluster de banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Para determinar se um cluster de banco de dados está no modo de pilha dupla usando o console, visualize o Network type (Tipo de rede) na guia Connectivity & security (Conectividade e segurança) para o cluster de banco de dados.

Modificar clusters de banco de dados somente IPv4 para usar o modo de pilha dupla

Você pode modificar clusters de banco de dados somente IPv4 para usar o modo de pilha dupla. Para isso, altere o tipo de rede do cluster de banco de dados. A modificação pode ocasionar tempo de inatividade.

É recomendável que você altere o tipo de rede dos clusters do Amazon Aurora durante uma janela de manutenção. No momento, não é possível definir o tipo de rede de novas instâncias para o modo de pilha dupla. É possível definir o tipo de rede manualmente usando o comando `modify-db-cluster`.

Antes de modificar um cluster de banco de dados para usar o modo de pilha dupla, verifique se seu grupo de sub-redes de banco de dados é compatível com o modo de pilha dupla. Se o grupo de sub-redes de banco de dados associado ao cluster de banco de dados não for compatível com o modo de pilha dupla, ao modificar o cluster de banco de dados especifique um grupo de sub-redes de banco de dados diferente que seja compatível. Modificar o grupo de sub-redes de banco de dados de um cluster de banco de dados pode ocasionar tempo de inatividade.

Se você modificar o grupo de sub-redes de banco de dados de um cluster de banco de dados antes de alterar o cluster de banco de dados para usar o modo de pilha dupla, verifique se o grupo de sub-redes de banco de dados é válido para o cluster de banco de dados antes e depois da alteração.

Recomendamos que você execute a API modify-db-cluster apenas com o parâmetro `--network-type` com valor `DUAL` para alterar a rede de um cluster do Amazon Aurora para o modo de pilha

dupla. Adicionar outros parâmetros com o parâmetro `--network-type` na mesma chamada de API pode ocasionar tempo de inatividade.

Se você não conseguir se conectar ao cluster de banco de dados após a alteração, verifique se os firewalls de segurança do cliente e do banco de dados e as tabelas de rotas estão configurados com precisão para permitir tráfego para o banco de dados na rede selecionada (IPv4 ou IPv6). Você também pode precisar modificar parâmetros, bibliotecas ou drivers do sistema operacional para se conectar por meio de um endereço IPv6.

Como modificar clusters de banco de dados somente IPv4 para usar o modo de pilha dupla

1. Modifique um grupo de sub-redes de banco de dados para ser compatível com o modo de pilha dupla ou crie um grupo de sub-redes de banco de dados que seja compatível com esse modo:

   a. Associe um bloco CIDR IPv6 à VPC.

      Para obter instruções, consulte Adicionar um bloco CIDR IPv6 a sua VPC no Manual do usuário do Amazon VPC.

   b. Anexe o bloco CIDR IPv6 a todas as sub-redes do grupo de sub-redes do banco de dados.

      Para obter instruções, consulte Adicionar um bloco CIDR IPv6 a sua sub-rede no Manual do usuário do Amazon VPC.

   c. Verifique se o grupo de sub-redes de banco de dados é compatível com o modo de pilha dupla.

      Se você estiver usando o Console de gerenciamento da AWS, selecione o grupo de sub-redes de banco de dados e verifique se o valor Supported network types (Tipos de rede compatíveis) é Dual, IPv4 (Duplo, IPv4).

      Se estiver usando a AWS CLI, execute o comando describe-db-subnet-groups e verifique se o valor `SupportedNetworkType` para a instância de banco de dados é `Dual, IPv4`.

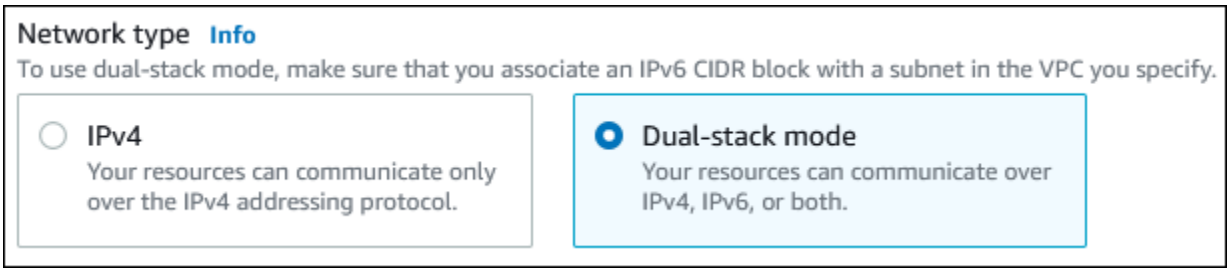
2. Modifique o grupo de segurança associado ao cluster de banco de dados para permitir conexões IPv6 com o banco de dados ou crie um grupo de segurança que permita conexões IPv6.

   Para obter instruções, consulte Regras do grupo de segurança no Guia do usuário da Amazon VPC.

3. Modifique o cluster de banco de dados para oferecer suporte ao modo de pilha dupla. Para fazer isso, defina a opção Network type (Tipo de rede) como Dual-stack mode (Modo de pilha dupla).

Se você estiver usando o console, verifique se as seguintes configurações estão corretas:

- Network type (Tipo de rede): Dual-stack mode (Modo de pilha dupla).



- DB subnet group (Grupo de sub-redes do banco de dados): o grupo de sub-redes de banco de dados que você configurou em uma etapa anterior
- Security group (Grupo de segurança): o grupo de segurança que você configurou em uma etapa anterior.

Se você estiver usando a AWS CLI, verifique se as seguintes configurações estão corretas:

- `--network-type` – `dual`
- `--db-subnet-group-name`: o grupo de sub-redes de banco de dados que você configurou em uma etapa anterior
- `--vpc-security-group-ids`: o grupo de segurança da VPC que você configurou em uma etapa anterior.

Por exemplo:

```
aws rds modify-db-cluster --db-cluster-identifier my-cluster --network-type "DUAL"
```

4. Verifique se o cluster de banco de dados é compatível com o modo de pilha dupla.

   Se estiver usando o console, selecione a guia Configuration (Configuração) para o cluster de banco de dados. Nessa guia, verifique se o valor de Network type (Tipo de rede) é Dual-stack mode (Modo de pilha dupla).

   Se estiver usando a AWS CLI, execute o comando describe-db-clusters e verifique se o valor `NetworkType` para o cluster de banco de dados é `dual`.

Execute o comando `dig` no endpoint da instância de banco de dados gravadora para identificar o endereço IPv6 associado a ele.

```
dig db-instance-endpoint AAAA
```

Use o endpoint da instância de banco de dados de gravador, não o endereço IPv6, para se conectar ao cluster de banco de dados.

Disponibilidade de clusters de banco de dados de rede de pilha dupla

Os clusters de banco de dados de rede de pilha dupla estão disponíveis em todas as Regiões da AWS, exceto nas seguintes:

- Ásia-Pacífico (Hyderabad)
- Ásia-Pacífico (Malásia)
- Ásia-Pacífico (Melbourne)
- Ásia-Pacífico (Tailândia)
- Oeste do Canadá (Calgary)
- Europa (Espanha)
- Europa (Zurique)
- Israel (Tel Aviv)
- México (Centro)
- Oriente Médio (Emirados Árabes Unidos)

As seguintes versões do mecanismo de banco de dados são compatíveis com clusters de banco de dados de rede de pilha dupla:

- Versões do Aurora MySQL:
  - Versão 3.02 e versões 3 posteriores
  - Versão 2.09.1 e versões 2 posteriores

  Para ter mais informações sobre as versões do Aurora MySQL, consulte as "Release Notes for Aurora MySQL" (Notas de lançamento do Aurora MySQL).
- Versões do Aurora PostgreSQL:

- 15.2 e todas as versões posteriores
- Versão 14.3 e versões 14 posteriores
- Versão 13.7 e versões 13 posteriores

Para ter mais informações sobre as versões do Aurora PostgreSQL, consulte as "Release Notes for Aurora PostgreeSQL" (Notas de lançamento do Aurora PostgreSQL).

Limitações para clusters de banco de dados de rede de pilha dupla

As seguintes limitações se aplicam aos clusters de banco de dados de rede de pilha dupla:

- Clusters de banco de dados não podem usar o protocolo IPv6 exclusivamente. É possível usar IPv4 exclusivamente ou o protocolo IPv4 e IPv6 (modo de pilha dupla).
- O Amazon RDS não é compatível com sub-redes IPv6 nativas.
- Você não pode usar o RDS Proxy com clusters de banco de dados do modo de pilha dupla.

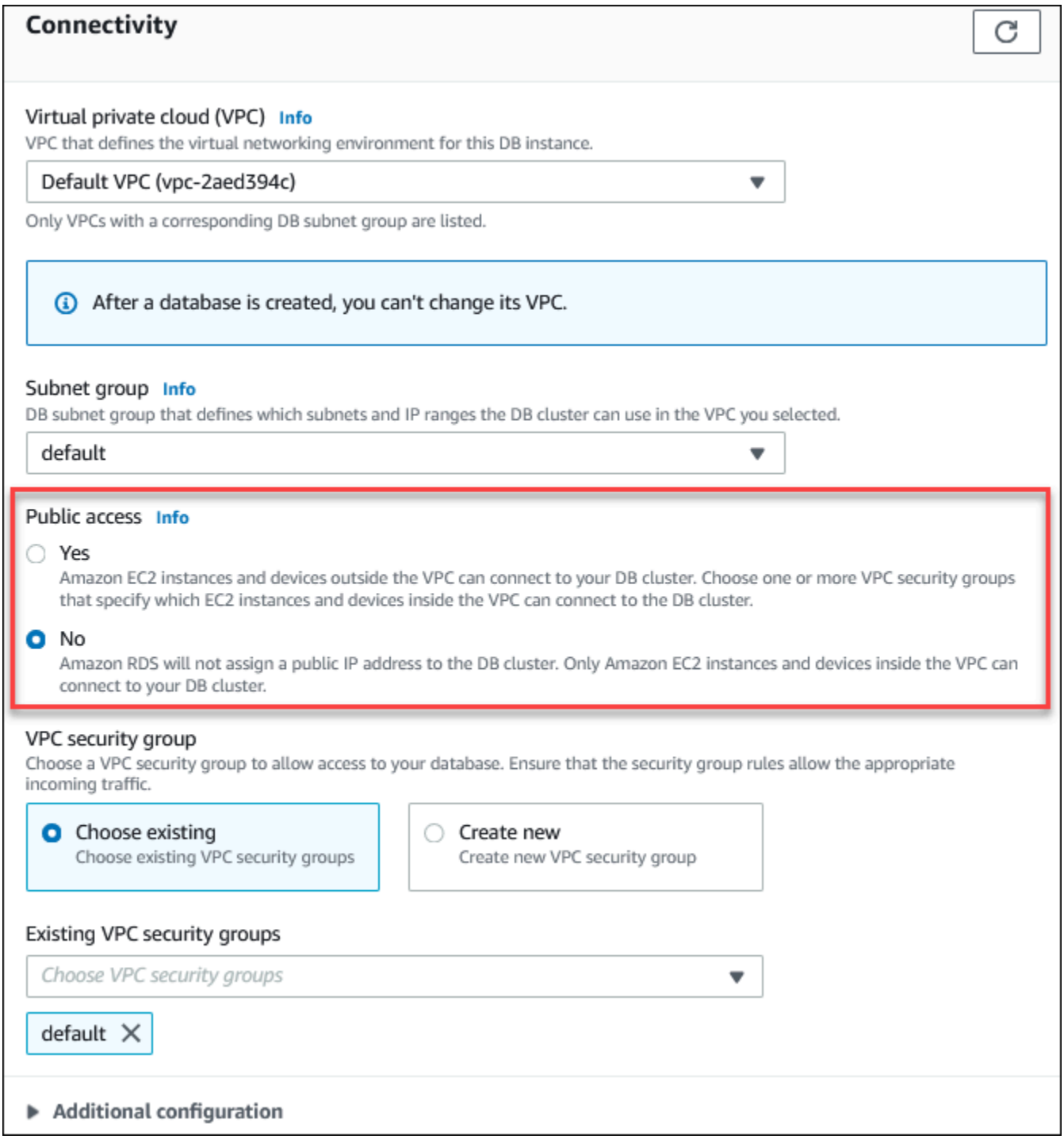
Ocultar um cluster de banco de dados em uma VPC da Internet

Um cenário comum do Amazon Aurora é ter uma VPC na qual exista uma instância do Amazon EC2 com uma aplicação web voltada para o público e um cluster de banco de dados com um banco de dados não acessível ao público. Por exemplo, você pode criar uma VPC que tenha uma sub-rede pública e uma sub-rede privada. As instâncias do EC2 que funcionam como servidores Web podem ser implantadas na sub-rede pública. os clusters de banco de dados são implantados na sub-rede privada. Nessa implantação, apenas os servidores Web têm acesso aos clusters de banco de dados. Para obter uma ilustração desse cenário, consulte Um cluster de banco de dados em uma VPC acessada por uma instância do Amazon EC2 na mesma VPC..

Quando você inicia um cluster de banco de dados dentro de uma VPC, o cluster de banco de dados tem um endereço IP privado para tráfego dentro da VPC. Esse endereço IP privado não é acessível ao público geral. Você pode usar a opção Public access (Acesso público) para designar se o cluster de banco de dados também deve ter um endereço IP público além do endereço IP privado. Se o cluster de banco de dados for acessível ao público, seu endpoint DNS resolverá para o endereço IP privado de dentro da VPC. Ele é resolvido para o endereço IP público de fora da VPC. O acesso ao cluster de banco de dados é controlado em última análise pelo grupo de segurança usado. Esse acesso público não será permitido se o grupo de segurança atribuído ao cluster de banco de dados não incluir regras de entrada que permitam isso. Além disso, para que um cluster de banco de dados seja acessível publicamente, as sub-redes no grupo de sub-redes de banco de dados devem ter um

gateway da Internet. Para ter mais informações, consulte Não é possível conectar-se à instância de banco de dados do Amazon RDS.

Você pode modificar um cluster de banco de dados para ativar ou desativar a acessibilidade ao público geral modificando a opção Public access (Acesso público). A ilustração a seguir mostra a opção Public access (Acesso público) na seção Additional connectivity configuration (Configuração de conectividade adicional). Para definir a opção, abra a seção Additional connectivity configuration (Configuração de conectividade adicional) na seção Connectivity (Conectividade).



Para obter informações sobre como modificar uma instância de banco de dados para definir a opção de Public access (Acesso público), consulte Modificar uma instância de banco de dados em um cluster de banco de dados.

Criar um cluster de banco de dados em uma VPC

Os procedimentos a seguir ajudam você a criar um cluster de banco de dados em uma VPC. Para usar a VPC padrão, você pode começar na etapa 2 e usar a VPC e o grupo de sub-redes de banco de dados que já foram criados para você. Se quiser criar uma VPC adicional, você poderá criar uma nova VPC.

> **Note**
>
> Se você deseja que seu cluster de banco de dados na VPC seja acessível ao público geral, atualize as informações de DNS da VPC, habilitando os atributos DNS hostnames (Nomes de host de DNS) e DNS resolution (Resolução de DNS). Para obter informações sobre como atualizar as informações de DNS para uma instância de VPC, consulte Atualização do suporte a DNS para sua VPC.

Siga estas etapas para criar uma instância de banco de dados em uma VPC:

- Etapa 1: Criar uma VPC
- Etapa 2: Criar um grupo de sub-redes de banco de dados
- Etapa 3: Criar um grupo de segurança da VPC
- Etapa 4: Criar uma instância de banco de dados na VPC

Etapa 1: Criar uma VPC

Crie uma VPC com duas sub-redes em pelo menos duas zonas de disponibilidade. Você usará essas sub-redes ao criar um grupo de sub-redes de banco de dados. Se você tiver uma VPC padrão, uma sub-rede será criada automaticamente para você em cada zona de disponibilidade na Região da AWS.

Para ter mais informações, consulte Criar uma VPC com sub-redes públicas e privadas ou Criar uma VPC no Guia do usuário da Amazon VPC.

Etapa 2: Criar um grupo de sub-redes de banco de dados

Um grupo de sub-redes de banco de dados é uma coleção de sub-redes (geralmente privadas) que você cria para uma VPC e designa aos seus clusters de banco de dados. Um grupo de sub-redes

de banco de dados permite que você especifique uma VPC particular ao criar clusters de banco de dados usando a AWS CLI ou a API do RDS. Se você usar o console, poderá escolher somente a VPC e as sub-redes que deseja usar. Cada grupo de sub-redes de banco de dados deve ter pelo menos uma sub-rede em pelo menos duas zonas de disponibilidade na Região da AWS. Como prática recomendada, cada grupo de sub-redes de banco de dados deve ter no mínimo uma sub-rede para cada zona de disponibilidade na Região da AWS.

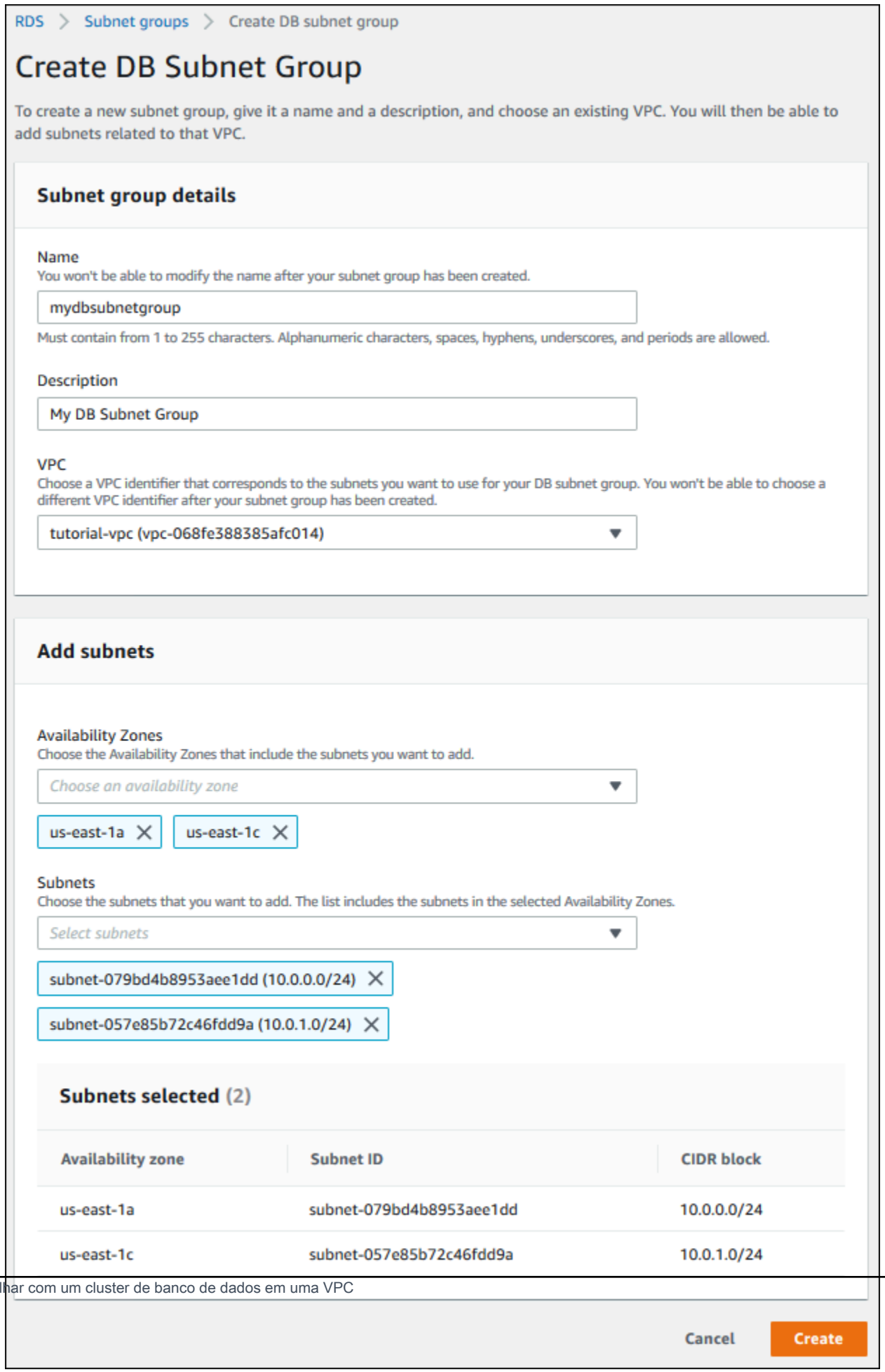
Para que um cluster de banco de dados seja acessível publicamente, as sub-redes no grupo de sub-redes de banco de dados devem ter um gateway da Internet. Para ter mais informações sobre gateways da Internet para sub-redes, consulte “Estabelecer conexão com a Internet usando um gateway da Internet no Manual do usuário da Amazon VPC.

Ao criar um cluster de banco de dados em uma VPC, você pode escolher um grupo de sub-redes de banco de dados. O Amazon Aurora escolhe uma sub-rede e um endereço IP dentro dessa sub-rede para associar ao cluster de banco de dados. Se não houver nenhum grupo de sub-redes de banco de dados, o Amazon Aurora criará um grupo de sub-redes padrão quando você criar um cluster de banco de dados. O Amazon Aurora cria e associa uma interface de rede elástica ao seu cluster de banco de dados com esse endereço IP. O cluster de banco de dados usa a zona de disponibilidade que contém a sub-rede.

Nesta etapa, você criará um grupo de sub-redes de banco de dados e adicionará as sub-redes criadas à sua VPC.

Como criar um grupo de sub-redes de banco de dados

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Subnet groups (Grupos de sub-redes).
3. Escolha Create DB Subnet Group (Criar grupo de sub-redes de banco de dados).
4. Em Name (Nome), digite o nome do grupo de sub-redes de banco de dados.
5. Em Description (Descrição), digite uma descrição para o grupo de sub-redes de banco de dados.
6. Para VPC, escolha a VPC padrão ou a VPC criada por você.
7. Na seção Adicionar sub-redes, escolha as zonas de disponibilidade que incluem as sub-redes de Zonas de disponibilidade e escolha as sub-redes de Sub-redes.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

8. Escolha Create (Criar).

   Seu novo grupo aparece na lista de grupos de sub-redes de banco de dados no console do RDS. Você pode selecionar o grupo de sub-redes de banco de dados para obter detalhes, incluindo todas as sub-redes associadas a esse grupo, no painel de detalhes, na parte inferior da janela.

Etapa 3: Criar um grupo de segurança da VPC

Antes de criar um cluster de banco de dados, você pode criar um grupo de segurança da VPC para associar a esse cluster. Se você não criar um grupo de segurança da VPC, poderá usar o grupo de segurança padrão ao criar um cluster de banco de dados. Para obter instruções sobre como criar um grupo de segurança para o cluster de banco de dados, consulte Criar um grupo de segurança da VPC para um cluster de banco de dados privada ou Controle o tráfego para recursos usando grupos de segurança no Guia do usuário da Amazon VPC.

Etapa 4: Criar uma instância de banco de dados na VPC

Nessa etapa, você cria um cluster de banco de dados e usa o nome da VPC, o grupo de sub-redes de banco de dados e o grupo de segurança de VPC que você criou nas etapas anteriores.

> **Note**
>
> Se quiser que seu cluster de banco de dados na VPC seja acessível ao público geral, você deverá habilitar os atributos DNS hostnames (Nomes de host de DNS) e DNS resolution (Resolução de DNS). Para ter mais informações, consulte Atributos de DNS para sua VPC no Guia do usuário da Amazon VPC.

Para obter detalhes sobre como criar um cluster de banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora.

Quando solicitado na seção Connectivity (Conectividade), insira o nome da VPC, o grupo de sub-redes de banco de dados e o grupo de segurança da VPC.

> **Note**
>
> A atualização de VPCs não é compatível com clusters do Aurora no momento.

Cenários para acessar um cluster de banco de dados em uma VPC

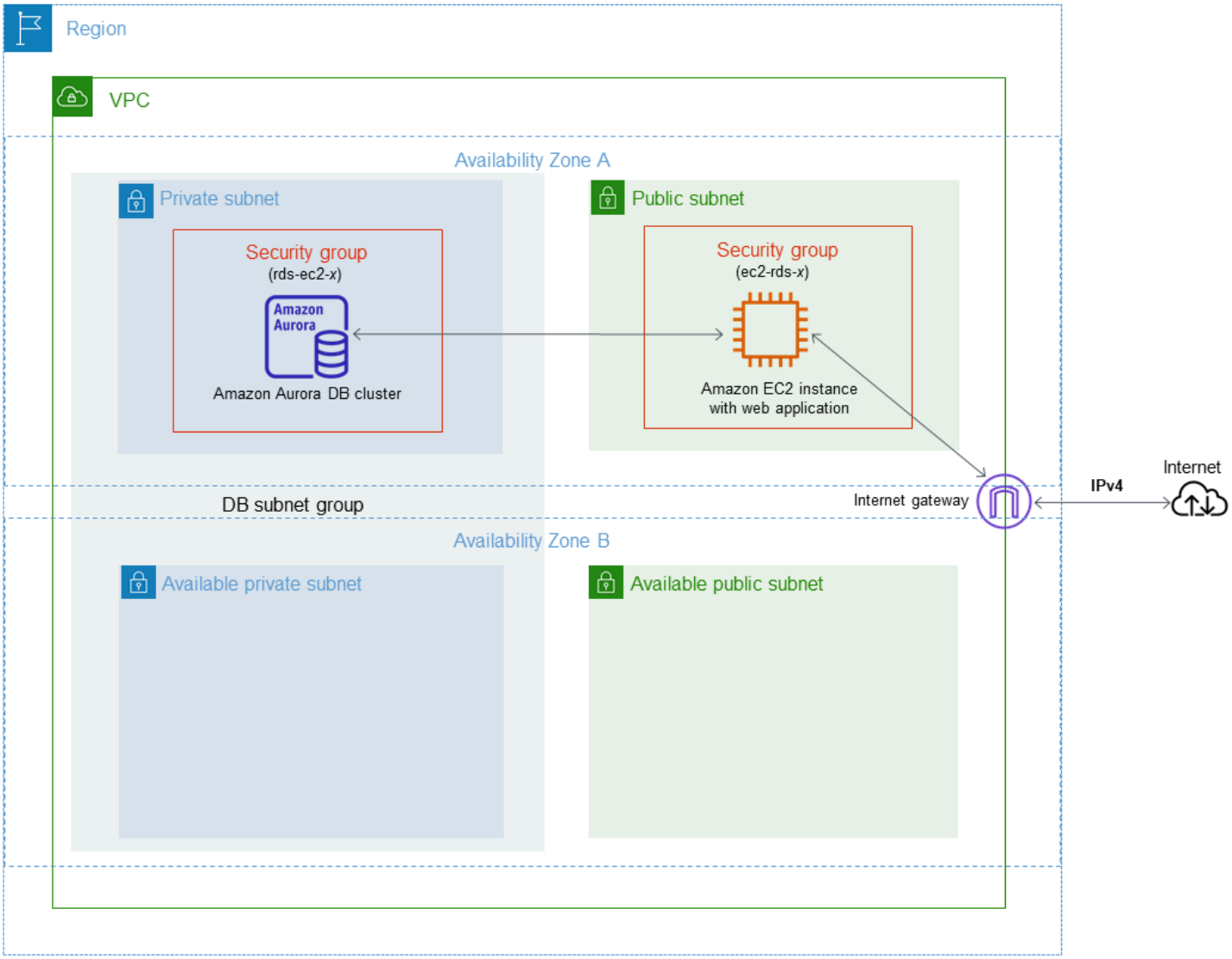
O Amazon Aurora é compatível com os seguintes cenários de acesso a um cluster de banco de dados em uma VPC:

- Uma instância do Amazon EC2 na mesma VPC
- Uma instância do EC2 em uma VPC diferente
- Uma aplicação cliente pela Internet
- Uma rede privada

Um cluster de banco de dados em uma VPC acessada por uma instância do Amazon EC2 na mesma VPC.

Um uso comum de um cluster de banco de dados em uma VPC é compartilhar dados com um servidor de aplicações que está sendo executado em uma instância do Amazon EC2 na mesma VPC.

O diagrama a seguir mostra esse cenário.



A maneira mais simples de gerenciar o acesso entre instâncias do EC2 e clusters de banco de dados na mesma VPC é fazer o seguinte:

- Crie um grupo de segurança de VPC no qual os seus clusters de banco de dados estarão. Esse grupo de segurança pode ser usado para restringir o acesso aos clusters de banco de dados. Por exemplo, você pode criar uma regra personalizada para esse grupo de segurança. Isso pode permitir o acesso TCP usando a porta que você atribuiu ao cluster de banco de dados quando a criou e um endereço IP para acessar o cluster de banco de dados para desenvolvimento ou outros fins.
- Crie um grupo de segurança da VPC em que as suas instâncias do EC2 (servidores Web e clientes) estarão. Esse grupo de segurança pode, se necessário, permitir o acesso à instância do

EC2 pela Internet usando a tabela de roteamento da VPC. Por exemplo, você pode definir regras nesse grupo de segurança para permitir o acesso TCP à instância do EC2 pela porta 22.

- Crie regras personalizadas no grupo de segurança para seus clusters de banco de dados que permitam conexões do grupo de segurança que você criou para suas instâncias do EC2. Essas regras podem permitir que qualquer membro do grupo de segurança acesse os clusters de banco de dados.

Há uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Um grupo de sub-redes do banco de dados RDS exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a migração para uma implantação de instância de banco de dados multi-AZ no futuro.

Para um tutorial que mostra como criar uma VPC com sub-redes públicas e privadas para esse cenário, consulte Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4).

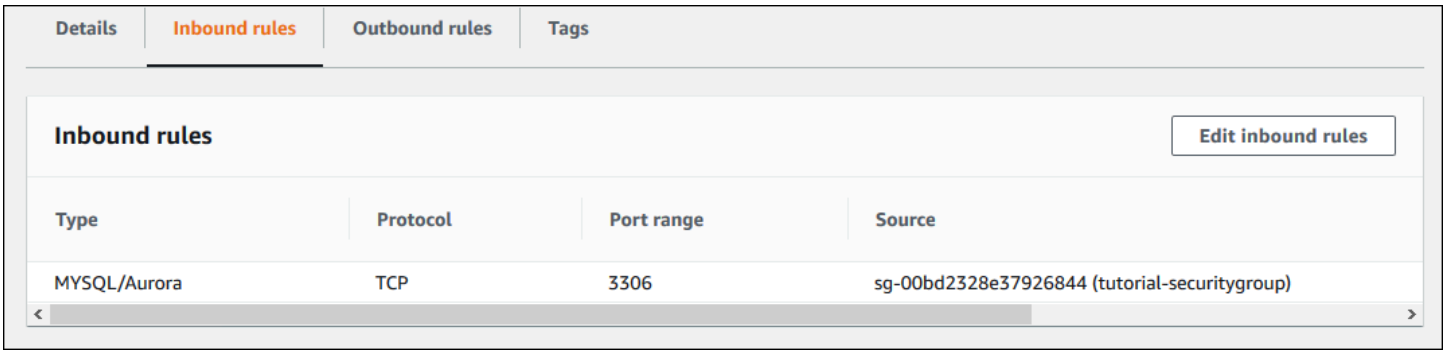
> **Tip**
>
> Você pode configurar a conectividade de rede entre uma instância do Amazon EC2 e um cluster de banco de dados automaticamente ao criar o cluster de banco de dados. Para obter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2 .

Para criar uma regra em um grupo de segurança de VPC que permita conexões de outro grupo de segurança, faça o seguinte:

1. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. No painel de navegação, selecione Grupos de segurança.
3. Escolha ou crie um grupo de segurança ao qual você deseja conceder acesso para membros de outro grupo de segurança. No cenário anterior, esse é o grupo de segurança usado para seus clusters de banco de dados. Vá para a guia Inbound Rules (Regras de entrada) e escolha Edit rules (Editar regras).
4. Na página Edit inbound rules (Editar regras de entrada), escolha Add Rule (Adicionar regra).
5. Em Type (Tipo), escolha a entrada que corresponde à porta usada ao criar seu cluster de banco de dados, como MYSQL/Aurora.

6. Na caixa Origem, comece a digitar o ID do grupo de segurança para listar os grupos de segurança correspondentes. Escolha o grupo de segurança com membros que você deseja que tenham acesso aos recursos protegidos por esse grupo de segurança. Esse é o grupo de segurança usado para sua instância do EC2 no cenário anterior.
7. Se necessário, repita as etapas para o protocolo TCP, criando uma regra com Todos os TCP como Tipo e seu grupo de segurança na caixa Origem. Se você pretende usar o protocolo UDP, crie uma regra com All UDP (Todos os UDP) como Type (Tipo) e seu grupo de segurança em Source (Origem).
8. Selecione Salvar regras.

A tela a seguir mostra uma regra de entrada com um grupo de segurança como origem.

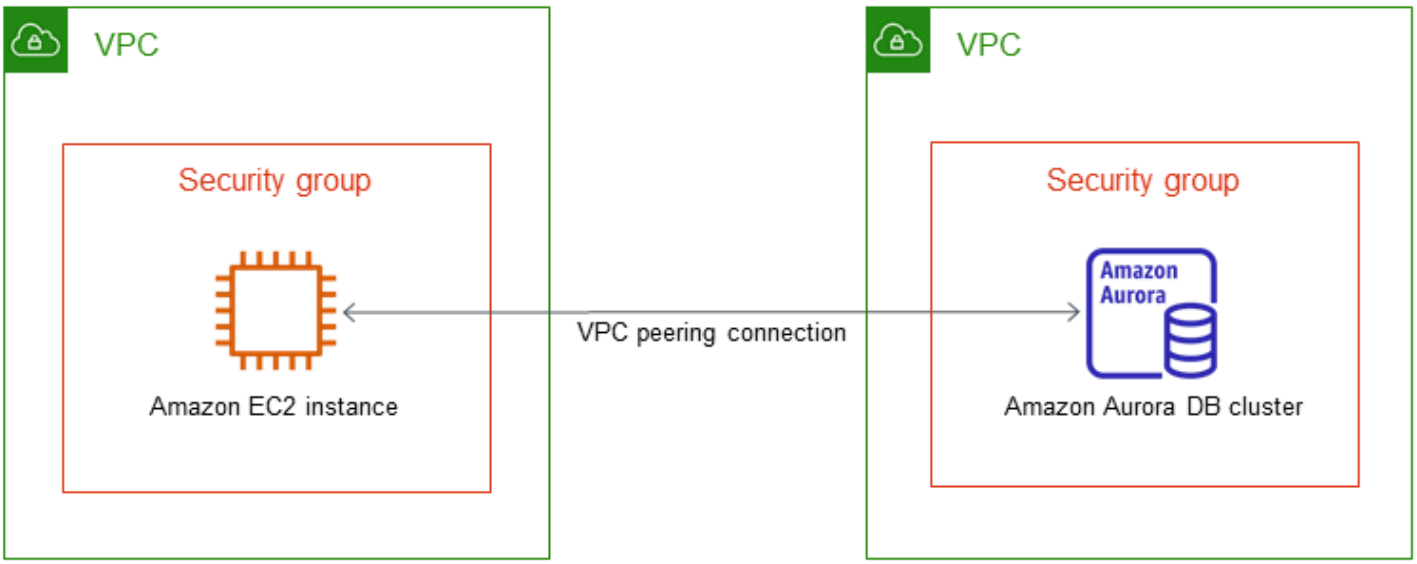


Para ter mais informações sobre como se conectar ao cluster de banco de dados por meio da instância do EC2, consulte Como conectar-se a um cluster de bancos de dados Amazon Aurora.

Um cluster de banco de dados em uma VPC acessada por uma instância do EC2 em uma VPC diferente

Quando seu cluster de banco de dados está em uma VPC diferente da instância do EC2 que você está usando para acessá-la, você pode usar emparelhamento de VPC para acessar o cluster de banco de dados.

O diagrama a seguir mostra esse cenário.

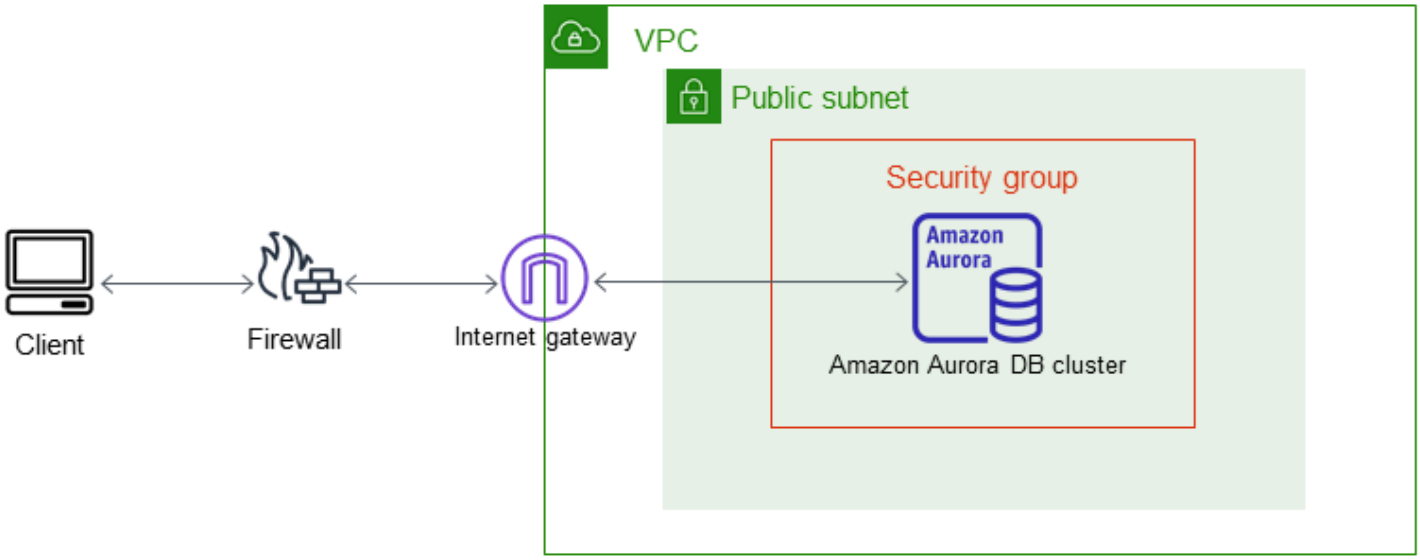


Uma conexão de emparelhamento da VPC é uma conexão de redes entre duas VPCs que permite direcionar o tráfego entre elas usando endereços IP privados. Recursos em qualquer VPC podem se comunicar uns com os outros como se estivessem na mesma rede. É possível criar uma conexão de emparelhamento da VPC entre suas próprias VPCs, com uma VPC em outra conta da AWS ou com uma VPC em uma Região da AWS diferente. Para saber mais sobre o emparelhamento de VPCs, consulte Emparelhamento de VPCs, no Guia do usuário do Amazon Virtual Private Cloud.

Um cluster de banco de dados em uma VPC acessada por uma aplicação cliente pela internet

Para acessar um cluster de banco de dados em uma VPC de uma aplicação cliente via Internet, configure uma VPC com uma sub-rede pública única e um gateway da Internet para permitir a comunicação pela Internet.

O diagrama a seguir mostra esse cenário.



Recomendamos a seguinte configuração:

- Uma VPC de tamanho /16 (por exemplo, CIDR: 10.0.0.0/16). Esse tamanho fornece 65.536 endereços IP privados.
- Uma sub-rede de tamanho /24 (por exemplo CIDR: 10.0.0.0/24). Esse tamanho fornece 256 endereços IP privados.
- Um cluster associado do Amazon Aurora à VPC e à sub-rede. O Amazon RDS atribui um endereço IP da sub-rede ao seu cluster de banco de dados.
- Um gateway da Internet que conecta a VPC à Internet e a outros produtos da AWS.
- Um grupo de segurança associado com o cluster de banco de dados. As regras de entrada de grupo de segurança permitem que sua aplicação cliente acesse o seu cluster de banco de dados.

Para ter mais informações sobre como criar um cluster de banco de dados em uma VPC, consulte Criar um cluster de banco de dados em uma VPC.

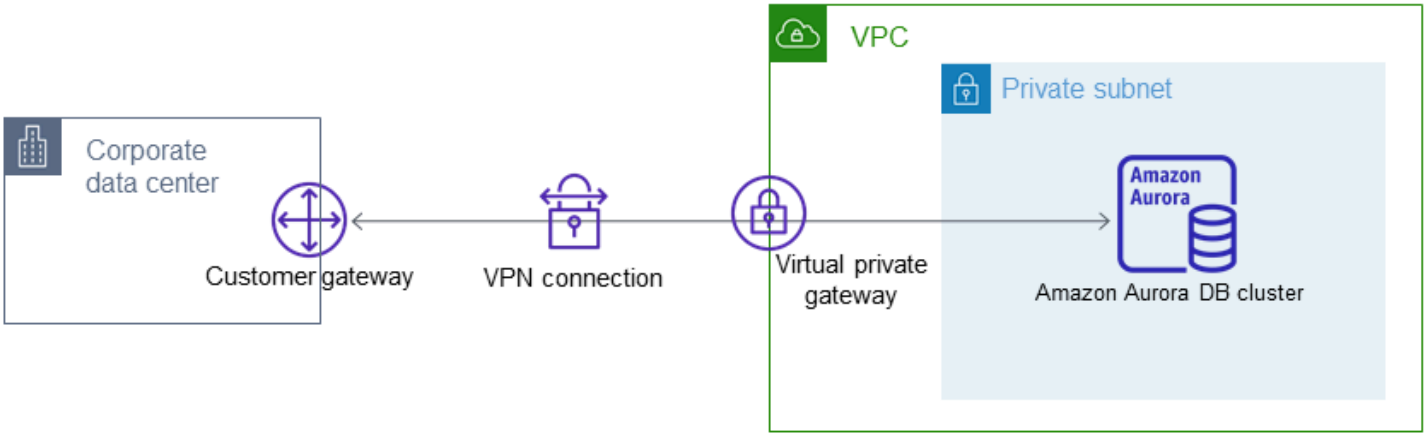
Um cluster de banco de dados em uma VPC acessado por uma rede privada

Se seu cluster de banco de dados não estiver acessível publicamente, você terá as seguintes opções para acessá-la a partir de uma rede privada:

- Uma conexão AWS Site-to-Site VPN. Para ter mais informações, consulte O que é o AWS Site-to-Site VPN?
- Uma conexão Direct Connect. Para obter mais informações, consulte O que é o Direct Connect?

- Uma conexão AWS Client VPN. Para ter mais informações, consulte O que é o AWS Client VPN?

O diagrama a seguir mostra um cenário com uma conexão AWS Site-to-Site VPN.

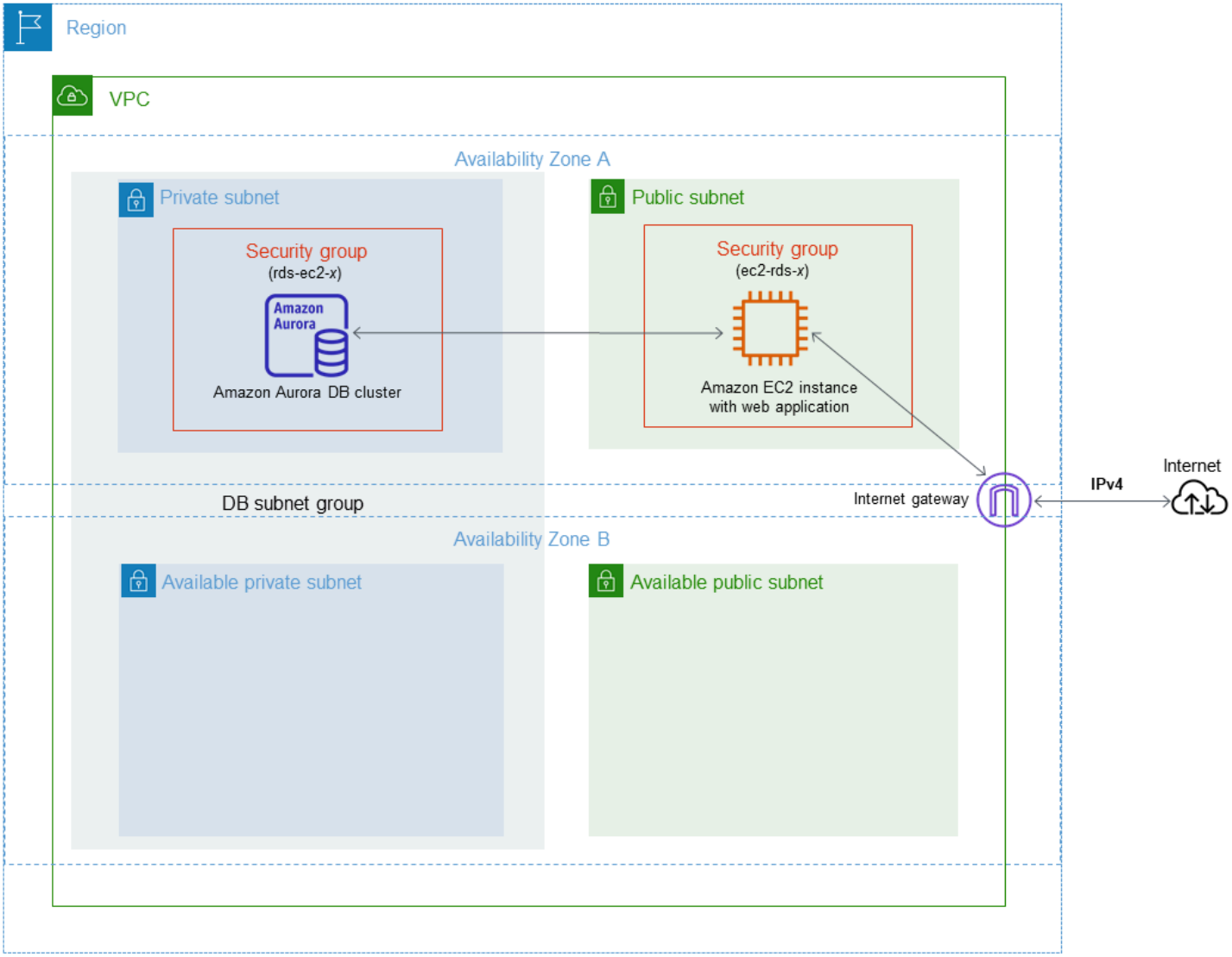


Para obter mais informações, consulte Privacidade do tráfego entre redes.

Tutorial: Criar uma VPC para usar com um cluster de banco de dados (somente IPv4)

Um cenário comum inclui um cluster de banco de dados em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Essa VPC compartilha dados com um servidor Web em execução na mesma VPC. Neste tutorial, você cria a VPC para esse cenário.

O diagrama a seguir mostra esse cenário. Para obter informações sobre outros cenários, consulte Cenários para acessar um cluster de banco de dados em uma VPC.



Seu cluster de banco de dados só precisa estar disponível para seu servidor Web e não para a Internet pública. Assim, você cria uma VPC com sub-redes públicas e privadas. O servidor Web está hospedado na sub-rede pública, para que ele possa chegar à Internet pública. O cluster de banco

de dados está hospedado em uma sub-rede privada. O servidor Web pode se conectar ao cluster de banco de dados porque ela está hospedado na mesma VPC. Mas o cluster de banco de dados não está disponível para a Internet pública, o que oferece maior segurança.

Esse tutorial configura uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Essas sub-redes não são usadas no tutorial. Um grupo de sub-redes do banco de dados RDS exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a configuração de mais de uma instância de banco de dados Aurora.

Este tutorial descreve como configurar uma VPC para clusters de bancos de dados Amazon Aurora. Para obter um tutorial que mostra como criar um servidor Web para esse cenário de VPC, consulte Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora. Para obter mais informações sobre uma Amazon VPC, consulte o Guia de conceitos básicos da Amazon VPC e Guia do usuário da Amazon VPC.

> **Tip**
>
> Você pode configurar a conectividade de rede entre uma instância do Amazon EC2 e um cluster de banco de dados automaticamente ao criar o cluster de banco de dados. A configuração da rede é semelhante à descrita neste tutorial. Para obter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2.

Criar uma VPC com sub-redes públicas e privadas

Use o seguinte procedimento para criar uma VPC com sub-redes públicas e privadas.

Para criar uma VPC e sub-redes

1. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região na qual deseja criar sua VPC. Este exemplo usa a região Oeste dos EUA (Oregon).
3. No canto superior esquerdo, escolha VPC dashboard (Painel da VPC). Para começar a criar uma VPC, escolha Create VPC (Criar VPC).
4. Para Resources to create (Recursos a serem criados) em VPC settings (Configurações da VPC), escolha VPC and more (VPC e mais).
5. Para VPC settings (Configurações da VPC), defina os seguintes valores:

- Name tag auto-generation (Geração automática da etiqueta de nome): **`tutorial`**
- IPv4 CIDR block (Bloco CIDR IPv4): **`10.0.0.0/16`**
- IPv6 CIDR block (Bloco CIDR IPv6): No IPv6 CIDR Block (Nenhum bloco CIDR IPv6)
- Tenancy (Locação): Default (Padrão)
- Number of Availability Zones (AZs) [Número de zonas de disponibilidade (AZs)]: 2
- Customize AZs (Personalizar AZs): mantenha os valores padrão.
- Number of public subnet (Número de sub-redes públicas):2
- Number of private subnets (Número de sub-redes privadas): 2
- Customize subnets CIDR blocks (Personalizar blocos CIDR de sub-redes): mantenha os valores padrão.
- NAT gateways ($) (Gateways NAT ($)): None (Nenhum)
- VPC endpoints (Endpoints da VPC): None (Nenhum)
- DNS options (Opções de DNS): mantenha os valores padrão.

6. Escolha Criar VPC.

Criar um grupo de segurança de VPC para um servidor Web público

Em seguida, você criará um grupo de segurança para acesso público. Para se conectar a instâncias públicas do EC2 em sua VPC, adicione regras de entrada ao seu grupo de segurança de VPC. Isso permite que o tráfego se conecte pela Internet.

Para criar um grupo de segurança de VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Security group name (Nome do grupo de segurança: **`tutorial-securitygroup`**
   - Descrição: **`Tutorial Security Group`**
   - VPC: escolha a VPC criada na etapa anterior; por exemplo, vpc-***`identifier`*** (tutorial-vpc)
4. Adicione regras de entrada ao grupo de segurança.

a. Determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, é possível usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP: `203.0.113.25/32`.

   Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Se sim, especifique o intervalo de endereços IP utilizado por computadores cliente.

   > ⚠ Warning
   >
   > Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias usando SSH.

b. Na seção Regras de entrada, escolha Adicionar regra.

c. Defina os valores a seguir para a sua nova regra de entrada, para possibilitar o acesso do SSH à sua instância do Amazon EC2. Se você fizer isso, poderá se conectar à sua instância do Amazon EC2 para instalar o servidor Web e outros utilitários. Você também se conecta à sua instância do EC2 para fazer upload de conteúdo para seu servidor Web.

   - Digite: **`SSH`**
   - Origem: o endereço IP ou o intervalo da etapa A, por exemplo: **`203.0.113.25/32`**.

d. Escolha Add rule (Adicionar regra).

e. Defina os seguintes valores para a sua nova regra de entrada a fim de permitir o acesso HTTP ao servidor Web.

   - Tipo: **`HTTP`**
   - Source (Origem: **`0.0.0.0/0`**

5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

   Anote o ID do grupo de segurança porque você precisa dele posteriormente neste tutorial.

Criar um grupo de segurança da VPC para um cluster de banco de dados privada

Para manter seu cluster de banco de dados particular, crie um segundo grupo de segurança para acesso privado. Para se conectar a clusters privados de banco de dados na VPC, você adiciona regras de entrada ao grupo de segurança da VPC que permitem o tráfego somente pelo seu servidor web.

Para criar um grupo de segurança de VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Security group name (Nome do grupo de segurança: **`tutorial-db-securitygroup`**
   - Descrição: **`Tutorial DB Instance Security Group`**
   - VPC: escolha a VPC criada na etapa anterior; por exemplo, vpc-***`identifier`*** (tutorial-vpc)
4. Adicione regras de entrada ao grupo de segurança.
   a. Na seção Regras de entrada, escolha Adicionar regra.
   b. Defina os valores a seguir para a sua nova regra de entrada, para permitir o tráfego MySQL na porta 3306 de sua instância do Amazon EC2. Se fizer isso, você poderá se conectar de seu servidor Web ao seu cluster de banco de dados. Ao fazer isso, você poderá armazenar e recuperar dados de sua aplicação Web para seu banco de dados.
      - Tipo: **`MySQL/Aurora`**
      - Source (Origem): o identificador do grupo de segurança tutorial-securitygroup criado anteriormente neste tutorial; por exemplo, sg-9edd5cfb.
5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

Criar um grupo de sub-redes de banco de dados

Um grupo de sub-redes de banco de dados é uma coleção de sub-redes que você cria em uma VPC e depois designa para seus clusters de bancos de dados. Um grupo de sub-redes de banco de dados permite que você especifique uma VPC ao criar clusters de banco de dados.

Como criar um grupo de sub-redes de banco de dados

1. Identifique as sub-redes privadas do seu banco de dados na VPC.

   a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.

   b. Escolha VPC Dashboard (Painel da VPC), depois selecione Subnets (Sub-redes).

   c. Anote os IDs das sub-redes chamadas tutorial-subnet-private1-us-west-2a e tutorial-subnet-private2-us-west-2b.

      Você precisará dos IDs de sub-rede ao criar seu grupo de sub-redes de banco de dados.

2. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

   Conecte-se ao console do Amazon RDS, não ao console da Amazon VPC.

3. No painel de navegação, escolha Subnet groups (Grupos de sub-redes).

4. Escolha Create DB Subnet Group (Criar grupo de sub-redes de banco de dados).

5. Na página Create DB subnet group (Criar grupo de sub-redes de banco de dados), defina esses valores em Subnet group details (Detalhes do grupo de sub-redes):

   - Nome: **`tutorial-db-subnet-group`**
   - Descrição: **`Tutorial DB Subnet Group`**
   - VPC: tutorial-vpc (vpc-***`identifier`***)

6. Na seção Adicionar sub-redes, escolha Zonas de disponibilidade e Sub-redes.

   Para este tutorial, escolha us-west-2a e us-west-2b para as Availability Zones (Zonas de disponibilidade). Para Subnets (Sub-redes), escolha as sub-redes privadas que você identificou na etapa anterior.

7. Escolha Criar.

   Seu novo grupo aparece na lista de grupos de sub-redes de banco de dados no console do RDS. Você pode selecionar o grupo de sub-redes de banco de dados para visualizar detalhes no painel de detalhes na parte inferior da janela. Esses detalhes incluem todas as sub-redes associadas ao grupo.

> **Note**
>
> Se você criou essa VPC para concluir Tutorial: crie um servidor Web e um cluster de banco de dados do Amazon Aurora, crie de cluster de banco de dados seguindo as instruções em Criar um cluster de banco de dados do Amazon Aurora.

Como excluir a VPC

Depois de criar a VPC e outros recursos para este tutorial, você poderá excluí-los se deixarem de ser necessários.

> **Note**
>
> Se você incluiu recursos na VPC que criou para este tutorial, talvez seja necessário excluí-los para poder excluir a VPC. Por exemplo, esses recursos podem incluir instâncias do Amazon EC2 ou clusters de banco de dados do Amazon RDS. Para obter mais informações, consulte Como excluir a sua VPC no Guia do usuário da Amazon VPC.

Para excluir uma VPC e recursos relacionados

1. Exclua o grupo de sub-redes de banco de dados.
    a. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
    b. No painel de navegação, escolha Subnet groups (Grupos de sub-redes).
    c. Selecione o grupo de sub-rede de banco de dados que deseja excluir, como, por exemplo, tutorial-db-subnet-group.
    d. Escolha Delete (Excluir) e, em seguida, Delete (Excluir) na janela de confirmação.
2. Anote o ID da VPC.
    a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
    b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, VPCs.
    c. Na lista, identifique a VPC que criou, como tutorial-vpc.
    d. Anote o VPC ID (ID da VPC) da VPC que você criou. Você precisará do ID da VPC nas etapas posteriores.
3. Exclua os grupos de segurança.

a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, Security Groups (Grupos de segurança).
c. Selecione o grupo de segurança para a instância de banco de dados do Amazon RDS, como, por exemplo, tutorial-db-securitygroup.
d. Em Actions (Ações), escolha Delete grupo de seguranças (Excluir grupos de segurança) e, depois, Delete (Excluir) na página de confirmação.
e. Na página Security Groups (Grupos de segurança), selecione o grupo de segurança para a instância do Amazon EC2, como, por exemplo, tutorial-securitygroup.
f. Em Actions (Ações), escolha Delete grupo de seguranças (Excluir grupos de segurança) e, depois, Delete (Excluir) na página de confirmação.

4. Exclua a VPC.

a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, VPCs.
c. Selecione a VPC que deseja excluir, como, por exemplo, tutorial-vpc.
d. Em Actions (Ações), escolha Delete VPC (Excluir a VPC).

   A página de confirmação mostra outros recursos associados à VPC que também serão excluídos, incluindo as sub-redes associadas a ela.
e. Na página de confirmação, insira **`delete`** e, em seguida, escolha Delete (Excluir).

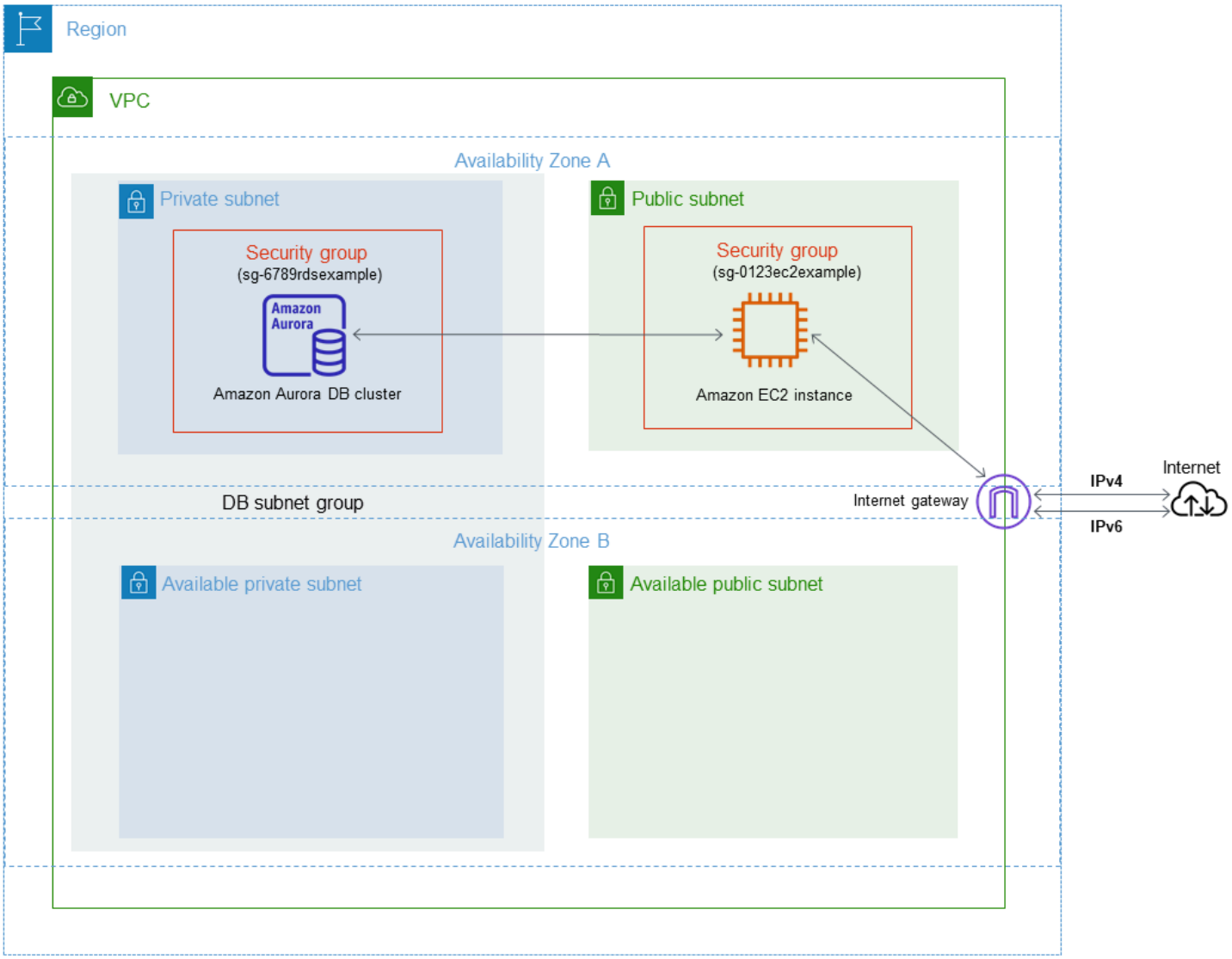
Tutorial: Criar uma VPC para uso com um cluster de banco de dados (modo de pilha dupla)

Um cenário comum inclui um cluster de banco de dados em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Essa VPC compartilha dados com uma instância pública do Amazon EC2 que está sendo executada na mesma VPC.

Neste tutorial, você criará a VPC para esse cenário que funciona com um banco de dados em execução no modo de pilha dupla. Modo de pilha dupla para permitir a conexão pelo protocolo de endereçamento IPv6. Para obter mais informações sobre endereçamento IP, consulte Endereçamento IP do Amazon Aurora.

Os clusters de rede de pilha dupla são compatíveis na maioria das regiões. Para obter mais informações, consulte Disponibilidade de clusters de banco de dados de rede de pilha dupla. Para ver as limitações do modo de pilha dupla, consulte Limitações para clusters de banco de dados de rede de pilha dupla.

O diagrama a seguir mostra esse cenário.



Para obter informações sobre outros cenários, consulte [Cenários para acessar um cluster de banco de dados em uma VPC].

Seu cluster de banco de dados só precisa estar disponível para sua instância do Amazon EC2 e não para a Internet pública. Assim, você cria uma VPC com sub-redes públicas e privadas. A instância do Amazon EC2 é hospedada na sub-rede pública, para que ela possa acessar a Internet pública. O cluster de banco de dados está hospedado em uma sub-rede privada. A instância do Amazon EC2 pode se conectar ao cluster de banco de dados porque ela está hospedada na mesma VPC. No entanto, o cluster de banco de dados não está disponível para a Internet pública, o que oferece maior segurança.

Esse tutorial configura uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Essas sub-redes não são usadas no tutorial. Um grupo de sub-redes do banco de dados

RDS exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a configuração de mais de uma instância de banco de dados Aurora.

Para criar um cluster de banco de dados que use o modo de pilha dupla, especifique Dual-stack mode (Modo de pilha dupla) para a configuração Network type (Tipo de rede). Você também pode modificar um cluster de banco de dados com a mesma configuração. Para ter mais informações sobre como criar um cluster de banco de dados, consulte Criar um cluster de bancos de dados do Amazon Aurora. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster de bancos de dados Amazon Aurora.

Este tutorial descreve como configurar uma VPC para clusters de bancos de dados Amazon Aurora. Para obter mais informações sobre o Amazon VPC, consulte o Manual do usuário do Amazon VPC.

Criar uma VPC com sub-redes públicas e privadas

Use o seguinte procedimento para criar uma VPC com sub-redes públicas e privadas.

Para criar uma VPC e sub-redes

1. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região na qual deseja criar sua VPC. Este exemplo usa a região Leste dos EUA (Ohio).
3. No canto superior esquerdo, escolha VPC dashboard (Painel da VPC). Para começar a criar uma VPC, escolha Create VPC (Criar VPC).
4. Para Resources to create (Recursos a serem criados) em VPC settings (Configurações da VPC), escolha VPC and more (VPC e mais).
5. Para o restante de VPC settings (Configurações da VPC), defina os seguintes valores:

    - Name tag auto-generation (Geração automática da etiqueta de nome – **`tutorial-dual-stack`**
    - IPv4 CIDR block (Bloco CIDR IPv – **`10.0.0.0/16`**
    - IPv6 CIDR block (Bloco CIDR IPv6): Amazon-provided IPv6 CIDR block (Bloco CIDR IPv6 fornecido pela Amazon)
    - Tenancy (Locação): Default (Padrão)
    - Number of Availability Zones (AZs) [Número de zonas de disponibilidade (AZs) – 2
    - Customize AZs (Personalizar AZs): mantenha os valores padrão.
    - Number of public subnet (Número de sub-redes públicas – 2

- Number of private subnets (Número de sub-redes privadas – 2
- Customize subnets CIDR blocks (Personalizar blocos CIDR de sub-redes): mantenha os valores padrão.
- NAT gateways ($) (Gateways NAT ($)): None (Nenhum)
- Egress only internet gateway (Gateway da Internet somente de saída): No (Não)
- VPC endpoints (Endpoints da VPC): None (Nenhum)
- DNS options (Opções de DNS): mantenha os valores padrão.

> **Note**
>
> O Amazon RDS exige pelo menos duas sub-redes em duas zonas de disponibilidade diferentes para oferecer suporte a implantações de instâncias de banco de dados multi-AZ. Este tutorial cria uma implantação single-AZ, mas o requisito facilita a conversão para uma implantação de instância de banco de dados multi-AZ no futuro.

6. Escolha Criar VPC.

Criar um grupo de segurança da VPC para uma instância pública do Amazon EC2

Em seguida, você criará um grupo de segurança para acesso público. Para se conectar a instâncias públicas do EC2 na sua VPC, adicione regras de entrada ao grupo de segurança de sua VPC que permitam que o tráfego se conecte da Internet.

Para criar um grupo de segurança de VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Security group name (Nome do grupo de segurança: **`tutorial-dual-stack-securitygroup`**
   - Descrição: **`Tutorial Dual-Stack Security Group`**
   - VPC: escolha a VPC criada na etapa anterior, por exemplo vpc-***`identifier`***(tutorial-dual-stack-vpc)

4. Adicione regras de entrada ao grupo de segurança.

   a. Determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH).

      Um exemplo de endereço IPv4 (Internet Protocol versão 4) é `203.0.113.25/32`.
      Um exemplo de intervalo de endereços de Protocolo de Internet versão 6 (IPv6) é `2001:db8:1234:1a00::/64`.

      Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Se sim, especifique o intervalo de endereços IP utilizado por computadores cliente.

      > **Warning**
      >
      > Se usar `0.0.0.0/0` para IPv4 ou `::0` para IPv6, você possibilitará que todos os endereços IP acessem suas instâncias públicas usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. No ambiente de produção, autorize apenas um endereço IP específico ou um intervalo de endereços a acessar as instâncias.

   b. Na seção Regras de entrada, escolha Adicionar regra.

   c. Defina os valores a seguir para a sua nova regra de entrada, para permitir o acesso Secure Shell (SSH) à sua instância do Amazon EC2. Se você fizer isso, poderá se conectar à instância do EC2 para instalar clientes SQL e outras aplicações. Especifique um endereço IP para que você possa acessar sua instância do EC2:

      - Tipo: **`SSH`**
      - Source (Fonte): o endereço IP ou o intervalo da etapa a. Um exemplo de endereço IP IPv4 é **`203.0.113.25/32`**. Um exemplo de endereço IP IPv6 é **`2001:DB8::/32`**.

5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

   Anote o ID do grupo de segurança porque você precisa dele posteriormente neste tutorial.

Criar um grupo de segurança da VPC para um cluster de banco de dados privada

Para manter seu cluster de banco de dados particular, crie um segundo grupo de segurança para acesso privado. Para se conectar a clusters de banco de dados privados em sua VPC, adicione regras de entrada ao seu grupo de segurança de VPC. Eles permitem o tráfego somente de sua instância do Amazon EC2.

Para criar um grupo de segurança de VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Security group name (Nome do grupo de segurança: **`tutorial-dual-stack-db-securitygroup`**
   - Descrição: **`Tutorial Dual-Stack DB Instance Security Group`**
   - VPC: escolha a VPC criada na etapa anterior, por exemplo vpc-***`identifier`***(tutorial-dual-stack-vpc)
4. Adicione regras de entrada ao grupo de segurança:
   a. Na seção Regras de entrada, escolha Adicionar regra.
   b. Defina os valores a seguir para a sua nova regra de entrada, para permitir o tráfego MySQL na porta 3306 de sua instância do Amazon EC2. Se fizer isso, você poderá se conectar de sua instância do EC2 ao seu cluster de banco de dados. Ao fazer isso, você poderá enviar dados da instância do EC2 para o banco de dados.
      - Digite: MySQL/Aurora
      - Source (Origem): o identificador do grupo de segurança tutorial-dual-stack-securitygroup criado anteriormente neste tutorial; por exemplo, sg-9edd5cfb.
5. Para criar o grupo de segurança, escolha Criar grupo de segurança.

Criar um grupo de sub-redes de banco de dados

Um grupo de sub-redes de banco de dados é uma coleção de sub-redes que você cria em uma VPC e depois designa para seus clusters de bancos de dados. Ao usar um grupo de sub-redes de banco de dados, você pode especificar uma VPC específica ao criar clusters de banco de dados. Para

criar um grupo de sub-redes de banco de dados que seja compatível com `DUAL`, todas as sub-redes devem ser compatíveis com `DUAL`. Para ser compatível com `DUAL`, uma sub-rede deve ter um CIDR IPv6 associado a ela.

Como criar um grupo de sub-redes de banco de dados

1. Identifique as sub-redes privadas do seu banco de dados na VPC.
   a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
   b. Escolha VPC Dashboard (Painel da VPC), depois selecione Subnets (Sub-redes).
   c. Anote os IDs das sub-redes chamadas tutorial-dual-stack-subnet-private1-us-west-2a e tutorial-dual-stack-subnet-private2-us-west-2b.

      Você precisará dos IDs de sub-rede ao criar seu grupo de sub-redes de banco de dados.
2. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.

   Conecte-se ao console do Amazon RDS, não ao console da Amazon VPC.
3. No painel de navegação, escolha Subnet groups (Grupos de sub-redes).
4. Escolha Create DB Subnet Group (Criar grupo de sub-redes de banco de dados).
5. Na página Create DB subnet group (Criar grupo de sub-redes de banco de dados), defina esses valores em Subnet group details (Detalhes do grupo de sub-redes):
   - Nome: **`tutorial-dual-stack-db-subnet-group`**
   - Descrição: **`Tutorial Dual-Stack DB Subnet Group`**
   - VPC: tutorial-dual-stack-vpc (vpc-***`identifier`***)
6. Na seção Add subnets (Adicionar sub-redes), escolha as opções Availability Zones (Zonas de disponibilidade) e Subnets (Sub-redes).

   Para este tutorial, escolha us-east-2a e us-east-2b para as Availability Zones (Zonas de disponibilidade). Para Subnets (Sub-redes), escolha as sub-redes privadas que você identificou na etapa anterior.
7. Escolha Criar.

Seu novo grupo aparece na lista de grupos de sub-redes de banco de dados no console do RDS. Você pode escolher o grupo de sub-redes de banco de dados para ver os detalhes. Isso inclui os

protocolos de endereçamento compatíveis e todas as sub-redes associadas ao grupo e o tipo de rede compatível com o grupo de sub-redes de banco de dados.

Criar uma instância do Amazon EC2 no modo de pilha dupla

Para criar uma instância do Amazon EC2, siga as instruções em Iniciar uma instância usando o novo assistente de inicialização de instância, versão beta no Guia do usuário do Amazon EC2.

Na página Configure Instance Details (Configurar os detalhes da instância), defina esses valores e mantenha os outros valores como padrão:

- Rede: escolha uma VPC existente com sub-redes públicas e privadas, como tutorial-dual-stack-vpc (vpc-*identifier*), criada em Criar uma VPC com sub-redes públicas e privadas.
- Subnet (Sub-rede): escolha uma sub-rede pública existente, como subnet-***identifier*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a criada em Criar um grupo de segurança da VPC para uma instância pública do Amazon EC2.
- Auto-assign Public IP (Atribuir automaticamente IP público): escolha Enable (Habilitar).
- Auto-assign IPv6 IP (Atribuir automaticamente IPv6): escolha Enable (Habilitar).
- Firewall (security groups) (Firewall (grupos de segurança)): escolha Select an existing security group (Selecionar um grupo de segurança existente).
- Common security groups (Grupos de segurança comuns): selecione um grupo de segurança existente, como o `tutorial-securitygroup` criado em Criar um grupo de segurança da VPC para uma instância pública do Amazon EC2. Verifique se o grupo de segurança escolhido inclui regras de entrada para Secure Shell (SSH) e acesso HTTP.

Criar um cluster de banco de dados no modo de pilha dupla

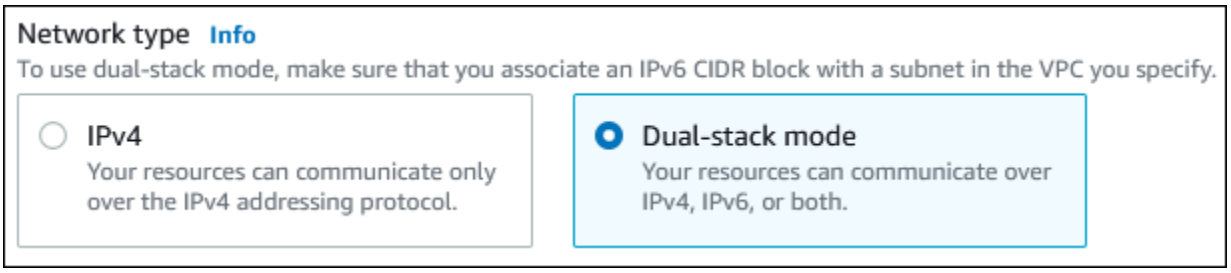
Nesta etapa, crie um cluster de banco de dados do Amazon RDS para execução no modo de pilha dupla.

Como criar uma instância de banco de dados

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No canto superior direito do console, escolha a Região da AWS na qual você quer criar o cluster de banco de dados. Este exemplo usa a região Leste dos EUA (Ohio).
3. No painel de navegação, escolha Databases (Bancos de dados).

4. Selecione Criar banco de dados.
5. Na página Create database (Criar banco de dados), verifique se a opção Standard create (Criação padrão) está selecionada e escolha o tipo de mecanismo de banco de dados Aurora MySQL.
6. Na seção Conectividade, defina estes valores:

   - Network type (Tipo de rede): escolha Dual-stack mode (Modo de pilha dupla).



   - Virtual private cloud (VPC) (Nuvem privada virtual (VPC): escolha uma VPC existente com sub-redes públicas e privadas, como a tutorial-dual-stack-vpc (vpc-*identifier*) criada em Criar uma VPC com sub-redes públicas e privadas.

     A VPC deve ter sub-redes em zonas de disponibilidade diferentes.
   - DB subnet group (Grupo de sub-redes de banco de dados): escolha um grupo de sub-redes de banco de dados para a VPC, como tutorial-dual-stack-db-subnet-group criado em Criar um grupo de sub-redes de banco de dados.
   - Public access (Acesso público): escolha No (Não).
   - VPC security group (firewall) (Grupo de segurança da VPC (firewall)): selecione Choose existing (Escolher existente).
   - Existing VPC grupo de seguranças (Grupos de segurança da VPC existentes): escolha um grupo de segurança da VPC existente configurado para acesso privado, como tutorial-dual-stack-db-securitygroup criado em Criar um grupo de segurança da VPC para um cluster de banco de dados privada.

     Remova outros grupos de segurança, como o grupo de segurança padrão, escolhendo o X associado.
   - Availability Zone (Zona de disponibilidade): escolha us-west-2a.

     Para evitar o tráfego entre zonas, certifique-se de que a instância de banco de dados e a instância do EC2 estejam na mesma zona de disponibilidade.

7. Nas seções restantes, especifique suas configurações de cluster de banco de dados. Para obter informações sobre cada configuração, consulte Configurações de clusters de bancos de dados do Aurora.

Conectar-se à sua instância do Amazon EC2 e ao cluster de banco de dados

Depois de criar a instância do Amazon EC2 e o cluster de banco de dados no modo de pilha dupla, você poderá se conectar a cada uma usando o protocolo IPv6. Para se conectar a uma instância do Amazon EC2 usando o protocolo IPv6, siga as instruções em Conecte-se à sua instância do Linux no Guia do usuário do Amazon EC2.

Para se conectar à instância de banco de dados do Aurora MySQL por meio da instância do Amazon EC2, siga as instruções em Conectar-se a um cluster de banco de dados do Aurora MySQL.

Como excluir a VPC

Depois de criar a VPC e outros recursos para este tutorial, você poderá excluí-los se deixarem de ser necessários.

Se você incluiu recursos na VPC que criou para este tutorial, talvez seja necessário excluí-los para poder excluir a VPC. Exemplos de recursos são instâncias do Amazon EC2 ou clusters de banco de dados. Para obter mais informações, consulte Como excluir a sua VPC no Guia do usuário da Amazon VPC.

Para excluir uma VPC e recursos relacionados

1. Exclua o grupo de sub-redes de banco de dados:
    a. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
    b. No painel de navegação, escolha Subnet groups (Grupos de sub-redes).
    c. Selecione o grupo de sub-redes de banco de dados a ser excluído, por exemplo, tutorial-db-subnet-group.
    d. Escolha Delete (Excluir) e, em seguida, Delete (Excluir) na janela de confirmação.
2. Anote o ID da VPC:
    a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.
    b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, VPCs.
    c. Na lista, identifique a VPC que você criou, como tutorial-dual-stack-vpc.

d. Anote o valor VPC ID (ID da VPC) da VPC que você criou. Você precisará do ID da VPC nas etapas subsequentes.

3. Exclua os grupos de segurança:

   a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.

   b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, Security Groups (Grupos de segurança).

   c. Selecione o grupo de segurança para a instância de banco de dados do Amazon RDS, como tutorial-dual-stack-db-securitygroup.

   d. Em Actions (Ações), escolha Delete grupo de seguranças (Excluir grupos de segurança) e, depois, Delete (Excluir) na página de confirmação.

   e. Na página Security Groups (Grupos de segurança), selecione o grupo de segurança para a instância do Amazon EC2, como tutorial-dual-stack-securitygroup.

   f. Em Actions (Ações), escolha Delete grupo de seguranças (Excluir grupos de segurança) e, depois, Delete (Excluir) na página de confirmação.

4. Exclua o gateway NAT:

   a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.

   b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, NAT Gateways (Gateways NAT).

   c. Selecione o gateway NAT da VPC que você criou. Use o ID da VPC para identificar o gateway NAT correto.

   d. Em Actions (Ações), escolha Delete NAT gateway (Excluir gateway NAT).

   e. Na página de confirmação, insira **`delete`** e, em seguida, escolha Delete (Excluir).

5. Exclua a VPC:

   a. Abra o console da Amazon VPC, em https://console.aws.amazon.com/vpc/.

   b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, VPCs.

   c. Selecione a VPC que deseja excluir, como a tutorial-dual-stack-vpc.

   d. Em Actions (Ações), escolha Delete VPC (Excluir a VPC).

      A página de confirmação mostra outros recursos associados à VPC que também serão excluídos, incluindo as sub-redes associadas a ela.

   e. Na página de confirmação, insira **`delete`** e, em seguida, escolha Delete (Excluir).

6. Libere os endereços de IP elásticos;

a. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
b. Escolha Painel do EC2 e, em seguida, IPs elásticos.
c. Selecione o endereço de IP elástico que deseja liberar.
d. Em Actions (Ações), escolha Release Elastic IP addresses (Liberar endereços de IP elásticos).
e. Na página de confirmação, escolha Release (Liberar).

Cotas e restrições do Amazon Aurora

A seguir, você pode encontrar uma descrição das cotas de recursos e restrições de nomenclatura do Amazon Aurora.

Tópicos

- Cotas no Amazon Aurora
- Restrições de nomenclatura no Amazon Aurora
- Limites de tamanho do Amazon Aurora

Cotas no Amazon Aurora

Cada conta da AWS tem cotas, para cada região da AWS, sobre o número de recursos do Amazon Aurora que podem ser criados. Depois que a cota de um recurso é atingida, as chamadas adicionais para criá-lo falham, com uma exceção.

A tabela a seguir lista os recursos e suas cotas por região da AWS.

Nome	Padrão	Ajustá	Descrição
Autorizações por grupos de segurança de banco de dados	Cada região compatível: 20	Não	Número de autorizações de grupo de segurança por grupo de segurança do banco de dados.
Endpoints personalizados por cluster de banco de dados	Cada região compatível: 5	Sim	O número máximo de endpoints personalizados que você pode criar por cluster de banco de dados do Aurora nessa conta na região atual. Esse valor reflete o maior número de endpoints personalizados em um cluster de banco de dados na conta. Outros

Nome	Padrão	Ajustá	Descrição
			clusters de banco de dados na conta podem ter um número menor de endpoints personalizados.
Versões de mecanismos personalizados	Cada região compatível: 40	Sim	O número máximo de versões de mecanismo personalizadas permitidas nessa conta na região atual.
Grupos de parâmetros de cluster de banco de dados	Cada região compatível: 50	Sim	O número máximo de grupos de parâmetros de cluster de banco de dados.
clusters de banco de dados	Cada região compatível: 40	Sim	O número máximo de clusters do Aurora para essa conta na Região atual.
Instâncias de banco de dados	ap-south-1: 20 Cada uma das outras regiões compatíveis: 40	Sim	O número máximo de instâncias de banco de dados permitidas nessa conta na Região atual.
Grupos de fragmentos de banco de dados	Cada região compatível: 5	Sim	O número máximo de grupos de fragmentos de banco de dados para o Aurora Limitless Database nesta conta na região atual

Nome	Padrão	Ajustá	Descrição
Grupos de sub-redes de banco de dados	ap-south-1: 20 Cada uma das outras regiões compatíveis: 50	Sim	O número máximo de grupos de sub-redes de banco de dados.
Tamanho do corpo da solicitação HTTP da API de dados	Cada região compatível: 4 megabytes	Não	O tamanho máximo permitido para o corpo de solicitação HTTP.
Máximo de pares simultâneos de segredo de cluster da API de dados	Cada região compatível: 30	Não	O número máximo de pares exclusivos de segredos e clusters de banco de dados do Aurora Sem Servidor v1 em solicitações simultâne as da API de dados para essa conta na região da AWS atual.
Número máximo de solicitações simultâneas da API de dados	Cada região compatível: 500	Não	O número máximo de solicitações da API de dados para um cluster de banco de dados do Aurora Sem Servidor v1 que usam o mesmo segredo e podem ser processadas simultane amente. Solicitações adicionais são colocadas em fila e processadas à medida que as solicitaç ões em andamento são concluídas.

Nome	Padrão	Ajustá	Descrição
Tamanho máximo do conjunto de resultados da API de dados	Cada região compatível: 1 megabyte	Não	O tamanho máximo do conjunto de resultados do banco de dados que pode ser retornado pela API de dados.
Tamanho máximo da API de dados da string de resposta JSON	Cada região compatível: 10 megabytes	Não	O tamanho máximo da string de resposta JSON simplificada retornada pela API de dados do RDS.
Solicitações de API de dados por segundo	Cada região compatível: 1.000 por segundo	Não	O número máximo de solicitações para a API de dados por segundo permitido nessa conta na região da AWS atual. Essa cota aplica-se somente aos clusters do Amazon Aurora Sem Servidor v1.
Assinaturas de eventos	Cada região compatível: 20	Sim	O número máximo de assinaturas de eventos.
Perfis do IAM por cluster de banco de dados	Cada região compatível: 5	Sim	O número máximo de perfis do IAM associados a um cluster de banco de dados.
Perfis do IAM por instância de banco de dados	Cada região compatível: 5	Sim	O número máximo de perfis do IAM associados a uma instância de banco de dados.

Nome	Padrão	Ajustá	Descrição
Integrações	Cada região compatível: 100	Não	O número máximo de integrações permitidas nessa conta na região da AWS atual.
Snapshots de cluster de banco de dados manual	Cada região compatível: 100	Sim	O número máximo de snapshots manuais de cluster de banco de dados.
Snapshots manuais da instância de banco de dados	Cada região compatível: 100	Sim	O número máximo de snapshots manuais de instância de banco de dados.
Grupos de opções	Cada região compatível: 20	Sim	O número máximo de grupos de opções.
Grupos de parâmetros	ap-south-1: 20 Cada uma das outras regiões compatíveis: 50	Sim	O número máximo de grupos de parâmetros.
Proxies	Cada região compatível: 20	Sim	O número máximo de proxies permitidos nessa conta na Região da AWS atual.
Réplicas de leitura por primário	Cada região compatível: 15	Sim	O número máximo de réplicas de leitura por instância de banco de dados primária. Essa cota não pode ser ajustada para o Amazon Aurora.

Nome	Padrão	Ajustá	Descrição
Instâncias de bancos de dados reservadas	ap-south-1: 20 Cada uma das outras regiões compatíveis: 40	Sim	O número máximo de instâncias de banco de dados reservadas permitidas nessa conta na Região da AWS atual.
Grupos de segurança	ap-south-1: 20 Cada uma das outras regiões compatíveis: 25	Sim	O número máximo de grupos de segurança do banco de dados.
Sub-redes por grupo de sub-redes do banco de dados	Cada região compatível: 20	Não	O número máximo de sub-redes para por grupo de sub-redes de banco de dados.
Armazenamento total para todas as instâncias de banco de dados	Cada região compatível: 100.000	Sim	O armazenamento total máximo (em GB) em volumes do EBS para todas as instâncias de banco de dados do Amazon RDS somadas. Essa cota não se aplica ao Amazon Aurora, que tem um volume máximo de cluster de 128 TiB para cada cluster de banco de dados.

Note

Por padrão, você pode ter um total de 40 instâncias de banco de dados. As instâncias de banco de dados do RDS, as instâncias de bancos de dados Aurora, as instâncias do Amazon Neptune e as instâncias do Amazon DocumentDB se aplicam a essa cota.

Se a sua aplicação exigir mais instâncias de banco de dados, você poderá solicitar instâncias de banco de dados adicionais abrindo o Console de Service Quotas. No painel de navegação, escolha AWSServiços da . Escolha Amazon Relational Database Service (Amazon RDS), escolha uma cota e siga as instruções para solicitar um aumento de cota. Para obter mais informações, consulte Como solicitar um aumento de cota no Guia do usuário do Service Quotas.

Para o RDS para Oracle, é possível criar até 15 réplicas de leitura por instância de banco de dados de origem em cada região, mas recomendamos limitar as réplicas a cinco para minimizar o atraso na replicação.

Os backups gerenciados pelo AWS Backup são considerados snapshots manuais de cluster de banco de dados, mas não são contabilizados para a cota de snapshot manual do cluster. Para obter mais informações sobre o AWS Backup, consulte o Guia do desenvolvedor do AWS Backup.

Se você usar qualquer operação de API do RDS e exceder a cota padrão do número de chamadas por segundo, a API do Amazon RDS emitirá um erro como o seguinte.

ClientError: ocorreu um erro (ThrottlingException) ao chamar a operação *API_name*: taxa excedida.

Aqui, reduza o número de chamadas por segundo. A cota destina-se a cobrir a maioria dos casos de uso. Se cotas maiores forem necessárias, solicite um aumento de cota usando uma das seguintes opções:

- No console, abra o console do Service Quotas.
- Na AWS CLI, use o comando request-service-quota-increase da AWS CLI.

Para acessar mais informações, consulte o Guia do usuário do Service Quotas.

Restrições de nomenclatura no Amazon Aurora

As restrições de nomenclatura no Amazon Aurora são as seguintes:

- Identificador do cluster de banco de dados:
  - Devem conter 1 a 63 caracteres alfanuméricos ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.

  - Deve ser exclusivo para todas as instâncias de Banco de Dados por conta da AWS, por região da AWS.
- Nome do banco de dados inicial: as restrições de nome de banco de dados entre o Aurora MySQL e o Aurora PostgreSQL. Para ter mais informações, consulte as configurações disponíveis ao criar cada cluster de banco de dados.
- Nome de usuário principal: restrições de nomes de usuário principal são diferentes para cada mecanismo de banco de dados. Para ter mais informações, consulte as configurações disponíveis ao criar cada cluster de banco de dados.
- Senha principal:
  - A senha do usuário principal do banco de dados pode incluir qualquer caractere ASCII imprimível, exceto `/`, `'`, `"`, `@` ou um espaço.
  - Dependendo do mecanismo de banco de dados, a senha pode conter o seguinte número de caracteres ASCII imprimíveis:
    - Aurora MySQL: 8 – 41
    - Aurora PostgreSQL: 8 – 99
- Grupo de parâmetros de banco de dados:
  - Devem conter de 1 a 255 caracteres alfanuméricos.
  - O primeiro caractere deve ser uma letra.
  - Os hifens são permitidos, mas o nome não pode terminar com um hífen nem conter dois hifens consecutivos.
- Grupo de sub-redes do banco de dados:
  - Devem conter de 1 a 255 caracteres.
  - Caracteres alfanuméricos, espaços, hifens, sublinhados e pontos são permitidos.

Limites de tamanho do Amazon Aurora

Limites de tamanho de armazenamento

O tamanho máximo do volume do cluster do Aurora varia de acordo com a versão do mecanismo:

No máximo 256 TiB:

- Versões do Aurora PostgreSQL:
  - 17.5 e todas as versões posteriores
  - 16.9 e posterior

- 15.13 e posterior
- Aurora MySQL versão 3.10 (compatível com MySQL 8.0.42) e versões posteriores

No máximo 128 TiB:

- Todas as versões anteriores do Aurora PostgreSQL
- Todas as versões 3 disponíveis do Aurora MySQL; Aurora MySQL versão 2, versões 2.09 e posteriores

Para ter mais informações sobre ajuste de escala automático de armazenamento, consulte Como o armazenamento do Aurora redimensiona automaticamente.

Para monitorar o espaço de armazenamento restante, você pode usar a métrica `AuroraVolumeBytesLeftTotal`. Para obter mais informações, consulte Métricas no nível do cluster do Amazon Aurora.

Limites de tamanho de tabelas de SQL

Num cluster de bancos de dados MySQL no Aurora, o tamanho máximo de cada tabela é de 64 tebibytes (TiB). Num cluster de bancos de dados PostgreSQL no Aurora, esse limite é de 32 tebibytes (TiB). Recomendamos que você siga as práticas recomendadas do design de tabelas, como o particionamento de tabelas grandes.

Limites de ID de espaço de tabela

O ID de espaço de tabela máximo para o Aurora MySQL é 2147483647. Se você cria e elimina tabelas com frequência, lembre-se de seus IDs de espaço de tabela e planeje usar despejos lógicos. Para obter mais informações, consulte Migração lógica do MySQL para o Amazon Aurora MySQL usando o mysqldump.

Solução de problemas do Amazon Aurora

Use as seções a seguir para solucionar problemas que possam surgir com instâncias de bancos de dados do Amazon RDS e do Amazon Aurora.

Tópicos

- Não é possível conectar-se à instância de banco de dados do Amazon RDS
- Problemas de segurança do Amazon RDS
- Redefinir a senha de proprietário da instância de banco de dados
- Interrupção ou reinicialização da instância de banco de dados do Amazon RDS
- Alterações de parâmetros de banco de dados do Amazon RDS que não entram em vigor
- Problemas de memória liberável no Amazon Aurora
- Problemas de replicação no Amazon Aurora MySQL

Para obter informações sobre como depurar problemas usando a API do Amazon RDS, consulte Solução de problemas de aplicações no Aurora.

Não é possível conectar-se à instância de banco de dados do Amazon RDS

Quando você não consegue se conectar a uma instância de banco de dados, as causas a seguir são motivos comuns:

- Regras de entrada – as regras de acesso impostas pelo firewall local e os endereços IP autorizados a acessar a instância de banco de dados podem não corresponder. O problema está provavelmente nas regras de entrada do seu grupo de segurança.

  Por padrão, as instâncias de banco de dados não permitem acesso. O acesso é concedido por meio de um grupo de segurança associado à VPC que permite o tráfego de entrada e saída da instância de banco de dados. Se necessário, adicione regras de entrada e saída para sua situação específica ao grupo de segurança. Você pode especificar um endereço IP, um intervalo de endereços IP ou outro grupo de segurança da VPC.

> **Note**
>
> Ao adicionar uma nova regra de entrada, escolha My IP (Meu IP) para a Source (Origem) a fim de permitir o acesso à instância de banco de dados do endereço IP detectado em seu navegador.

Para ter mais informações sobre como configurar um grupo de segurança, consulte Fornecer acesso ao cluster de banco de dados na VPC criando um grupo de segurança.

> **Note**
>
> Conexões de cliente de endereços IP dentro do intervalo 169.254.0.0/16 não são permitidas. Esse é o APIPA (Automatic Private IP Addressing Range, Intervalo de endereçamento IP privado automático), usado para o endereçamento de link local.

- Acessibilidade pública – para se conectar à sua instância de banco de dados de fora da VPC, como por exemplo, usando uma aplicação cliente, a instância deve ter um endereço IP público atribuído a ela.

  Para tornar a instância acessível publicamente, modifique-a e escolha Yes (Sim) em Public accessibility (Acessibilidade pública). Para obter mais informações, consulte Ocultar um cluster de banco de dados em uma VPC da Internet.
- Porta: a porta que você especificou quando criou a instância de banco de dados não pode ser usada para enviar ou receber comunicações devido às restrições de firewall locais. Verifique com seu administrador de rede para determinar se a rede permite que a porta especificada seja usada para a comunicação de entrada e saída.
- Disponibilidade – a instância de banco de dados recém-criada fica com o status `creating` até que esteja pronta para uso. Quando o estado for alterado para `available`, será possível se conectar à instância de banco de dados. Dependendo do tamanho da sua instância de banco de dados, pode demorar até 20 minutos para que uma instância esteja disponível.
- Gateway da Internet – para que uma instância de banco de dados seja acessível publicamente, as sub-redes no grupo de sub-redes de banco de dados devem ter um gateway da Internet.

Como configurar um gateway da Internet para uma sub-rede

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Databases (Bancos de dados) e escolha o nome da instância de banco de dados.
3. Na guia Connectivity & security (Conectividade e segurança) anote os valores do ID da VPC em VPC e o ID da sub-rede em Subnets (Sub-redes).
4. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
5. No painel de navegação, escolha Gateways da Internet. Verifique se há um gateway de internet associado à sua VPC. Caso contrário, escolha Criar gateway da internet para criar um gateway da Internet. Selecione o gateway de internet e escolha Associar à VPC e siga as instruções para associá-la à sua VPC.
6. No painel de navegação, escolha Sub-redes e selecione sua sub-rede.
7. Na guia Tabela de rotas, verifique que há uma rota com `0.0.0.0/0` como o destino e o gateway de Internet para sua VPC como destino.

   Se você estiver se conectando à sua instância usando o endereço IPv6, verifique se há uma rota para todo o tráfego IPv6 (`::/0`) que aponta para o gateway de Internet. Caso contrário, faça o seguinte:

   a. Escolha o ID da tabela de rotas (rtb-xxxxxxxx) para navegar para a tabela de rotas.
   b. Na guia Routes (Rotas), escolha Edit routes (Editar rotas). Escolha Add route (Adicionar rota), use `0.0.0.0/0` como o destino, e o gateway da Internet como o destino.

      Para IPv6, escolha Add route (Adicionar rota), use `::/0` como o destino, e o gateway da Internet como o destino.
   c. Escolha Save routes (Salvar rotas).

   Além disso, se você estiver tentando se conectar ao endpoint IPv6, verifique se o intervalo de endereços IPv6 do cliente está autorizado a se conectar à instância de banco de dados.

Para ter mais informações, consulte Trabalhar com um cluster de banco de dados em uma VPC.

Testar uma conexão a uma instância de banco de dados

É possível testar sua conexão a uma instância de banco de dados usando ferramentas comuns do Linux ou do Microsoft Windows.

Em um terminal Linux ou Unix, teste a conexão inserindo o seguinte. Substitua *DB-instance-endpoint* pelo endpoint e *port* pela porta de sua instância de banco de dados.

```
nc -zv DB-instance-endpoint port
```

Veja a seguir um exemplo de comando e o valor de retorno.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Os usuários do Windows podem usar o Telnet para testar a conexão com uma instância de banco de dados. As ações do Telnet não têm suporte além do teste da conexão. Se uma conexão for bem-sucedida, a ação não retorna uma mensagem. Se uma conexão não for bem-sucedida, você receberá uma mensagem de erro, como a seguinte:

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Se as ações do Telnet retornarem êxito, seu grupo de segurança está corretamente configurado.

> **Note**
>
> O Amazon RDS não aceita o tráfego pelo protocolo ICMP (protocolo de mensagens de controle da Internet), incluindo ping.

Solução de problemas de autenticação da conexão

Em alguns casos, você pode se conectar à sua instância de banco de dados, mas recebe erros de autenticação. Nesses casos, convém redefinir a senha do usuário principal para a instância de banco de dados. Isso pode ser feito ao modificar a instância do RDS.

Problemas de segurança do Amazon RDS

Para evitar problemas de segurança, nunca use o nome do usuário-raiz e a senha da Conta da AWS para uma conta de usuário. A prática recomendada é usar o usuário-raiz para criar usuários e atribuí-los a contas de usuário de banco de dados. Também é possível usar o usuário-raiz para criar outras contas de usuário, se necessário.

Para obter informações sobre a criação de usuários, consulte Criar um usuário do IAM na sua Conta da AWS. Para obter informações sobre como criar usuários no Centro de Identidade do AWS IAM, consulte Manage identities in IAM Identity Center (Gerenciar identidades no IAM Identity Center).

Mensagem de erro "Falha ao recuperar atributos da conta, certas funções do console podem ser prejudicadas."

Esse erro pode ser exibido por vários motivos. Pode ser porque sua conta não tem as permissões ou não tenha sido configurada corretamente. Se a sua conta for nova, talvez você não tenha esperado que ela ficasse pronta. Se for uma conta existente, talvez você não tenha permissões nas suas políticas de acesso para realizar determinadas ações, como criar uma instância de banco de dados. Para corrigir o problema, o administrador precisa fornecer os perfis necessários para a sua conta. Para ter mais informações, consulte a documentação do IAM.

Redefinir a senha de proprietário da instância de banco de dados

Se não conseguir acessar seu cluster de banco de dados, você poderá fazer login como o usuário mestre. Depois, você poderá redefinir as credenciais de outros usuários administrativos ou funções. Se não conseguir fazer login como usuário mestre, o proprietário da conta da AWS poderá redefinir a senha do usuário mestre. Para obter detalhes sobre quais contas administrativas ou funções deverão ser redefinidas, consulte Privilégios da conta de usuário mestre.

É possível alterar a senha da instância de banco de dados usando o console do Amazon RDS, o comando da AWS CLI modify-db-instance ou a operação de API ModifyDBInstance. Para ter mais

informações sobre como modificar uma instância de banco de dados em um cluster de banco de dados, consulte Modificar uma instância de banco de dados em um cluster de banco de dados.

Interrupção ou reinicialização da instância de banco de dados do Amazon RDS

Uma interrupção da instância de banco de dados pode ocorrer quando a instância de banco de dados é reinicializada. A interrupção também pode ocorrer quando a instância de banco de dados é colocada em um estado que impede o acesso a ela e quando o banco de dados é reiniciado. Uma reinicialização pode ocorrer ao reinicializar manualmente a instância de banco de dados. Uma reinicialização também pode ocorrer quando você altera uma configuração da instância de banco de dados que exija uma reinicialização para que tenha efeito.

A reinicialização de uma instância de banco de dados só ocorre quando você altera uma configuração que exija uma reinicialização ou quando você faz uma reinicialização manualmente. Uma reinicialização poderá ocorrer imediatamente se você alterar uma configuração e solicitar que ela tenha efeito imediato. Ou isso pode ocorrer durante a janela de manutenção da instância de banco de dados.

Uma reinicialização de instância de banco de dados ocorre imediatamente quando ocorre um dos seguintes eventos:

- Você altera o período de retenção de backup para uma instância de banco de dados de 0 para um valor diferente de zero ou vice-versa. Depois, defina Apply Immediately (Aplicar imediatamente) como `true`.
- Você altera a classe de instância de banco de dados, e Apply Immediately (Aplicar imediatamente) é definido como `true`.

A reinicialização de uma instância de banco de dados ocorre durante a janela de manutenção quando ocorre um dos seguintes:

- Você altera o período de retenção de backup para uma instância de banco de dados de 0 para um valor diferente de zero ou vice-versa e define Apply Immediately (Aplicar imediatamente) como `false`.
- Você altera a classe de instância de banco de dados, e Apply Immediately (Aplicar imediatamente) é definido como `false`.

Quando você altera um parâmetro estático em um grupo de parâmetros de banco de dados, a alteração não terá efeito até que a instância de banco de dados associada ao grupo de parâmetros seja reinicializada. A alteração requer uma reinicialização manual. A instância de banco de dados não é reinicializada automaticamente durante a janela de manutenção.

Alterações de parâmetros de banco de dados do Amazon RDS que não entram em vigor

Em alguns casos, talvez você altere um parâmetro em um grupo de parâmetros do banco de dados, mas não veja as alterações entrarem em vigor. Nesse caso, provavelmente será necessário reinicializar a instância de banco de dados associada ao grupo de parâmetros do banco de dados. Quando você altera um parâmetro dinâmico, a alteração entra em vigor imediatamente. Quando você altera um parâmetro estático, a alteração não entrará em vigor até que você reinicie a instância de banco de dados associada ao grupo de parâmetros.

Você pode reinicializar uma instância de banco de dados usando o console do RDS. Ou você pode chamar explicitamente a operação `RebootDBInstance` da API. Você poderá reinicializar sem failover se a instância de banco de dados estiver em uma implantação multi-AZ. A exigência de reinicializar a instância de banco de dados associada após uma alteração de parâmetro estático ajuda a atenuar o risco de que uma configuração incorreta de parâmetro afete uma chamada de API. Um exemplo disso é chamar `ModifyDBInstance` para alterar a classe de instância de banco de dados. Para ter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Amazon Aurora.

Problemas de memória liberável no Amazon Aurora

Memória liberávelé a memória total de acesso aleatório (RAM) em uma instância de banco de dados que pode ser disponibilizada para o mecanismo de banco de dados. É a soma da memória livre do sistema operacional (SO) e o buffer e a memória cache de página disponíveis. O mecanismo de banco de dados usa a maior parte da memória no host, mas os processos do sistema operacional também usam RAM. A memória atualmente alocada ao mecanismo de banco de dados ou usada pelos processos do sistema operacional não está incluída na memória liberável. Quando o mecanismo de banco de dados está ficando sem memória, a instância de banco de dados pode usar o espaço temporário normalmente usado para buffer e armazenamento em cache. Como mencionado anteriormente, esse espaço temporário está incluído na memória liberável.

Você usa a métrica `FreeableMemory` no Amazon CloudWatch para monitorar a memória liberável. Para ter mais informações, consulte Ferramentas de monitoramento do Amazon Aurora.

Se a instância de banco de dados for executada consistentemente com memória liberável ou usar espaço de troca, pense em aumentar a escala verticalmente para uma classe de instância de banco de dados maior. Para ter mais informações, consulte Classes de instâncias de banco de dados do Amazon Aurora.

Também é possível alterar as configurações de memória. Por exemplo, no Aurora MySQL , você pode ajustar o tamanho do parâmetro `innodb_buffer_pool_size`. Esse parâmetro é definido por padrão como 75% da memória física. Para obter mais dicas sobre solução de problemas do MySQL, consulte How can I troubleshoot low freeable memory in an Amazon RDS para MySQL database? (Como posso solucionar problemas de pouca memória liberável em um banco de dados do Amazon RDS para MySQL?)

No caso do Aurora Serverless v2, `FreeableMemory` representa a quantidade de memória não utilizada que está disponível quando a instância de banco de dados do Aurora Serverless v2 é dimensionada para sua capacidade máxima. Você pode reduzir a escala da instância verticalmente para capacidade relativamente baixa, mas ela ainda relata um valor alto para `FreeableMemory`, porque é possível aumentar a escala da instância verticalmente. Essa memória não está disponível no momento, mas você poderá obtê-la se for necessário.

Para cada unidade de capacidade do Aurora (ACU) em que a capacidade atual está abaixo da capacidade máxima, a `FreeableMemory` aumenta em cerca de 2 GiB. Assim, essa métrica não se aproxima de zero até que a escala da instância de banco de dados seja aumentada na vertical ao nível mais alto possível.

Se essa métrica se aproximar de um valor de `0`, a escala da instância de banco de dados foi aumentada verticalmente o máximo possível. Está se aproximando do limite de sua memória disponível. Considere aumentar a configuração máxima de ACU para o cluster. Se essa métrica se aproximar de um valor de `0` em uma instância de banco de dados do leitor, considere adicionar mais instâncias de banco de dados do leitor ao cluster. Dessa forma, é possível distribuir a parte somente leitura da workload por mais instâncias de banco de dados, reduzindo o uso de memória em cada instância de banco de dados do leitor. Para ter mais informações, consulte Métricas importantes do Amazon CloudWatch para o Aurora Serverless v2.

No caso do Aurora Serverless v1, você pode alterar o intervalo de capacidade para usar mais ACUs. Para obter mais informações, consulte Modificar um cluster de banco de dados do Aurora Serverless v1.

Problemas de replicação no Amazon Aurora MySQL

Alguns problemas de replicação do MySQL também se aplicam ao Aurora MySQL. É possível diagnosticar e corrigir esses problemas.

Tópicos

- Diagnosticar e resolver atrasos entre réplicas de leitura
- Diagnosticar e resolver uma falha de replicação de leitura do MySQL
- Erro de replicação interrompida
- A replicação da réplica de leitura falha ao inicializar a estrutura de metadados

Diagnosticar e resolver atrasos entre réplicas de leitura

Depois de criar uma réplica de leitura MySQL e quando ela estiver disponível, o Amazon RDS primeiro replicará as alterações feitas na instância de banco de dados de origem a partir do momento em que a operação de criação da réplica de leitura foi iniciada. Durante essa fase, o tempo de atraso da replicação para a réplica de leitura será maior que 0. Você pode monitorar esse tempo de atraso no Amazon CloudWatch visualizando a métrica `AuroraBinlogReplicaLag` do Amazon RDS.

A métrica `AuroraBinlogReplicaLag` informa o valor do campo `Seconds_Behind_Master` do comando `SHOW REPLICA STATUS` do MySQL. Para ter mais informações, consulte Instrução SHOW REPLICA STATUS na documentação do MySQL.

Quando a métrica `AuroraBinlogReplicaLag` chega a 0, isso mostra que a réplica alcançou a instância do banco de dados de origem. Se a métrica `AuroraBinlogReplicaLag` retornar -1, a replicação pode não estar ativa. Para solucionar um erro de replicação, consulte Diagnosticar e resolver uma falha de replicação de leitura do MySQL . Um `AuroraBinlogReplicaLag` com um valor de -1 também pode significar que o valor de `Seconds_Behind_Master` não pode ser determinado ou é `NULL`.

> **Note**
>
> As versões anteriores do Aurora MySQL usavam `SHOW SLAVE STATUS` em vez de `SHOW REPLICA STATUS`. Se você estiver utilizando o Aurora MySQL versão 1 ou 2, utilize `SHOW SLAVE STATUS`. Use `SHOW REPLICA STATUS` para o Aurora MySQL versão 3 e posteriores.

A métrica `AuroraBinlogReplicaLag` retorna -1 durante uma interrupção da rede ou quando um patch é aplicado durante a janela de manutenção. Nesse caso, aguarde até que a conectividade de rede seja restaurada ou a janela de manutenção seja finalizada antes de verificar novamente a métrica `AuroraBinlogReplicaLag`.

A tecnologia de replicação de leitura do MySQL é assíncrona. Portanto, é possível esperar aumentos ocasionais da métrica `BinLogDiskUsage` na instância de banco de dados de origem e da métrica `AuroraBinlogReplicaLag` na réplica de leitura. Por exemplo, considere uma situação em que um alto volume de operações de gravação para a instância de banco de dados de origem ocorra em paralelo. Ao mesmo tempo, as operações de gravação na réplica de leitura são serializadas usando um único thread de E/S. Tal situação pode levar a um atraso entre a instância de origem e a réplica de leitura.

Para ter mais informações sobre réplicas de leitura e o MySQL, consulte Detalhes da implementação da replicação na documentação do MySQL. .

É possível reduzir o atraso entre as atualizações em uma instância de banco de dados de origem e as atualizações subsequentes da réplica de leitura fazendo o seguinte:

- Defina a classe da instância de banco de dados da réplica de leitura para que ela tenha um tamanho de armazenamento comparável ao da instância de banco de dados de origem.
- Certifique-se de que as configurações de parâmetros nos grupos de parâmetros do banco de dados utilizados pela instância de banco de dados de origem e pela réplica de leitura sejam compatíveis. Para ter mais informações e um exemplo, consulte a discussão sobre o parâmetro `max_allowed_packet` na próxima seção.
- Desabilite o cache de consulta. Para tabelas que são modificadas com frequência, o uso do cache de consulta pode aumentar o atraso das réplicas, pois o cache é bloqueado e atualizado com frequência. Se esse for o caso, talvez você veja um atraso menor de réplicas se desabilitar o cache de consulta. É possível desabilitar o cache de consultas definindo `query_cache_type parameter` como 0 no grupo de parâmetros de banco de dados da instância de banco de dados. Para ter mais informações sobre o cache de consulta, consulte Configuração do cache de consulta.
- Aqueça o grupo de buffers na réplica de leitura do InnoDB para MySQL. Por exemplo, suponha que você tenha um pequeno conjunto de tabelas sendo atualizadas com frequência e esteja usando o esquema de tabela InnoDB ou XtraDB. Nesse caso, despeje essas tabelas na réplica de leitura. Isso faz com que o mecanismo de banco de dados explore as linhas dessas tabelas no disco e armazene-as em cache no grupo de buffers, o que pode reduzir o atraso das réplicas. Essa abordagem pode reduzir o atraso da réplica. Por exemplo:

Para Linux, macOS ou Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

Para Windows:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnosticar e resolver uma falha de replicação de leitura do MySQL

O Amazon RDS monitora o status de replicação de suas réplicas de leitura. O RDS atualiza o campo Replication State (Estado de replicação) da instância da réplica de leitura para `Error` caso a replicação seja interrompida por qualquer motivo. É possível analisar os detalhes do erro associado lançado pelos mecanismos do MySQL visualizando o campo Replication Error (Erro de replicação). Os eventos que indicam o status da réplica de leitura também são gerados, incluindo RDS-EVENT-0045, RDS-EVENT-0046 e RDS-EVENT-0057. Para ter mais informações sobre eventos e como se inscrever neles, consulte Trabalhar com a notificação de eventos do Amazon RDS. Se for retornada uma mensagem de erro do MySQL, verifique o erro na documentação de mensagens de erro do MySQL. .

Situações comuns que podem causar erros de replicação incluem o seguinte:

- O valor do parâmetro `max_allowed_packet` para uma réplica de leitura é menor que o parâmetro `max_allowed_packet` para a instância do banco de dados de origem.

  O parâmetro `max_allowed_packet` é um parâmetro personalizado que você pode definir em um grupo de parâmetros de banco de dados. O parâmetro `max_allowed_packet` é usado para especificar o tamanho máximo de linguagem de manipulação de dados (DML) que pode

ser executado no banco de dados. Em alguns casos, o valor `max_allowed_packet` para a instância de banco de dados de origem pode ser maior do que o valor `max_allowed_packet` para a réplica de leitura. Se sim, o processo de replicação poderá lançar um erro e interromper a replicação. O erro mais comum é `packet bigger than 'max_allowed_packet' bytes`. É possível corrigir o erro fazendo com que a origem e a réplica de leitura usem grupos de parâmetros de banco de dados com os mesmos valores do parâmetro `max_allowed_packet`.

- A gravação em tabelas em uma réplica de leitura. Se você estiver criando índices em uma réplica de leitura, será necessário que o parâmetro `read_only` seja definido como 0 para criar os índices. Se você estiver gravando em tabelas na réplica de leitura, isso poderá interromper a replicação.
- Uso de um mecanismo de armazenamento não transacional, como o MyISAM. As réplicas de leitura exigem um mecanismo de armazenamento transacional. A replicação só é compatível com os seguintes mecanismos de armazenamento: InnoDB para MySQL ou MariaDB.

  Você pode converter uma tabela MyISAM para o InnoDB com o seguinte comando:

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```
- Usando consultas não deterministas inseguras, como `SYSDATE()`. Para ter mais informações, consulte Determinar instruções seguras e não seguras em registros em logs binários na documentação do MySQL.

As seguintes etapas podem ajudar a resolver seu erro de replicação:

- Se você encontrar um erro lógico e puder ignorar o erro com segurança, siga as etapas descritas em Ignorar o erro de replicação atual. Sua instância de banco de dados do Aurora MySQL deve estar executando uma versão que inclua o procedimento `mysql_rds_skip_repl_error`. Para ter mais informações, consulte mysql_rds_skip_repl_error.
- Se encontrar um problema de posição de log binário, você poderá alterar a posição de reprodução da réplica. Você faz isso com o comando `mysql.rds_next_master_log` do Aurora MySQL versões 1 e 2. Você faz isso com o comando `mysql.rds_next_source_log` do Aurora MySQL versões 3 e posteriores. Sua instância de banco de dados do Aurora MySQL deve estar executando uma versão com suporte a esse comando para alterar a posição de reprodução da réplica. Para obter informações sobre a versão, consulte mysql_rds_next_master_log.
- Se encontrar um problema de performance temporário devido à alta carga DML, você poderá definir o parâmetro `innodb_flush_log_at_trx_commit` como 2 no grupo de parâmetros do banco de dados da réplica de leitura. Fazer isso pode ajudar na recuperação da réplica de leitura,

embora isso reduza temporariamente a atomicidade, a consistência, o isolamento e a durabilidade (ACID).

- É possível excluir a réplica de leitura e criar uma instância usando o mesmo identificador de instância de banco de dados. Dessa forma, o endpoint permanecerá igual ao da réplica de leitura antiga.

Se um erro de replicação for corrigido, o valor de Replication State (Estado de replicação) mudará para replicating (replicando). Para ter mais informações, consulte Solução de problemas da réplica de leitura do MySQL.

Erro de replicação interrompida

Quando você chama o comando `mysql.rds_skip_repl_error`, você pode receber uma mensagem de erro informando que a replicação está inativa ou desabilitada.

Esta mensagem de erro é exibida porque a replicação parou e não foi possível reiniciá-la.

Se você precisar ignorar um grande número de erros, o atraso de replicação pode aumentar além do período de retenção padrão para arquivos de log binário. Nesse caso, você poderá encontrar um erro fatal, com os arquivos do log binário sendo limpos antes de sua reprodução na réplica. Essa remoção faz com que a replicação pare, e você não consegue chamar o comando `mysql.rds_skip_repl_error` para ignorar erros de replicação.

Você pode mitigar esse problema aumentando o número de horas em que os arquivos de log binário são retidos na origem da replicação. Após aumentar o período de retenção de log binário, você pode reiniciar a replicação e chamar o comando `mysql.rds_skip_repl_error` conforme necessário.

Para definir o tempo de retenção do log binário, use o procedimento mysql_rds_set_configuration . Especifique um parâmetro de configuração de "horas de retenção do log binário" juntamente com o número de horas para reter arquivos de log binário no cluster de banco de dados, até 2160 (90 dias). O padrão para Aurora MySQL é 24 (1 dia). O exemplo a seguir define o período de retenção para arquivos de log binário em 48 horas.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

A replicação da réplica de leitura falha ao inicializar a estrutura de metadados

Ao tentar iniciar a replicação, você recebeu a seguinte mensagem de erro:

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

Esse erro ocorre quando há um problema com a estrutura de metadados da réplica. Para corrigir a estrutura de metadados, você deve criar outra réplica.

Para evitar que isso aconteça no futuro, execute uma das seguintes ações:

- Se possível, desabilite o multithreading nas réplicas. A partir do MySQL 8.0.27, o multithreading é habilitado por padrão.
- Se você precisar usar multithreading nas réplicas, recomendamos que use a replicação baseada em GTID. Para obter mais informações, consulte Usar a replicação baseada em GTID.

Referência da API do Amazon RDS

Além do Console de gerenciamento da AWS e do AWS Command Line Interface (AWS CLI), o Amazon RDS também fornece uma API. Você pode usar a API para automatizar tarefas de gerenciamento de suas instâncias de banco de dados e outros objetos no Amazon RDS.

- Para obter uma lista alfabética de operações da API, consulte Ações.
- Para obter uma lista alfabética de tipos de dados, consulte Tipos de dados.
- Para obter uma lista de parâmetros de consulta comuns, consulte Parâmetros comuns.
- Para obter descrições dos códigos de erro, consulte Erros comuns.

Para obter mais informações sobre a AWS CLI, consulte a Referência da AWS Command Line Interface para o Amazon RDS.

Tópicos

- Uso da API de consulta
- Solução de problemas de aplicações no Aurora

Uso da API de consulta

As seções a seguir discutem brevemente os parâmetros e a autenticação de solicitação usados com a API de consulta.

Para obter informações gerais sobre como a API de consulta funciona, consulte Solicitações de consulta no Amazon EC2 API Reference.

Parâmetros de consulta

As solicitações baseadas em consulta HTTP são solicitações HTTP que usam o verbo HTTP GET ou POST e um parâmetro de consulta chamado `Action`.

Cada solicitação de consulta deve incluir alguns parâmetros comuns para lidar com a autenticação e a seleção de uma ação.

Algumas operações levam listas de parâmetros. Essas listas são especificadas usando a notação `param.`*n*. Os valores de *n* são inteiros a partir de 1.

Para ter informações sobre as regiões e os endpoints do Amazon RDS, acesse Amazon Relational Database Service (RDS) na seção Regiões e endpoints da Referência geral da Amazon Web Services.

Autenticação de solicitação de consulta

Só é possível enviar solicitações de consulta por meio de HTTPS, e é preciso incluir uma assinatura em todas as solicitações de consulta. Você deve usar a versão de assinatura 4 da AWS ou a versão de assinatura 2. Para obter informações, consulte Processo de assinatura do Signature versão 4 e Processo de assinatura do Signature versão 2.

Solução de problemas de aplicações no Aurora

O Amazon RDS fornece erros específicos e descritivos para ajudar você a solucionar problemas enquanto interage com a API do Amazon RDS.

Tópicos

- Recuperação de erros
- Dicas de solução de problemas

Para obter informações sobre solução de problemas para instâncias de banco de dados do Amazon RDS, consulte Solução de problemas do Amazon Aurora.

Recuperação de erros

Normalmente, espera-se que o aplicativo verifique se uma solicitação gerou um erro antes que você precise processar os resultados. A maneira mais fácil de descobrir se ocorreu um erro é procurar por um nó `Error` na resposta da API do Amazon RDS.

A sintaxe XPath apresenta uma maneira simples de procurar pela presença de um nó de `Error`. Ela também fornece uma maneira relativamente fácil de recuperar o código e a mensagem de erro. O snippet de código a seguir usa Perl e o módulo XML::XPath para determinar se ocorreu um erro durante uma solicitação. Caso tenha ocorrido, o código imprimirá o primeiro código de erro e a mensagem na resposta.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
```

```
    {print "There was an error processing your request:\n", " Error code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Dicas de solução de problemas

Recomendamos os seguintes processos para diagnosticar e resolver problemas com a API do Amazon RDS:

- Confirme se o Amazon RDS está funcionando normalmente na região da AWS que você está almejando acessando http://status.aws.amazon.com.
- Verificação da estrutura de sua solicitação.

  Cada operação do Amazon RDS tem uma página de referência na Amazon RDS Referência da API. Verifique novamente se você está usando os parâmetros corretamente. Para ter ideias sobre o que pode estar errado, consulte as amostras de solicitações ou cenários de usuários para ver se esses exemplos realizam operações similares.
- Confira o AWS re:Post.

  O Amazon RDS conta com um fórum de comunidade de desenvolvimento onde você pode procurar soluções para os problemas que outros enfrentaram ao longo do caminho. Para visualizar os tópicos, acesse AWS re:Post.

Histórico do documento

Versão atual da API: 2014-10-31

A tabela a seguir descreve as alterações importantes feitas no Manual do usuário do Amazon Aurora. Para receber notificações sobre atualizações dessa documentação, você pode se inscrever em um feed RSS. Para obter informações sobre o Amazon Relational Database Service (Amazon RDS), consulte o Guia do usuário do Amazon Relational Database Service.

> **Note**
>
> Antes de 31 de agosto de 2018, a documentação do Amazon Aurora fazia parte do Manual do usuário do Amazon Relational Database Service. Para ver o histórico anterior de documentos do Aurora, consulte Histórico de documentos no Manual do usuário do Amazon Relational Database Service.

Você pode filtrar novos recursos do Amazon Aurora na página What's New with Database? (Novidades sobre bancos de dados). Em Products (Produtos), escolha Amazon Aurora. Em seguida, pesquise usando palavras-chave como **`global database`** ou **`Serverless`**.

Alteração	Descrição	Data
As integrações ETL zero do Aurora PostgreSQL com Amazon SageMaker comportam regiões adicionais	As Integrações ETL zero do Aurora PostgreSQL com Amazon SageMaker já estão disponíveis em regiões adicionais. Para acessar mais informações, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com integrações ETL zero.	15 de outubro de 2025
O Aurora PostgreSQL Limitless Database está disponível em uma região adicional da AWS	O Aurora PostgreSQL Limitless Database agora está disponível nas AWS GovCloud (US) Regions. Para acessar	16 de setembro de 2025

	mais informações, consulte Requisitos e considerações do Aurora PostgreSQL Limitless Database.	
A API de dados do Amazon RDS comporta IPv6	A API de dados do Amazon RDS agora comporta a conectividade IPv6, permitindo que você se conecte aos seus bancos de dados Aurora usando endereços IPv6. Para acessar mais informações, consulte Usar a API de dados.	30 de agosto de 2025
O Amazon Aurora está disponível na região da Ásia-Pacífico (Nova Zelândia)	O Amazon Aurora já está disponível na região da Ásia-Pacífico (Nova Zelândia). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	28 de agosto de 2025
A API de dados do Amazon RDS está disponível em uma região adicional da AWS	A API de dados do Amazon RDS agora está disponível na região da Europa (Espanha). Para ter mais informações, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com a API de dados do RDS.	1.º de agosto de 2025

As Integrações ETL zero com o lakehouse do Amazon SageMaker estão disponíveis ao público	As Integrações ETL zero disponibilizam dados transacionais no Amazon SageMaker segundos depois que eles são gravados em uma instância de banco de dados do Aurora MySQL. Para ter mais informações, consulte Integrações ETL zero do Aurora.	30 de junho de 2025
O Amazon Aurora está disponível na região da Ásia-Pacífico (Taipei)	O Amazon Aurora já está disponível na região da Ásia-Pacífico (Taipei). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	5 de junho de 2025
Informações sobre o fim da vida útil do Insights de Performance	A AWS anunciou a data de fim da vida útil do Insights de Performance: 30 de novembro de 2025. Após essa data, o Amazon RDS não oferecerá mais suporte à experiência do console, aos períodos de retenção flexíveis e aos preços associados do Insights de Performance. Recomendamos atualizar seus clusters de banco de dados para o Database Insights antes dessa data. Para ter mais informações, consulte Monitorar a carga de banco de dados com o Performance Insights no Amazon Aurora.	30 de maio de 2025

O Aurora PostgreSQL Limitless Database é compatível com o CloudWatch Database Insights	Agora é possível usar o CloudWatch Database Insights para monitorar o Amazon Aurora PostgreSQL Limitless Database em grande escala. Para ter mais informações, consulte Monitorar o Aurora PostgreSQL Limitless Database com o CloudWatch Database Insights.	22 de maio de 2025
Visualizar datas de suporte de versões principais de mecanismo de código aberto	Agora você pode visualizar informações sobre as datas de início e término de versões principais de mecanismo de código aberto no suporte padrão do RDS e no Suporte estendido do RDS usando a AWS CLI e a API do RDS. Para obter mais informações, consulte Visualizar datas de suporte de versões de mecanismo no Suporte estendido do Amazon RDS.	20 de maio de 2025
É possível usar Integrações ETL zero do Aurora PostgreSQL em regiões adicionais	Agora as Integrações ETL zero do Aurora PostgreSQL estão disponíveis em regiões adicionais. Para ter mais informações, consulte Regiões e mecanismos de banco de dados do Aurora compatíveis com integrações ETL zero com o Amazon Redshift.	17 de fevereiro de 2025

O Amazon Aurora está disponível na região do México (Centro).	O Amazon Aurora já está disponível na região do México (Centro). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	14 de janeiro de 2025
O Amazon Aurora está disponível na região da Ásia-Pacífico (Tailândia).	O Amazon Aurora já está disponível na região da Ásia-Pacífico (Tailândia). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	7 de janeiro de 2025

Informações sobre o fim da vida útil do Aurora Serverless v1	A AWS anunciou a data de fim da vida útil do Aurora Serverless v1: 31 de março de 2025. Recomendamos atualizar quaisquer clusters de banco de dados do Aurora Serverless v1 para Aurora Serverless v2 antes dessa data. A atualização pode envolver uma alteração no número da versão principal do mecanismo de banco de dados. Portanto, é important e planejar, testar e implement ar essa transição antes da data de fim da vida útil. A partir de 8 de janeiro de 2025, os clientes não poderão mais criar novos clusters ou instâncias do Aurora Serverless v1 com o Console de gerenciamento da AWS ou a CLI. Para obter informaçõ es sobre o procedimento de atualização de Aurora Serverless v1 para Aurora Serverless v2, consulte Atualização de um cluster de Aurora Serverless v1 para Aurora Serverless v2.	06 de dezembro de 2024

O Aurora é compatível com o CloudWatch Database Insights	Agora é possível usar o CloudWatch Database Insights para monitorar seus clusters de banco de dados do Amazon Aurora em grande escala. Para obter mais informações, consulte Monitoramento de bancos de dados do Amazon Aurora com o CloudWatch Database Insights.	1.º de dezembro de 2024
O Amazon Aurora é compatível com as classes de instância db.r8g	Agora você pode usar as classes de instância db.r8g para criar clusters de banco de dados do Aurora. Para obter mais informações, consulte Classes de instância de banco de dados do Aurora.	21 de novembro de 2024
Aurora Serverless v2 O oferece suporte a pausa e retomada automáticas	As instâncias de banco de dados do Aurora Serverless v2 agora podem ser reduzidas até zero unidades de capacidade Aurora (ACUs) após um período de inatividade. Para habilitar esse recurso, defina o valor de capacidade mínima como 0 ACUs. Para obter mais informações, consulte Pausa e retomada automáticas para Aurora Serverless v2.	20 de novembro de 2024

O Amazon Aurora oferece suporte à migração automática de bancos de dados do EC	Você pode usar o console do RDS para migrar um banco de dados do EC2 para o Aurora. O Aurora usa o AWS Database Migration Service (AWS DMS) para migrar seu banco de dados do EC2 de origem. O AWS DMS permite que você migre bancos de dados relacionais para a Nuvem AWS. Para obter mais informações, consulte Migração automática de bancos de dados do EC2 para o Amazon Aurora usando o AWS Database Migration Service.	20 de novembro de 2024
AWS Wrapper avançado de NodeJS da geralmente disponível	O wrapper avançado de NodeJS da Amazon Web Services (AWS) complemen ta e amplia a funcionalidade de um driver NodeJS existente . Para ter mais informações, consulte Connecting to Aurora DB clusters with the AWS drivers.	19 de novembro de 2024
O Amazon Aurora é compatíve l com as classes de instância db.r7i	Agora você pode usar as classes de instância db.r7i para criar clusters de banco de dados do Aurora. Para obter mais informações, consulte Classes de instância de banco de dados do Aurora.	18 de novembro de 2024

A API de dados do RDS para Aurora MySQL é compatível com todas as visualizações do Insights de Performance	Agora é possível usar as visualizações Principais hosts ou Principais aplicações do Insights de Performance para monitorar as operações SQL realizadas usando a API de dados do RDS para o Aurora MySQL. Consulte informaçõ es sobre o suporte do Insights de Performance em Monitorin g RDS Data API queries with Performance Insights.	31 de outubro de 2024
Aurora PostgreSQL Limitless Database	O Amazon Aurora PostgreSQ L Limitless Database é um novo recurso automatizado de escalabilidade horizonta l (fragmentação) do Amazon Aurora. O Aurora PostgreSQ L Limitless Database permite que você escale além dos limites existentes do Aurora para throughput de gravação e armazenamento, distribui ndo um workload de banco de dados em várias instância s de gravação do Aurora, mantendo a capacidade de usá-lo como um único banco de dados. Para obter mais informações, consulte Amazon Aurora PostgreSQL Limitless Database.	31 de outubro de 2024

Fim do suporte padrão do Amazon Aurora MySQL versão	O Aurora MySQL versão 2 atingiu o fim do suporte padrão em 31 de outubro de 2024. Para obter mais informações, consulte Preparação para o fim do suporte padrão do Amazon Aurora, edição compatível com MySQL versão 2.	31 de outubro de 2024
O Amazon Aurora oferece suporte a atualizações de sistema operacional no nível do cluster	Agora você pode aplicar atualizações do sistema operacional do Aurora no nível do cluster de banco de dados. As atualizações contínuas aplicam automaticamente as atualizações a algumas instâncias de banco de dados de leitura por vez, preservando a disponibilidade de leitura. Para obter informações, consulte Atualizações do sistema operacional para clusters de banco de dados do Aurora.	30 de outubro de 2024

Clusters globais do Aurora aceitam marcação	Já é possível adicionar etiquetas aos clusters globais do Aurora. Um cluster global do Aurora é o recurso pai que contém os clusters de banco de dados primários e secundários em um Aurora Global Database. Para ter informações gerais sobre marcação, consulte Marcar recursos do Amazon Aurora e do Amazon RDS. Para ter informações sobre marcação com o Aurora Global Database, consulte Tagging Aurora Global Database resources.	24 de outubro de 2024
Endpoints do gravador do Aurora Global Database	Cada Aurora Global Database vem com um endpoint do gravador que é atualizado automaticamente pelo Aurora para direcionar solicitações à instância atual do gravador do cluster de banco de dados primário após uma operação de transição e de failover do Aurora Global Database. Para saber mais sobre como usar o endpoint do gravador com a transição e o failover do Aurora Global Database, consulte Usar a transição ou o failover em um Amazon Aurora Global Database.	22 de outubro de 2024

Disponibilidade geral de integrações ETL zero do Aurora PostgreSQL com o Amazon Redshift	As integrações ETL zero disponibilizam dados transacionais no Amazon Redshift em questão de segundos depois de serem gravados em um cluster de banco de dados de origem do Aurora PostgreSQL. O recurso já está disponível ao público. Para ter mais informações, consulte Integrações ETL zero do Aurora com o Amazon Redshift.	15 de outubro de 2024
Aurora Serverless v2 O comporta 256 ACUs	Agora é possível criar clusters de banco de dados do Aurora Serverless v2 com capacidade máxima de 256 unidades de capacidade do Aurora (ACUs). Consulte mais informações em Capacidade do Aurora Serverless v2.	3 de outubro de 2024
O Amazon Aurora é compatível com o Console-to-Code	Agora é possível usar o Console-to-Code para gerar código por meio das ações que você executa no console do RDS. O código gerado pode ajudar você a escrever código para automatizar o uso de outros serviços da AWS. Consulte mais informações em Use Console-to-Code to generate code for your Aurora console actions.	3 de outubro de 2024

A API de dados do RDS é compatível com o Insights de Performance	Agora é possível usar o Insights de Performance para monitorar as operações de SQL realizadas usando a API de dados do RDS. Consulte informações sobre o suporte do Insights de Performance em Monitoring RDS Data API queries with Performance Insights.	26 de setembro de 2024
A API de dados do RDS para o Aurora Serverless v2 e provisionado está disponível para o Aurora MySQL	A API de dados do RDS agora está disponível para clusters do Aurora MySQL que usam o Aurora Serverless v2 ou instâncias provisionadas. Consulte informações sobre o uso da API de dados do RDS em Usar a API de dados do RDS. Consulte informações sobre o suporte de versão e região da AWS do Aurora MySQL para a API de dados do RDS em API de dados do RDS para o Aurora PostgreSQL Sem Servidor v2 e provisionado.	26 de setembro de 2024
O Amazon Aurora está disponível na região Ásia-Pacífico (Malásia	O Amazon Aurora já está disponível na região Ásia-Pacífico (Malásia). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	22 de agosto de 2024

atualizar para a política existente	O Amazon RDS removeu a permissão `sns:Publish` da `AmazonRDSBetaServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDSBeta` . Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	7 de agosto de 2024
atualizar para a política existente	O Amazon RDS removeu a permissão `sns:Publish` da `AmazonRDSPreviewServiceRolePolicy` do perfil vinculado ao serviço `AWSServiceRoleForRDSPreview` . Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	7 de agosto de 2024
Aurora Serverless v2 O oferece suporte para pausa automática (escala até zero)	As instâncias de banco de dados do Aurora Serverless v2 agora podem ser reduzidas até zero unidades de capacidade Aurora (ACUs) após um período de inatividade. Para habilitar esse recurso, defina o valor de capacidade mínima como 0. Para obter mais informações, consulte Usar o Aurora Serverless v2.	25 de julho de 2024

AWS Driver ODBC para MySQL da disponível para o público em geral	O Driver ODBC para MySQL da Amazon Web Services (AWS) é um driver de cliente projetado para a alta disponibilidade do Aurora MySQL. Para ter mais informações, consulte Connecting to Aurora MySQL with the Amazon Web Services (AWS) ODBC Driver for MySQL.	18 de julho de 2024
A API de dados do RDS para Aurora Serverless v2 está disponível em mais regiões	A API de dados do RDS agora está disponível para o Aurora PostgreSQL em várias regiões adicionais da AWS. Para ter informações sobre o suporte regional da API Data do RDS, consulte API de dados do RDS para o Aurora PostgreSQL Sem Servidor v2 e provisionado.	9 de julho de 2024
AWS Driver Python da disponível ao público	O driver Python da Amazon Web Services (AWS) foi projetado como um wrapper Python avançado. Esse wrapper é complementar e amplia a funcionalidade do driver Psycopg de código aberto. Para ter mais informações, consulte Connecting to Aurora DB clusters with the AWS drivers.	23 de maio de 2024

Integrações ETL zero disponíveis nas regiões da China	Integrações ETL zero agora estão disponíveis nas Regiões da AWS da China (Pequim) e da China (Ningxia) . Para ter mais informações, consulte Integrações ETL zero do Amazon Aurora com Amazon Redshift.	21 de maio de 2024
O RDS Proxy está disponível em mais regiões	O RDS Proxy agora está disponível nas seguintes regiões: Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Melbourne), Oriente Médio (EAU), Israel (Tel Aviv), Oeste do Canadá (Calgary) e Europa (Zurique). Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	21 de maio de 2024

O Amazon RDS comporta acesso refinado para o Insights de Performance	Agora é possível permitir ou negar o acesso a dimensões individuais no Insights de Performance. Esse acesso refinado pode ser usado para as ações `GetResourceMetrics` , `DescribeDimensionKeys` e `GetDimensionKeyDetails` . Para obter mais informações, consulte https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.access-control.dimensionAccess-policy.html.	21 de maio de 2024

Suporte estendido do Amazon RDS	A criação ou restauração de um banco de dados do Aurora MySQL versão 2 ou 3 ou Aurora PostgreSQL versão 11 agora inscreve automaticamente esse banco de dados no Suporte estendido do Amazon RDS para que as aplicações existentes continuem funcionando como estão. É possível desativar o Suporte estendido do RDS para evitar cobranças após a data de término do suporte padrão do Aurora para o mecanismo de banco de dados. Para obter mais informações, consulte Usar o suporte estendido do Amazon RDS.	21 de março de 2024
Filtragem de dados para integrações ETL zero	O Amazon RDS é compatível com a filtragem de dados no nível do banco de dados e da tabela para integrações ETL zero com o Amazon Redshift. Consulte mais informações em Data filtering for Aurora zero-ETL integrations with Amazon Redshift.	20 de março de 2024

Integrações do Aurora MySQL com o Amazon Bedrock	Agora é possível integrar bancos de dados do Amazon Aurora MySQL com o Amazon Bedrock para potencializar aplicações de IA generativa. Consulte mais informações em Usar o machine learning do Amazon Aurora com o Aurora MySQL.	8 de março de 2024
Nova política gerenciada pela AWS	O Amazon RDS adicionou uma nova política gerenciada chamada `AmazonRDSCustomInstanceProfileRolePolicy` para permitir que o RDS Custom execute ações de automação e tarefas de gerenciamento de banco de dados por meio de um perfil de instância do EC2. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	27 de fevereiro de 2024
Suporte do Amazon RDS para AWS Secrets Manager na região de Israel (Tel Aviv)	O Amazon RDS é compatível com o Secrets Manager na região de Israel (Tel Aviv). Para ter mais informações, consulte Gerenciamento de senhas com o Amazon RDS e o AWS Secrets Manager.	21 de fevereiro de 2024

Suporte estendido do Amazon RDS	O Amazon RDS agora habilita automaticamente o Suporte estendido do Amazon RDS quando as versões principais do mecanismo do Aurora MySQL e do Aurora PostgreSQL nos clusters de banco de dados e clusters globais atingem a data de término do suporte padrão do Aurora. Para obter mais informações, consulte Usar o suporte estendido do Amazon RDS.	15 de fevereiro de 2024
O Aurora PostgreSQL 16.1 é compatível com o Babelfish para Aurora PostgreSQL 4.0.0	O Aurora PostgreSQL 16.1 é compatível com o Babelfish 4.0.0. Consulte uma lista de novos recursos em 16.1. Para ver uma lista de recursos compatíveis em cada versão do Babelfish, consulte Funcionalidade compatível no Babelfish por versão. Para ter mais informações sobre uso, consulte o tópico Trabalhar com o Babelfish for Aurora PostgreSQL.	31 de janeiro de 2024
Atualização para o certificado CA padrão	O certificado CA padrão está definido como `rds-ca-rsa2048-g1` . Para ter mais informações, consulte Como usar SSL/TLS para criptografar uma conexão com um cluster de banco de dados.	26 de janeiro de 2024

O RDS Proxy está disponível na região da Europa (Espanha	O RDS Proxy agora está disponível na região da Europa (Espanha). Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	8 de janeiro de 2024
API de dados do RDS com o Aurora PostgreSQL Sem Servidor v2 e provisionado	Agora é possível usar a API de dados do RDS com o Aurora PostgreSQL Sem Servidor v2 e clusters de banco de dados provision ados. Com a API de dados do RDS, é possível acessar os clusters do Aurora por meio de um endpoint HTTP seguro e executar declarações SQL sem usar drivers de banco de dados nem gerenciar conexões. Para ter mais informações, consulte Usar a API de dados do RDS.	21 de dezembro de 2023
Integrações do Aurora PostgreSQL com o Amazon Bedrock	Agora é possível integrar bancos de dados do Amazon Aurora PostgreSQL com o Amazon Bedrock para potencializar aplicações de IA generativa. Para ter mais informações, consulte Usar machine learning do Amazon Aurora com o Aurora PostgreSQL.	21 de dezembro de 2023

O Amazon Aurora está disponível na região do Oeste do Canadá (Calgary	O Amazon Aurora agora está disponível na região do Oeste do Canadá (Calgary). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	20 de dezembro de 2023
O Amazon RDS é compatível com a visualização e a resposta às recomendações	As recomendações do Amazon Aurora agora incluem recomendações proativas baseadas em limites e recomendações reativas baseadas em machine learning. Para ter mais informações, consulte Recommendations for Amazon Aurora.	19 de dezembro de 2023
Integrações ETL zero do Aurora PostgreSQL com o Amazon Redshift (pré-visualização)	Agora é possível criar integrações ETL zero com o Amazon Redshift usando um cluster de banco de dados de origem do Aurora PostgreSQL. Para a versão de pré-visualização, é necessário criar todas as integrações no Ambiente de Pré-visualização do Banco de Dados do Amazon RDS, na Região da AWS do Leste dos EUA (Ohio) (us-east-2). Para ter mais informações, consulte Integrações ETL zero do Aurora com o Amazon Redshift.	28 de novembro de 2023

Adição de suporte ao banco de dados global do Amazon Aurora PostgreSQL	Agora é possível habilitar o encaminhamento de gravação em clusters secundários em um banco de dados global baseado no Aurora. Para obter mais informações, consulte Using write forwarding in an Aurora PostgreSQL global database.	9 de novembro de 2022
Suporte ao Aurora PostgreSQL para leituras otimizadas	É possível acelerar o processamento de consultas para o Aurora PostgreSQL com o Aurora Optimized Reads. Para ter mais informações, consulte Melhorar a performance das consultas com o Aurora Optimized Reads.	8 de novembro de 2022

O Amazon RDS publica métricas do contador do Insights de Performance para o Amazon CloudWatch	O Performance Insights permite exportar os painéis de métricas pré-configurados ou personalizados para o Amazon CloudWatch. Os painéis de métricas exportados estão disponíveis para visualização no console do CloudWatch. Você também pode exportar um widget de métrica do Performance Insights selecionado e visualizar os dados de métricas no console do CloudWatch. Para ter mais informações, consulte Métricas do Performance Insights publicadas no CloudWatch.	8 de novembro de 2022
Disponibilidade geral de integrações ETL zero do Aurora com o Amazon Redshift	Integrações ETL zero com o Amazon Redshift agora estão disponíveis ao público em geral para o Aurora MySQL. Para ter mais informações, consulte Integrações ETL zero do Aurora com o Amazon Redshift.	7 de novembro de 2022

Suporte do Aurora para implantações azul/verde do RDS	Agora você pode criar uma implantação azul/verde a partir de um cluster de banco de dados Aurora PostgreSQL. Para ter mais informações, consulte Using Amazon RDS Blue/Green Deployments for database updates (Usar implantações azul/verde do Amazon RDS para atualizações de banco de dados).	26 de outubro de 2023
O Aurora MySQL é compatível com criptografia do servidor com o AWS KMS keys (SSE-KMS)	No Aurora MySQL versão 3.05 e superior, você pode usar o SSE-KMS, incluindo Chaves gerenciadas pela AWS e chaves gerenciadas pelo cliente, para criptografia do lado do servidor dos dados que você carrega ou salva no Amazon S3. Para obter mais informações, consulte Carregar dados em um cluster de banco de dados Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3 e Salvar dados de um cluster de banco de dados Amazon Aurora MySQL a partir de arquivos de texto em um bucket do Amazon S3.	20 de outubro de 2023

As otimizações do Aurora MySQL reduzem o tempo de reinicialização do banco de dados	No Aurora MySQL versão 3.05 e superior, introduzimos otimizações que reduzem o tempo de reinicialização do banco de dados. Essas otimizações fornecem até 65% menos tempo de inatividade do que sem otimizações e menos interrupções nas workloads do banco de dados após uma reinicialização. Para obter mais informações, consulte Optimizations to reduce database restart time.	20 de outubro de 2023
Atualizações do AWS para políticas gerenciadas pela	As `AmazonRDSPerformanceInsightsReadOnly` políticas `AmazonRDSPerformanceInsightsFullAccess` gerenciadas agora incluem `Sid` (ID da instrução) como identificador na declaração de política. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	23 de outubro de 2023

O Amazon RDS publica métricas do contador do Insights de Performance para o Amazon CloudWatch	A função matemática métrica DB_PERF_INSIGHTS no console do CloudWatch permite que você consulte métricas do contador do Amazon RDS do Insights de Performance. Para obter mais informações, consulte Criação de alarmes CloudWatch para monitorar Amazon Aurora.	20 de setembro de 2023
O Amazon Aurora agora comporta recuperação para um ponto no tempo (PITR) com o AWS Backup	Agora você pode gerenciar backups automatizados (contínuos) do Aurora no AWS Backup e restaurar para horários especific ados com base neles. Para obter informações, consulte Restaurar um cluster de banco de dados em um horário especificado usando o AWS Backup.	7 de setembro de 2023
Suporte estendido do Amazon RDS	O Amazon Aurora anuncia a possibilidade futura de continuar executando as versões principais do mecanismo do Aurora MySQL e do Aurora PostgreSQL nas instâncias de banco de dados após a data de término do suporte padrão do Aurora. Para obter mais informaçõ es, consulte Usar o suporte estendido do Amazon RDS.	1º de setembro de 2023

O Amazon Aurora MySQL amplia o suporte para o Percona XtraBackup	Agora você pode realizar migrações físicas de bancos de dados MySQL 8.0 para clusters de banco de dados do Aurora MySQL versão 3. Para obter mais informaçõ es, consulte Migração física do MySQL usando o Percona XtraBackup e o Amazon S3.	24 de agosto de 2023
O banco de dados global do Aurora agora comporta failover de banco de dados global	O banco de dados global do Aurora agora comporta failover global gerenciad o, permitindo que você se recupere com maior facilidad e de um desastre regional real ou de uma interrupç ão total no nível de serviço. Para saber mais sobre esse recurso, consulte Executar failovers gerenciados para bancos de dados globais do Aurora. O recurso anteriorm ente denominado “failover planejado gerenciado” agora é chamado de “transição”. Para obter informações sobre transições, consulte Realizar transições para o Amazon Aurora Global Database.	21 de agosto de 2023

Atualização das permissões de políticas gerenciadas pela AWS	A política gerenciada `AmazonRDSFullAccess` tem novas permissões que autorizam gerar, visualizar e excluir o relatório de análise de performance por um período. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	17 de agosto de 2023
Atualização das permissões de políticas gerenciadas pela AWS	A adição de novas permissões à política gerenciada `AmazonRDSPerformanceInsightsReadOnly` e a adição de uma nova política gerenciada `AmazonRDSPerformanceInsightsFullAccess` permite gerar um relatório de análise de carga do banco de dados por um período. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	16 de agosto de 2023

O Amazon RDS é compatível com a análise do período de carga do banco de dados com o Insights de Performance	O Insights de Performance permite que você crie relatórios de análise de performance por um período específico. O relatório fornece as recomendações e os insights identificados para resolver os problemas de performance. Para receber mais informações, consulte Analisar a carga do banco de dados por um período.	16 de agosto de 2023
O Amazon Aurora é compatível com a retenção de backups automatizados para clusters de banco de dados	Agora você pode reter backups automatizados para clusters do Aurora excluídos e restaurá-los em um momento específico. Para receber mais informações, consulte Reter backups automatizados.	4 de agosto de 2023
O Amazon Aurora está disponível na região de Israel (Tel Aviv	O Amazon Aurora está disponível na região de Israel (Tel Aviv). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	1º de agosto de 2023

O Amazon Aurora MySQL é compatível com o encaminhamento de gravação local (em cluster	Agora você pode encaminhar operações de gravação de uma instância de banco de dados do leitor para uma instância de banco de dados do gravador em um cluster de banco de dados do Aurora MySQL. Para receber mais informações, consulte Usar o encaminhamento de gravação em um cluster de banco de dados do Amazon Aurora MySQL.	31 de julho de 2023
Amazon Aurora oferece suporte ao Aurora Serverless v2 em mais uma Região da AWS	Agora é possível criar clusters de banco de dados do Aurora Serverless v2 na Região da AWS Ásia-Pacífico (Melbourne). Para obter mais informações sobre o Aurora Serverless v2, consulte Usar Aurora Serverless v2.	28 de junho de 2023
O Amazon Aurora apresenta integrações ETL zero com o Amazon Redshift (pré-visualização)	As integrações ETL zero oferecem uma solução totalmente gerenciada para disponibilizar dados transacionais no Amazon Redshift em questão de segundos depois de serem gravados em um cluster de banco de dados do Aurora MySQL. Para ter mais informações, consulte Integrações ETL zero do Aurora com o Amazon Redshift.	28 de junho de 2023

O Amazon RDS fornece uma visualização combinada de métricas do Insights de Performance e do CloudWatch no painel do Insights de Performance	O Amazon RDS agora fornece uma visão consolidada das métricas do Insights de Performance e do CloudWatch no painel do Insights de Performance. Para ter mais informações, consulte Viewing combined metrics with the Performance Insights dashboard.	24 de maio de 2023
O Amazon Aurora é compatível com as classes de instância db.r7g	Agora você pode usar as classes de instância db.r7g para criar clusters de banco de dados do Aurora. Para obter mais informações, consulte Classes de instância de banco de dados do Aurora.	11 de maio de 2023
O Amazon Aurora é compatível com uma nova configuração de armazenamento do cluster de banco de dados	Com o Aurora I/O-Optimized, você paga somente pelo uso e armazenamento de seus clusters de banco de dados, sem custos adicionais pelas operações de E/S de leitura e gravação. Para obter mais informações, consulte Configurações de armazenamento para clusters de banco de dados do Amazon Aurora.	11 de maio de 2023

Amazon Aurora oferece suporte ao Aurora Serverless v2 em Regiões da AWS adicionais	Agora, é possível criar clusters de bancos de dados do Aurora Serverless v2 nas seguintes Regiões da AWS: Ásia-Pacífico (Hyderabad), Europa (Espanha), Europa (Zurique), Oriente Médio (EAU). Para obter mais informações sobre o Aurora Serverless v2, consulte Usar Aurora Serverless v2.	4 de maio de 2023
Aurora Serverless v1 O é compatível com a conversão em provisionado	É possível converter um cluster de banco de dados do Aurora Serverless v1 em um cluster de banco de dados provisionado. Para ter mais informações, consulte Converter um cluster de banco de dados do Aurora Serverless v1 em provisionado.	27 de abril de 2023
Aurora Serverless v1 O é compatível com o Amazon Aurora PostgreSQL versão 13	Agora você pode criar clusters de banco de dados do Aurora Serverless v1 que executam o Aurora PostgreSQL versão 13. Para obter mais informações, consulte Aurora Serverless v1.	27 de abril de 2023
Suporte do Amazon Aurora para AWS Secrets Manager nas regiões da China	O Amazon Aurora é compatível com o Secrets Manager nas regiões China (Pequim) e China (Ningxia). Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.	20 de abril de 2023

O Amazon Aurora é compatível com a publicação de eventos com tags para os assinantes de tópicos	As notificações de eventos do Amazon Aurora enviadas ao Amazon Simple Notification Service (Amazon SNS) ou ao Amazon EventBridge agora contêm tags de eventos no corpo da mensagem. Essas tags fornecem dados sobre o recurso que foi afetado pelo evento do serviço. Para ter mais informações, consulte Tags e atributos de notificações de eventos do Amazon RDS.	17 de abril de 2023
Atualizar permissões de perfil vinculado a serviços do IAM	As políticas `AmazonRDSFullAccess` e `AmazonRDSReadOnlyAccess` agora concedem permissões adicionais para possibilitar a exibição das descobertas do Amazon DevOps Guru no console do RDS. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	30 de março de 2023
O Amazon Aurora é compatível com bancos de dados globais na região Ásia-Pacífico (Melbourne	Agora você pode criar bancos de dados globais do Aurora na região Ásia-Pacífico (Melbourne). Para ter informações sobre os bancos de dados globais do Aurora, consulte Usar bancos de dados globais do Amazon Aurora.	22 de março de 2023

Atualização das permissões de políticas gerenciadas pela AWS	As políticas `AmazonRDSFullAccess` e `AmazonRDSReadOnlyAccess` agora concedem permissões adicionais ao Amazon CloudWatch. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	16 de março de 2023
O RDS Proxy está disponível nas regiões da China	O RDS Proxy agora está disponível nas regiões China (Pequim) e China (Ningxia). Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	15 de março de 2023
O Amazon Aurora é compatível com o Aurora Serverless v2 nas regiões da China	O Aurora Serverless v2 agora está disponível nas regiões China (Pequim) e China (Ningxia). Para obter mais informações, consulte Aurora Serverless v2.	15 de março de 2023
O RDS Proxy está disponível na região Ásia-Pacífico (Jacarta	O RDS Proxy agora está disponível na região Ásia-Pacífico (Jacarta). Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	8 de março de 2023

O Amazon Aurora MySQL é compatível com a autenticação Kerberos	Agora é possível usar a autenticação Kerberos para autenticar usuários quando eles se conectam aos clusters de bancos de dados do Aurora MySQL. Para ter mais informações, consulte Usar a autenticação Kerberos para Aurora MySQL.	8 de março de 2023
O Amazon Aurora é compatível com bancos de dados globais em outras Regiões da AWS	Agora você pode criar bancos de dados globais do Aurora nas seguintes regiões: África (Cidade do Cabo), Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Europa (Milão), Europa (Espanha), Europa (Zurique), Oriente Médio (Bahrein) e Oriente Médio (EAU). Para ter informações sobre os bancos de dados globais do Aurora, consulte Usar bancos de dados globais do Amazon Aurora.	6 de março de 2023

O Amazon Aurora é compatível com a cópia de snapshots de clusters de banco de dados em outras Regiões da AWS	Agora você pode copiar snapshots de cluster de banco de dados nas seguintes regiões: África (Cidade do Cabo), Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Europa (Milão), Europa (Espanha), Europa (Zurique), Oriente Médio (Bahrein) e Oriente Médio (EAU). Para ter informações sobre a cópia de snapshots de cluster de banco de dados, consulte Copiar um snapshot de cluster de banco de dados.	6 de março de 2023
Amazon DevOps Guru para RDS oferece suporte a insights proativos	O Amazon DevOps Guru para RDS publica insights proativos com recomendações para ajudar você a resolver problemas em seus bancos de dados Aurora antes que eles aconteçam. Para ter mais informações, consulte Como o DevOps Guru para RDS funciona.	28 de fevereiro de 2023
O Amazon Aurora MySQL versão 1 foi descontinuado	O Aurora MySQL versão 1 (compatível com o MySQL 5.6) foi descontinuado. Para ter mais informações, consulte Por quanto tempo as versões principais do Amazon Aurora permanecem disponíveis.	28 de fevereiro de 2023

Aurora Serverless v1 oferece suporte à configuração da janela de manutenção de cluster de banco de dados	Agora você pode definir a janela de manutenção para clusters de banco de dados do Aurora Serverless v1. Para ter mais informações, consulte Ajustar a janela de manutenção preferencial para clusters de banco de dados.	27 de fevereiro de 2023
O Amazon Aurora é compatível com fluxos de atividades de bancos de dados nas regiões Ásia-Pacífico (Haiderabade), Europa (Espanha) e Oriente Médio (EAU.	Para ter mais informações, consulte Database Activity Streams.	27 de janeiro de 2023
Amazon Aurora disponível na região Ásia-Pacífico (Melbourne	O Amazon Aurora já está disponível na região Ásia-Pacífico (Melbourne). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	23 de janeiro de 2023
Especificar a autoridade de certificação (CA) durante a criação do cluster de banco de dados	Agora você pode especificar qual CA usar para o certificado de servidor de um cluster de banco de dados durante a criação do cluster de banco de dados. Para ter mais informações, consulte Certificate authorities (Autoridades de certificação).	5 de janeiro de 2023

Compatibilidade com o Aurora MySQL 3.* para retrocesso	As versões do Aurora MySQL 3.* agora oferecem uma maneira rápida de recuperação após erros de usuários, como descartar a tabela incorreta ou excluir a linha incorreta. O Backtrack permite que você mova seu banco de dados para um período anterior, sem precisar restaurar a partir de um backup, e realiza isso em questão de segundos, até mesmo para bancos de dados maiores. Para obter detalhes, consulte Retroceder um cluster de bancos de dados Aurora.	4 de janeiro de 2023
Use as implantações azuis/verdes do Amazon RDS disponíveis em adicionais Regiões da AWS	O recurso Implantações azul/verde agora está disponível nas regiões China (Pequim) e China (Ningxia). Para ter mais informações, consulte Using Amazon RDS Blue/Green Deployments for database updates (Usar implantações azul/verde do Amazon RDS para atualizações de banco de dados).	22 de dezembro de 2022

Atualizar permissões de perfil vinculado a serviços do IAM	A política AmazonRDS ServiceRolePolicy agora concede permissões adicionais ao AWS Secrets Manager. Para ter mais informações, consulte Atualizações do Amazon RDS para políticas gerenciadas pela AWS.	22 de dezembro de 2022
O Amazon Aurora integra-se ao AWS Secrets Manager para gerenciamento de senhas	O Aurora pode gerenciar a senha de usuário principal para um cluster de banco de dados do Secrets Manager. Para ter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e AWS Secrets Manager.	22 de dezembro de 2022
Amazon Aurora oferece suporte ao Aurora Serverless v2 em Regiões da AWS adicionais	Aurora Serverless v2Agora o está disponível nas regiões África (Cidade do Cabo) e Europa (Milão). Para obter mais informações, consulte Aurora Serverless v2.	21 de dezembro de 2022
O Aurora PostgreSQL é compatível com o RDS Proxy com PostgreSQL 14	Agora, é possível criar um RDS Proxy com um cluster de bancos de dados do Aurora PostgreSQL 14. Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	13 de dezembro de 2022

O Amazon Aurora alerta você sobre anomalias recentes detectadas pelo Amazon DevOps Guru	A página de detalhes do banco de dados do console alerta você sobre anomalias atuais e as que ocorreram nas últimas 24 horas. Para ter mais informações, consulte Como o DevOps Guru para RDS funciona.	13 de dezembro de 2022
O Amazon RDS Proxy é compatível com bancos de dados globais	Agora você pode usar o RDS Proxy com bancos de dados globais do Aurora Para ter mais informações, consulte Using RDS Proxy with Aurora global databases (Usar o RDS Proxy com bancos de dados globais do Aurora).	7 de dezembro de 2022
Os clusters de banco de dados do Aurora PostgreSQL é compatível com Trusted Language Extensions para PostgreSQL	O Trusted Language Extensions para PostgreSQL é um kit de desenvolvimento de código aberto que permite criar extensões PostgreSQL de alta performance e executá-las com segurança em seu cluster de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Working with Trusted Language Extensions for PostgreSQL (Trabalhar com Trusted Language Extensions para PostgreSQL).	30 de novembro de 2022

O Amazon GuardDuty RDS Protection monitora ameaças de acesso	Quando você ativa o GuardDuty RDS Protection, o GuardDuty consome eventos de login do RDS de seus bancos de dados do Aurora, monitora esses eventos e cria perfis deles para possíveis ameaças internas ou agentes externos. Quando o GuardDuty RDS Protection detecta uma ameaça em potencial, o GuardDuty gera uma nova descoberta com detalhes sobre o banco de dados possivelmente comprometido. Para ter mais informações, consulte Monitoring threats with GuardDuty RDS Protection (Monitorar ameaças com o GuardDuty RDS Protection).	30 de novembro de 2022

Usar implantações azuis/verdes do Amazon RDS para atualizações de banco de dados	Você pode fazer alterações em um cluster de banco de dados em um ambiente de teste e testar as alterações sem afetar seu cluster de banco de dados de produção. Quando estiver pronto, você poderá promover o ambiente de teste como o novo ambiente de banco de produção, com tempo de inatividade mínimo. Para ter mais informações, consulte Using Amazon RDS Blue/Green Deployments for database updates (Usar implantações azul/verde do Amazon RDS para atualizações de banco de dados).	27 de novembro de 2022
O Amazon Aurora está disponível na região Ásia-Pacífico (Hyderabad	O Amazon Aurora já está disponível na região Ásia-Pacífico (Haiderabade). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	22 de novembro de 2022
O Amazon Aurora está disponível na região da Europa (Espanha	O Amazon Aurora já está disponível na região da Europa (Espanha). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	16 de novembro de 2022

Amazon Aurora disponível na região da Europa (Zurique	O Amazon Aurora já está disponível na região da Europa (Zurique). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	9 de novembro de 2022
O Amazon Aurora é compatível com a exportação de dados para o Amazon S3 de clusters de banco de dados	Agora você pode exportar dados do cluster do Aurora diretamente para o S3, sem precisar criar um snapshot primeiro. Para ter mais informações, consulte Exporting DB cluster data to Amazon S3 (Exportar dados de cluster de banco de dados para o Amazon S3).	27 de outubro de 2022
O Amazon Aurora MySQL é compatível com exportações mais rápidas para o Amazon S3	Agora você pode ver uma performance até 10 vezes mais rápida ao exportar dados de snapshot de cluster de banco de dados para o S3 para clusters do Aurora MySQL compatíveis com MySQL 5.7 e 8.0. Para ter mais informações, consulte Exportar dados de snapshot de cluster de banco de dados para o Amazon S3.	20 de outubro de 2022

O Amazon Aurora é compatível com a configuração automática da conectividade entre um cluster de banco de dados do Aurora e uma instância do EC	Você pode usar o Console de gerenciamento da AWS para configurar a conectividade entre um cluster de banco de dados existente do Aurora e uma instância do EC2. Para ter mais informações, consulte Connecting an EC2 instance and an Aurora DB cluster automatically (Conectar uma instância do EC2 e de um cluster de banco de dados do Aurora automaticamente).	14 de outubro de 2022
AWS JDBC Driver para PostgreSQL disponível para o público em geral	O JDBC Driver para PostgreSQL da AWS é um driver cliente desenvolvido para o Aurora PostgreSQL. O JDBC Driver para PostgreSQL da AWS já está disponível para o público em geral. Para ter mais informações, consulte Conectar-se ao AWS JDBC Driver para PostgreSQL.	6 de outubro de 2022
O Amazon Aurora é compatível com a atualização no local para o Aurora MySQL compatível com MySQL	Você pode realizar uma atualização no local para alterar um cluster existente do Aurora MySQL compatível com MySQL 5.6 para um cluster do Aurora MySQL compatível com MySQL 8.0. Para ter mais informações, consulte Atualizar o Aurora MySQL 2.x para 3.x.	26 de setembro de 2022

O Performance Insights mostra as 25 principais consultas SQL	No painel do Performance Insights, a guia Top SQL (SQL principal) mostra as 25 consultas SQL que mais estão contribuindo para a carga do banco de dados. Para ter mais informações, consulte Visão geral da guia Top SQL (SQL principal).	13 de setembro de 2022
O Aurora MySQL é compatível com uma nova classe de instância de banco de dados	Agora é possível usar a classe de instância de banco de dados db.r6i em clusters de banco de dados do Aurora MySQL. Para ter mais informações, consulte Classes de instância de banco de dados.	13 de setembro de 2022
Amazon Aurora disponível na região Oriente Médio (EAU	O Amazon Aurora já está disponível na região Oriente Médio (EAU). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	30 de agosto de 2022

Amazon Aurora oferece suporte à configuração automática da conectividade com uma instância do EC	Ao criar um cluster de banco de dados do Aurora, você pode usar o Console de gerenciamento da AWS para configurar a conectividade entre uma instância do Amazon Elastic Compute Cloud e o novo cluster de banco de dados. Para ter mais informações, consulte Configurar a conectividade automática de rede com uma instância do EC2.	22 de agosto de 2022
Amazon Aurora é compatível com o modo de pilha dupla	Os clusters de banco de dados agora podem ser executadas no modo de pilha dupla. Nesse modo, os recursos podem se comunicar com o cluster de banco de dados por IPv4, IPv6 ou ambos. Para ter mais informações, consulte Endereçamento IP do Amazon Aurora.	17 de agosto de 2022

Amazon Aurora oferece suporte a upgrade no local para compatível com PostgreSQL Aurora Serverless v1	Você pode fazer um upgrade no local de um cluster do Aurora Serverless v1 compatível com PostgreSQL 10 a fim de alterar um cluster existente para um cluster do Aurora Serverless v1 compatível com PostgreSQL 11. Para ver o procedimento de upgrade no local, consulte Modificar um cluster de banco de dados do Aurora Serverless v1.	8 de agosto de 2022
Performance Insights compatível com a Região Ásia-Pacífico (Jacarta	Anteriormente, não era possível usar o Performance Insights na Região Ásia-Pacífico (Jacarta). Essa restrição não existe mais. Para ter mais informações, consulte Compatibilidade de Região da AWS para o Performance Insights.	21 de julho de 2022
Amazon Aurora compatível com uma nova classe de instância de banco de dados	Agora é possível usar a classe de instância de banco de dados db.r6i em clusters de banco de dados do Aurora PostgreSQL. Para ter mais informações, consulte Classes de instância de banco de dados.	14 de julho de 2022

RDS Performance Insights oferece suporte a períodos de retenção adicionais	Anteriormente, o Performance Insights oferecia apenas dois períodos de retenção: 7 dias (padrão) ou 2 anos (731 dias). Agora, se você precisar reter seus dados de desempenho por mais de 7 dias, especifiq ue de 1 a 24 meses. Para ter mais informações, consulte Preços e retenção de dados para o Performance Insights.	1° de julho de 2022
Amazon Aurora oferece suporte a upgrade no local para compatível com MySQL Aurora Serverless v1	Você pode realizar uma atualização no local de um cluster do Aurora Serverles s v1 compatível com MySQL 5.6 a fim de alterar um cluster existente para um cluster do Aurora Serverless v1 compatível com MySQL 5.7. Para ver o procedimento de upgrade no local, consulte Modificar um cluster de banco de dados do Aurora Serverles s v1.	16 de junho de 2022
O Aurora é compatível com a ativação do Amazon DevOps Guru no console do RDS	É possível ativar a cobertura do DevOps Guru para seus bancos de dados do Aurora pelo console do RDS. Para ter mais informações, consulte Configurar o DevOps Guru para RDS.	9 de junho de 2022

O Amazon Aurora é compatível com a publicação de eventos em tópicos criptografados do Amazon SNS	O Amazon Aurora agora pode publicar eventos em tópicos do Amazon Simple Notification Service (Amazon SNS) com criptografia do lado do servidor (SSE) ativada, para proteção adicional de eventos que transportam dados confidenciais. Para ter mais informações, consulte Assinar notificações de eventos do Amazon RDS.	1º de junho de 2022
O Amazon RDS publica métricas de uso no Amazon CloudWatch	O namespace `AWS/Usage` no Amazon CloudWatch inclui métricas de uso específico da conta para suas cotas de serviço do Amazon RDS. Para ter mais informações, consulte Métricas de uso do Amazon CloudWatch para o Amazon Aurora.	28 de abril de 2022

Conjuntos de resultados da API de dados no formato JSON	Um parâmetro opcional para o `ExecuteStatement` faz com que o conjunto de resultados da consulta seja retornado como uma string no formato JSON. O conjunto de resultados JSON é simples e convenien te para ser transformado em uma estrutura de dados na linguagem da aplicação . Para ter mais informações, consulte Processar resultados da consulta em formato JSON.	27 de abril de 2022
Amazon Aurora Serverless v2 O agora está disponível ao público em geral	Amazon Aurora Serverles s v2O geralmente está disponível para todos os usuários. Para ter mais informações, consulte Usar o Aurora Serverless v2.	21 de abril de 2022
O Aurora MySQL é compatíve l com pacotes de criptografia configuráveis	Com o Aurora MySQL, agora é possível usar pacotes de criptografia configuráveis para controlar a criptografia de conexão aceita pelo seu servidor de banco de dados. Para ter mais informações, consulte Configurar pacotes de criptografia para conexões com clusters de banco de dados MySQL do Aurora.	15 de abril de 2022

O Aurora PostgreSQL é compatível com o proxy do RDS com PostgreSQL 13	Agora, é possível criar um proxy do RDS com um cluster de bancos de dados do Aurora PostgreSQL 13. Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	4 de abril de 2022
Notas de lançamento do Aurora PostgreSQL	Agora existe um guia separado para as notas de lançamento do Amazon Aurora PostgreSQL. Para ter mais informações, consulte Notas de lançamento do Aurora PostgreSQL.	22 de março de 2022
Notas de lançamento do Aurora MySQL	Agora existe um guia separado para as notas de lançamento do Amazon Aurora MySQL. Para ter mais informações, consulte Notas de lançamento do Aurora MySQL.	22 de março de 2022
Aurora PostgreSQL é compatível com atualizações para versões principais múltiplas	Agora você pode fazer atualizações de versão dos clusters de banco de dados Aurora PostgreSQL em várias versões principais. Para ter mais informações, consulte Como realizar uma atualização de versão principal.	4 de março de 2022

O Aurora PostgreSQL é compatível com conjuntos de cifras configuráveis	Com o Aurora PostgreSQL versões 11.8 e posteriores, agora você pode usar conjuntos de cifras configuráveis para controlar a criptografia de conexão aceita pelo servidor de banco de dados. Para obter informações sobre como usar conjuntos de cifras configuráveis com o Aurora PostgreSQL, consulte Configurar conjuntos de cifras para conexões com clusters de banco de dados Aurora PostgreSQL.	4 de março de 2022
AWS JDBC Driver para MySQL da disponível para o público em geral	O JDBC Driver para MySQL da AWS é um driver de cliente projetado para a alta disponibilidade do Aurora MySQL. O JDBC Driver para MySQL da AWS já está disponível para o público em geral. Para ter mais informações, consulte Conectar com o JDBC Driver para MySQL da Amazon Web Services.	2 de março de 2022

O Aurora PostgreSQL 13.5 é compatível com o Babelfish para Aurora PostgreSQL 1.1.0	O Aurora PostgreSQL 13.5 é compatível com o Babelfish 1.1.0. Para ver uma lista de novos recursos, consulte 13.5. Para ver uma lista de recursos compatíveis em cada versão do Babelfish, consulte Funcionalidade compatível no Babelfish por versão. Para ter mais informações sobre uso, consulte o tópico Trabalhar com o Babelfish for Aurora PostgreSQL.	28 de fevereiro de 2022
O Amazon Aurora é compatível com transmissões de atividades de bancos de dados na região Ásia-Pacífico (Jacarta	Para ter mais informações, consulte Compatibilidade de Regiões da AWS com transmissões de atividades de bancos de dados.	16 de fevereiro de 2022
O Performance Insights é compatível com novas operações de API	O Performance Insights é compatível com as seguintes operações de API: `GetResourceMetadata` , `ListAvailableResourceDimensions` e `ListAvailableResourceMetrics` . Para ter mais informações, consulte Recuperando métricas com a API do Performance Insights neste manual e na Referência de API do Amazon RDS Performance Insights.	12 de janeiro de 2022

O Proxy do Amazon RDS é compatível com eventos	O proxy do RDS agora gera eventos que você pode assinar e visualizar no CloudWatch Events ou configurar para enviar para o Amazon EventBridge. Para ter mais informações consulte Trabalhando com eventos RDS Proxy.	11 de janeiro de 2022
Proxy do RDS disponível em adicionais Regiões da AWS	O proxy do RDS já está disponível nas seguintes regiões: África (Cidade do Cabo), Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Osaka), Europa (Milão), Europa (Paris), Oriente Médio (Bahrein) e América do Sul (São Paulo). Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	5 de janeiro de 2022
Amazon Aurora disponível na região Ásia-Pacífico (Jacarta	O Amazon Aurora já está disponível na região Ásia-Pacífico (Jacarta). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	13 de dezembro de 2021

O DevOps Guru for Amazon RDS fornece insights detalhados e recomendações para o Amazon Aurora	O DevOps Guru for RDS explora o Performance Insights em busca de dados de performance. Utilizando esses dados, o serviço analisa a performance das suas instâncias de bancos de dados Amazon Aurora e pode ajudar a solucionar problemas de performance. Para saber mais, consulte o tópico sobre como Analisar anomalias de performance com o DevOps Guru for RDS, neste guia, e a Visão geral do DevOps Guru for RDS, no Guia do usuário do Amazon DevOps Guru.	1º de dezembro de 2021
O Aurora PostgreSQL é compatível com o proxy do RDS com PostgreSQL 12	Agora, é possível criar um proxy do RDS com um cluster de bancos de dados Aurora PostgreSQL 12. Para ter mais informações sobre o RDS Proxy, consulte o tópico sobre como Utilizar o proxy do Amazon RDS.	22 de novembro de 2021
O Aurora inclui suporte para classes de instância do AWS Graviton2 para fluxos de atividades de banco de dados	É possível utilizar fluxos de atividades de banco de dados com a classe de instância db.r6g para o Aurora MySQL e o Aurora PostgreSQL. Para ter mais informações, consulte o tópico sobre Classes de instâncias de banco de dados com suporte.	3 de novembro de 2021

Suporte do Amazon Aurora para entre contas AWS KMS keys	É possível utilizar uma chave do KMS de outra Conta da AWS para criptografia ao exportar snapshots de banco de dados para o Amazon S3. Para ter mais informações, consulte Exportar dados de snapshot de banco de dados para o Amazon S3.	3 de novembro de 2021
O Amazon Aurora é compatível com o Babelfish for Aurora PostgreSQL	O Babelfish for Aurora PostgreSQL amplia seu banco de dados Amazon Aurora, edição compatível com PostgreSQL, com a capacidade de aceitar conexões de banco de dados provenientes de clientes Microsoft SQL Server. Para ter mais informações, consulte o tópico sobre como Trabalhar com o Babelfish for Aurora PostgreSQL.	28 de outubro de 2021
Aurora Serverless O v1 pode exigir SSL para conexões	Os parâmetros de cluster `force_ssl` para PostgreSQL e `require_secure_transport` para MySQL do Aurora já são compatíveis com o Aurora Serverless v1. Veja mais informações em Usar TLS/SSL com Aurora Serverless v1.	26 de outubro de 2021

O Amazon Aurora é compatível com o Performance Insights em adicionais Regiões da AWS	O Insights Performance está disponível nas regiões Oriente Médio (Bahrein), África (Cidade do Cabo), Europa (Milão) e Ásia-Pacífico (Osaka). Para ter mais informações, consulte Compatibilidade de Região da AWS para o Insights de Performance.	5 de outubro de 2021
Tempo limite de autoescalabilidade configurável para o Aurora Serverless v1	Você pode escolher quanto tempo o Aurora Serverless v1 aguardará para encontrar um ponto de autoescalabilidade. Se nenhum ponto de autoescalabilidade for encontrado durante esse período, o Aurora Serverless v1 cancelará o evento de escalabilidade ou imporá a alteração de capacidade, dependendo da ação de tempo limite selecionada. Para ter mais informações, consulte Autoescalabilidade para o Aurora Serverless v1.	10 de setembro de 2021

O Aurora é compatível com as classesde instâncias X2g e T4g	O Aurora MySQL e o Aurora PostgreSQL agora podem usar classes de instâncias X2g e T4g. As classes de instâncias permitidas para uso dependem da versão do Aurora MySQL ou Aurora PostgreSQL. Para obter informações sobre os tipos de instâncias compatíveis, consulte Classes de instâncias de bancos de dados.	10 de setembro de 2021
O Amazon RDS é compatível com o proxy do RDS em uma VPC compartilhada	Agora você pode criar um proxy do RDS em uma nuvem privada virtual (VPC) compartilhada. Para ter mais informações sobre o RDS Proxy, consulte "Gerenciamento de conexões com o proxy do Amazon RDS" no Manual do usuário do Amazon RDS ou no Manual do usuário do Aurora.	6 de agosto de 2021
Página da política de versão do Aurora	O Manual do usuário do Amazon Aurora agora contém uma seção com informações gerais sobre as versões do Aurora e as políticas associadas. Para obter mais detalhes, consulte versões do Amazon Aurora.	14 de julho de 2021

Exclua eventos da API de dados de uma trilha do AWS CloudTrail	É possível excluir eventos da API de dados de uma trilha do CloudTrail. Para ter mais informações, consulte Excluir eventos da API de dados de uma trilha do AWS CloudTrail.	2 de julho de 2021
O Amazon Aurora edição compatível com PostgreSQL comporta extensões adicionais	Entre as novas extensões compatíveis estão: pg_bigm, pg_cron, pg_partman e pg_proctab. Para ter mais informações, consulte Versões de extensão para o Amazon Aurora, Edição compatível com PostgreSQL.	17 de junho de 2021
Clonagem para clusters do Aurora Serverless	Agora é possível criar clusters clonados que são Aurora Serverless. Para obter informações sobre clonagem, consulte Clonar um volume do cluster de bancos de dados Aurora.	16 de junho de 2021
Os bancos de dados globais do Aurora estão disponíveis nas regiões China (Pequim) e China (Ningxia	Agora você pode criar bancos de dados globais do Aurora nas regiões China (Ningxia) e China (Pequim). Para obter informações sobre os bancos de dados globais do Aurora, consulte Trabalhar com bancos de dados globais do Amazon Aurora.	19 de maio de 2021

Compatibilidade com o FIPS 140-2 para API de dados	A API de dados oferece suporte à publicação do Padrão federal de processamento de informações 140-2 (FIPS 140-2) para conexões SSL/TLS. Para ter mais informações, consulte Disponibilidade da API de dados.	14 de maio de 2021
AWS Driver JDBC da para PostgreSQL (pré-visualização)	O Driver JDBC da AWS para PostgreSQL está disponível na pré-visualização e é um driver do cliente desenvolvido para a alta disponibilidade do Aurora PostgreSQL. Para ter mais informações, consulte Conectando com o Driver JDBC da Amazon Web Services para PostgreSQL (pré-visualização).	27 de abril de 2021
A API de dados disponível em adicionais Regiões da AWS	A API de dados agora está disponível nas regiões Ásia-Pacífico (Seul) e Canadá (Central). Para ter mais informações, consulte Disponibilidade da API de dados.	9 de abril de 2021

O Amazon Aurora é compatível com as classes de instância de banco de dados Graviton	Agora é possível usar as classes de instância db.r6g.x de banco de dados Graviton2 para criar clusters de banco de dados que executam MySQL ou PostgreSQL. Para ter mais informações, consulte Tipos de classe de instância de banco de dados.	12 de março de 2021
Aprimoramentos de endpoint de proxy do RDS	Você pode criar endpoints adicionais associados a cada proxy do RDS . A criação de um endpoint em uma VPC diferente permite o acesso do proxy entre VPCs. Proxies para clusters do Aurora MySQL também podem ter endpoints somente leitura. Esses endpoints de leitor se conectam a instância s de banco de dados de leitores nos clusters e podem melhorar a escalabilidade e a disponibilidade de leitura para aplicações com uso intensivo de consultas. Para ter mais informações sobre o RDS Proxy, consulte "Gerencia r conexões com o proxy do Amazon RDS" no Guia do usuário do Amazon RDS ou no Guia do usuário do Aurora.	8 de março de 2021

Amazon Aurora disponível na região Ásia-Pacífico (Osaka	O Amazon Aurora já está disponível na região Ásia-Pacífico (Osaka). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	1.º de março de 2021
O Aurora PostgreSQL permite a habilitação da autenticação do IAM e do Kerberos no mesmo cluster de banco de dados	O Aurora PostgreSQL agora suporta a habilitação da autenticação do IAM e do Kerberos no mesmo cluster de banco de dados. Para ter mais informações, consulte Autenticação de banco de dados com o Amazon Aurora.	24 de fevereiro de 2021
O banco de dados global Aurora agora oferece suporte a failover planejado gerenciado	O banco de dados global Aurora agora oferece suporte a failover planejado gerenciado, permitindo que você altere com mais facilidade a região da AWS principal do seu banco de dados global Aurora. Você pode usar o failover planejado gerenciado somente com bancos de dados Aurora globais íntegros. Para saber mais, consulte Recuperação de desastres e Amazon Aurora bancos de dados globais. Para obter informações de referência, consulte `FailoverGlobalCluster` no Amazon RDS API Reference.	11 de fevereiro de 2021

API Data para Aurora Serverless agora oferece suporte para mais tipos de dados	Com a API de dados para Aurora Serverless, agora você pode usar `UUID` e `JSON` tipos de dados como entrada para seu banco de dados. Também com a API de dados para Aurora Serverless, agora você pode ter um valor de tipo `LONG` retornado do seu banco de dados como um valor `STRING`. Para saber mais, consulte Chamar API de dados. Para obter informaçõ es de referência sobre tipos de dados compatíveis, consulte `SqlParameter` no Referênci a de API do Data Service do Amazon RDS.	2 de fevereiro de 2021
O Aurora PostgreSQL é compatível com atualizações de versões principais para o PostgreSQL 12	Com o Aurora PostgreSQ L, agora é possível atualizar o mecanismo de banco de dados para a versão principal 12. Para ter mais informaçõ es, consulte Atualizar o mecanismo de banco de dados PostgreSQL para o Aurora PostgreSQL.	28 de janeiro de 2021

O Aurora MySQL é compatível com atualização no local	Você pode atualizar seu cluster Aurora MySQL 1.x para Aurora MySQL 2.x e preservar as instâncias de banco de dados, endpoints e assim por diante do cluster original. Essa técnica de atualizaç ão no local evita a inconveni ência de configurar um cluster totalmente novo restauran do um snapshot. Ele também evita a sobrecarga de copiar todos os dados da sua tabela em um novo cluster. Para ter mais informações, consulte Atualizar a versão principal de um cluster de Aurora MySQL banco de dados de 1.x para 2.x.	11 de janeiro de 2021
AWS JDBC Driver for MySQL (previsualização)	O Driver JDBC da AWS para MySQL, agora está disponíve l para demonstração, é um driver do cliente desenvolv ido para a alta disponibi lidade do Aurora MySQL. Para ter mais informações, consulte Conectar-se com o driver JDBC da Amazon Web Services para MySQL (pré-visualização).	7 de janeiro de 2021

O Aurora é compatível com fluxos de atividades de banco de dados em clusters secundários de um banco de dados global	É possível iniciar um banco de dados de um fluxo de atividade de banco de dados em um cluster primário ou secundário de Aurora PostgreSQL ou Aurora MySQL. Para obter versões de mecanismos compatíve is, consulte Limitações de bancos de dados globais do Aurora.	22 de dezembro de 2020
Clusters de vários mestres com quatro instâncias de banco de dados	O número máximo de instâncias de banco de dados em um cluster Aurora MySQL de vários mestre agora é quatro. Antes, o máximo era duas instâncias de banco de dados. Para ter mais informações, consulte Trabalhar com clusters de vários mestres do Aurora.	17 de dezembro de 2020
O Aurora PostgreSQL oferece suporte a funções do AWS Lambda	Agora você pode chamar a função do AWS Lambda para seus clusters de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Como invocar uma função Lambda a partir de um cluster de bancos de dados Aurora PostgreSQL.	11 de dezembro de 2020

O Amazon Aurora é compatível com as classes de instância de banco de dados Graviton2 na pré-visualização	Agora é possível usar as classes de instância db.r6g.x de banco de dados Graviton2 na demonstração para criar clusters de banco de dados executando MySQL ou PostgreSQL. Para ter mais informações, consulte Tipos de classe de instância de banco de dados.	11 de dezembro de 2020
Amazon Aurora Serverless v2 O agora está disponível na pré-visualização.	Amazon Aurora Serverless v2 está disponível na pré-visualização. Para trabalhar com o Amazon Aurora Serverless v2, solicite acesso. Para ter mais informações, consulte a Aurora Serverless v2 página.	1º de dezembro de 2020

O Aurora PostgreSQL agora está disponível para o Aurora Serverless em mais Regiões da AWS.	O Aurora PostgreSQL agora está disponível para o Aurora Serverless em mais Regiões da AWS. Agora você pode optar por executar o Aurora PostgreSQL Serverless v1 nas mesmas Regiões da AWS que oferecem o Aurora MySQL Serverless v1. Outras Regiões da AWS compatíveis com o Aurora Serverless incluem: Oeste dos EUA (Norte da Califórnia), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Seul), Ásia-Pacífico (Mumbai), Canadá (Central), Europa (Londres) e Europa (Paris). Para ter uma lista de todas as regiões e os mecanismos de banco de dados do Aurora para Aurora Serverless, consulte Supported Regions and Aurora DB engines for Aurora Serverless v1. A API de dados do Amazon RDS para o Aurora Serverless também está disponível nessas mesmas Regiões da AWS. Para ter uma lista de todas as regiões com suporte para API de dados para Aurora Serverless, consulte API de dados com o Aurora MySQL Sem Servidor v1.	24 de novembro de 2020

O Amazon RDS Performance Insights introduz novas dimensões	Você pode agrupar a carga do banco de dados de acordo com os grupos de dimensões para banco de dados, aplicação (PostgreSQL) e tipo de sessão (PostgreSQL). O Amazon RDS também é compatível com as dimensões db.name, db.application.name (PostgreSQL) e db.session_type.name (PostgreSQL). Para ter mais informações, consulte Tabela de carga superior.	24 de novembro de 2020
Aurora Serverless O oferece suporte para a versão 10.12 do Aurora PostgreSQL 10.12	O Aurora PostgreSQL for Aurora Serverless foi atualizado para o Aurora PostgreSQL versão 10.12 em todas as regiões da AWS que oferecem suporte para o Aurora PostgreSQL for Aurora Serverless. Para ter mais informações, consulte Supported Regions and Aurora DB engines for Aurora Serverless v1.	4 de novembro de 2020

A API de dados agora é compatível com a autorização baseada em etiqueta	A API de dados oferece suporte à autorização baseada em tag. Se você rotulou seus recursos de cluster do RDS com tags, será possível usar essas tags nas instruções de política para controlar o acesso por meio da API de dados. Para ter mais informações, consulte Autorizar o acesso à API de dados.	27 de outubro de 2020
O Amazon Aurora amplia o suporte para exportação de snapshots para o Amazon S3	Agora, é possível exportar dados de snapshots do banco de dados para o Amazon S3 em todas as comerciais Regiões da AWS. Para ter mais informações, consulte Exportar dados de snapshot de banco de dados para o Amazon S3.	22 de outubro de 2020

O banco de dados global Aurora é compatível com a clonagem	Agora é possível criar clones dos clusters de banco de dados primários e secundários de seus bancos de dados globais do Aurora. É possível fazer isso usando o Console de gerenciamento da AWS e escolhendo a opção de menu Create clone (Criar clone). Também é possível usar a AWS CLI e executar o comando `restore-db-cluster-to-point-in-time` com a opção `--restore-type copy-on-write` . Usando o Console de gerenciamento da AWS ou a AWS CLI, também é possível clonar clusters de banco de dados de seus bancos de dados globais do Aurora em todas as contas da AWS. Para ter mais informações sobre clonagem, consulte Clonar um volume de cluster de bancos de dados Aurora.	19 de outubro de 2020

O Amazon Aurora é compatível com o redimensionamento dinâmico para o volume do cluster	A partir do Aurora MySQL 1.23 e 2.09 e o Aurora PostgreSQL 3.3.0 e Aurora PostgreSQL 2.6.0, o Aurora reduz o tamanho do volume do cluster depois de remover dados por meio de operações como `DROP TABLE`. Para aproveitar esse aprimoramento, atualize para uma das versões apropriadas, dependendo do mecanismo de banco de dados usado pelo cluster. Para obter informações sobre esse atributo e como verificar o espaço de armazenamento usado e disponível para um cluster do Aurora, consulte Gerenciar performance e escalabilidade para clusters de bancos de dados Aurora.	13 de outubro de 2020
O Amazon Aurora é compatível com tamanhos de volume de até 128 TiB	Os volumes de cluster novos e existentes do Aurora agora podem aumentar até um tamanho máximo de 128 tebibytes (TiB). Para ter mais informações, consulte Como o armazenamento do Aurora aumenta.	22 de setembro de 2020

O Aurora PostgreSQL é compatível com as classes de instância de banco de dados db.r5 e db.t3 na região da China (Ningxia	Agora é possível criar clusters de bancos de dados Aurora PostgreSQL na região da China (Ningxia) que usam as classes de instância de banco de dados db.r5 e db.t3. Para ter mais informações, consulte Classes de instância de banco de dados.	3 de setembro de 2020

Aprimoramentos de consulta paralela do Aurora	A partir do Aurora MySQL 2.09 e 1.23, você pode aproveitar os aprimoramentos do recurso de consulta paralela. Criar um cluster de consulta paralela não requer mais um modo de mecanismo especial. Agora você pode ativar e desativar a consulta paralela usando a opção de configuração `aurora_parallel_query` para qualquer cluster provisionado que esteja executando uma versão do Aurora MySQL compatível. Você pode atualizar um cluster existente para uma versão do Aurora MySQL compatível e usar a consulta paralela, em vez de criar um cluster e importar dados para ele. Você pode usar o Performance Insights para clusters de consulta paralela. Você pode interromper e iniciar clusters de consulta paralela. Você pode criar clusters de consulta paralela do Aurora compatíveis com o MySQL 5.7. A consulta paralela funciona para tabelas que usam o formato de linha `DYNAMIC`. Os clusters de consulta paralela podem usar a autenticação do AWS Identity and Access Management (IAM).	2 de setembro de 2020

	As instâncias de banco de dados de leitor em clusters de consulta paralela podem aproveitar o nível de isolament o `READ COMMITTED` . Agora você também pode criar clusters de consulta paralela em adicionais Regiões da AWS. Para ter mais informaçõ es sobre o recurso de consulta paralela e esses aprimoram entos, consulte Parallel query for Aurora MySQL.	
Alterações no parâmetro binlog_rows_query_log_events do Aurora MySQL	Agora, é possível alterar o valor do parâmetro de configuração do Aurora MySQ `binlog_rows_query_ log_events` . Anteriorm ente, esse parâmetro não era modificável.	26 de agosto de 2020

Suporte para atualizações automáticas de versões secundárias para o Aurora MySQL	Com o Aurora MySQL, a configuração Enable auto minor version upgrade (Habilitar atualização automática de versão secundária) agora entra em vigor quando você a especific a para um cluster de bancos de dados Aurora MySQL. Ao habilitar a atualizaç ão automática de versão secundária, o Aurora atualiza automaticamente para novas versões secundárias à medida que elas são lançadas. As atualizações automáticas ocorrem durante a janela de manutenção do banco de dados. Para o Aurora MySQL, este recurso aplica-se apenas ao Aurora MySQL versão 2, que é compatível com o MySQL 5.7. Inicialmente, o procedimento de atualizaç ão automática traz clusters de bancos de dados Aurora MySQL para a versão 2.07.2. Para ter mais informações sobre como esse recurso funciona com o Aurora MySQL, consulte Atualizaç ões e patches de banco de dados para o Amazon Aurora MySQL.	3 de agosto de 2020

O Aurora PostgreSQL é compatível com atualizações de versões principais para o PostgreSQL versão 11	Com o Aurora PostgreSQL, agora é possível atualizar o mecanismo de banco de dados para a versão principal 11. Para ter mais informações, consulte Atualizar o mecanismo de banco de dados PostgreSQL para o Aurora PostgreSQL.	28 de julho de 2020
O Amazon Aurora é compatível com o AWS PrivateLink	O Amazon Aurora agora oferece suporte à criação de endpoints da Amazon VPC para Amazon RDS chamadas de API a fim de manter o tráfego entre aplicações e o Aurora na rede da AWS. Para ter mais informações, consulte Amazon Aurora e endpoints da VPC de interface (AWS PrivateLink).	9 de julho de 2020
Proxy do RDS disponível para o público em geral	O proxy do RDS agora está disponível ao público em geral. É possível usar o proxy do RDS com o RDS for MySQL, Aurora MySQL, RDS for PostgreSQL e Aurora PostgreSQL para workloads de produção. Para ter mais informações sobre o RDS Proxy, consulte "Gerenciar conexões com o proxy do Amazon RDS" no Guia do usuário do Amazon RDS ou no Guia do usuário do Aurora.	30 de junho de 2020

Encaminhamento de gravação do banco de dados global Aurora	Agora é possível habilitar o recurso de gravação em clusters secundários em um banco de dados global. Com o encaminhamento de gravação, você emite instruções DML em um cluster secundário, o Aurora encaminha a gravação para o cluster primário e os dados atualizados são replicados para todos os clusters secundários. Para ter mais informações, consulte Encaminhar gravação para Regiões da AWS secundárias com um banco de dados global do Aurora.	18 de junho de 2020
O Aurora é compatível com a integração com o AWS Backup	Você pode usar o AWS Backup para gerenciar backups de clusters de bancos de dados Aurora. Para ter mais informações, consulte Visão geral de backup e restauração de um cluster de bancos de dados Aurora.	10 de junho de 2020
O Aurora PostgreSQL é compatível com as classesde instância de banco de dados db.t3.large	Agora é possível criar clusters de bancos de dados do Aurora PostgreSQL que usam as classes de instâncias de banco de dados db.t3.large. Para ter mais informações, consulte Classes de instância de banco de dados.	5 de junho de 2020

O banco de dados global Aurora é compatível com o PostgreSQL versão 11.7 e o objetivo de ponto de recuperação gerenciado (RPO)	Agora é possível criar bancos de dados globais do Aurora para o mecanismo de banco de dados PostgreSQL versão 11.7. Também é possível gerenciar como um banco de dados global PostgreSQL se recupera de uma falha usando um Objetivo de ponto de recuperação (RPO). Para ter mais informações, consulte Recuperação de desastres entre regiões para bancos de dados globais do Aurora.	4 de junho de 2020
O Aurora MySQL é compatível como monitoramento de banco de dados com fluxos de atividades de banco de dados	O Aurora MySQL agora inclui fluxos de atividades de banco de dados, que fornecem um stream de dados quase em tempo real da atividade de banco de dados do seu banco de dados relacional. Para ter mais informações, consulte Usar os streams de atividade de banco de dados.	2 de junho de 2020
O editor de consultas está disponível em adicionais Regiões da AWS	O editor de consultas do Aurora Serverless agora está disponível em adicionais Regiões da AWS. Para ter mais informações, consulte Disponibilidade do editor de consultas.	28 de maio de 2020

A API de dados disponível em adicionais Regiões da AWS	A API de dados agora está disponível em adicionais Regiões da AWS. Para ter mais informações, consulte Disponibilidade da API de dados.	28 de maio de 2020
Proxy do RDS disponível em Região do Canadá (Central	Agora você pode usar a visualização do proxy do RDS em Região do Canadá (Central). Para ter mais informações sobre o RDS Proxy, consulte Gerenciar conexões com o proxy do Amazon RDS (pré-visu alização).	28 de maio de 2020
Banco de dados global Aurora e réplicas de leitura entre regiões	Para um banco de dados global Aurora, agora você pode criar uma réplica de leitura entre regiões Aurora MySQL do cluster primário na mesma região que um cluster secundário. Para ter mais informações sobre réplicas de leitura entre regiões e banco de dados global do Aurora, consulte Trabalhar com o Amazon Aurora Global Database e Replicar o banco de dados MySQL do Amazon Aurora.	18 de maio de 2020

Proxy do RDS disponível em mais Regiões da AWS	Agora é possível usar a visualização do proxy do RDS em Região do Oeste dos EUA (Norte da Califórnia), Região Europa (Londres), Região Europa (Frankfurt), Região da Ásia-Pacífico (Seul), Região da Ásia-Pacífico (Mumbai), Região Ásia-Pacífico (Cingapura) e Região Ásia-Pacífico (Sydney). Para ter mais informações sobre o RDS Proxy, consulte Gerenciar conexões com o proxy do Amazon RDS (pré-visualização).	13 de maio de 2020
O Aurora edição compatível com PostgreSQL é compatível com o Microsoft Active Directory on-premises ou auto-hospedado	Agora é possível usar um Active Directory no local ou auto-hospedado para a autenticação Kerberos de usuários quando eles se conectam aos seus clusters de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Usar a autenticação Kerberos com o Aurora PostgreSQL.	7 de maio de 2020

Clusters de vários mestres do Aurora MySQL disponíveis em mais Regiões da AWS	Agora é possível criar clusters de vários mestres do Aurora nas regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Tóquio), Ásia-Pacífico (Mumbai) e Europa (Frankfurt). Para ter mais informações sobre clusters de vários mestres, consulte Trabalhar com clusters de vários mestres do Aurora.	7 de maio de 2020
O Performance Insights é compatível com a análise de estatísticas de consultas do Aurora MySQL em execução	Agora, é possível analisar estatísticas de consultas em execução com o Performance Insights para instâncias de bancos de dados Aurora MySQL. Para ter mais informações, consulte Analisar estatísticas de consultas em execução.	5 de maio de 2020
Biblioteca de cliente Java para API de dados disponível ao público em geral	A biblioteca de cliente Java para a API de dados agora está disponível ao público em geral. Você pode baixar e usar uma biblioteca cliente Java para a API de dados. Ela permite mapear suas classes do lado do cliente para solicitações e respostas da API de dados. Para ter mais informações, consulte Usar a biblioteca cliente Java para a API de dados.	30 de abril de 2020

Amazon Aurora disponível na região Europa (Milão	O Amazon Aurora já está disponível na região Europa (Milão). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	28 de abril de 2020
Amazon Aurora disponível na região Europa (Milão	O Amazon Aurora já está disponível na região Europa (Milão). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	27 de abril de 2020
Amazon Aurora disponível na região da África (Cidade do Cabo	O Amazon Aurora já está disponível na região da África (Cidade do Cabo). Para ter mais informações, consulte Regiões e zonas de disponibilidade.	22 de abril de 2020
Agora o Aurora PostgreSQL é compatível com as classes de instância de banco de dados db.r5.16xlarge e db.r5.8xlarge	Agora é possível criar clusters de bancos de dados Aurora PostgreSQL que executam o PostgreSQL que usam as classes de instância de banco de dados db.r5.16xlarge e db.r5.8xlarge. Para ter mais informações, consulte Especificações de hardware para classes de instância de banco de dados para o Aurora.	8 de abril de 2020

Proxy do Amazon RDS para PostgreSQL	Agora o proxy do Amazon RDS está disponível para o PostgreSQL. É possível usar o proxy do RDS para reduzir a sobrecarga do gerenciamento de conexões no cluster e também a chance de erros de "muitas conexões". No momento, o proxy do RDS está em visualização pública para o PostgreSQL. Para ter mais informações, consulte Gerenciar conexões com o Amazon RDS Proxy (visualização) .	8 de abril de 2020
Os bancos de dados globais do Aurora agora são compatíveis com o Aurora PostgreSQL	Agora é possível criar bancos de dados globais do Aurora para o mecanismo de banco de dados PostgreSQL. Um banco de dados global do Aurora abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e recuperação de desastres decorrentes de interrupções de energia em toda a região. Para ter mais informações, consulte Como trabalhar com o Amazon Aurora Global Database.	10 de março de 2020

Compatibilidade com atualizações de versões principais para o Aurora PostgreSQL	Com o Aurora PostgreSQL, agora é possível atualizar o mecanismo de banco de dados para uma versão principal. Ao fazer isso, é possível avançar para uma versão principal mais nova ao atualizar versões selecionadas do mecanismo de banco de dados PostgreSQL. Para ter mais informações, consulte Atualizar o mecanismo de banco de dados PostgreSQL para o Aurora PostgreSQL.	4 de março de 2020
O Aurora PostgreSQL é compatível com a autenticação Kerberos	Agora é possível usar a autenticação Kerberos para autenticar usuários quando eles se conectam aos clusters de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Usar a autenticação Kerberos com o Aurora PostgreSQL.	28 de fevereiro de 2020
A API de dados é compatível com o AWS PrivateLink	A API de dados agora oferece suporte à criação de endpoints da Amazon VPC para chamadas da API de dados para manter o tráfego entre as aplicações e a API de dados na rede da AWS. Para ter mais informações, consulte Criar um endpoint da Amazon VPC (AWS PrivateLink) para a API de dados.	6 de fevereiro de 2020

Compatibilidade com machine learning do Aurora no Aurora PostgreSQL	A extensão `aws_ml` do Aurora PostgreSQL fornece funções que você usa em suas consultas de banco de dados para chamar o Amazon Comprehend para análise de sentimento e o SageMaker AI para executar seus próprios modelos de machine learning. Para ter mais informações, consulte Usar recursos de machine learning (ML) com o Aurora.	5 de fevereiro de 2020
O Aurora PostgreSQL é compatível com a exportação de dados para o Amazon S3	É possível consultar dados em um cluster de bancos de dados Aurora PostgreSQL e exportá-los diretamente para arquivos armazenados em um bucket do Amazon S3. Para ter mais informações, consulte Exportar dados de um cluster de banco de dados PostgreSQL do Aurora para o Amazon S3.	5 de fevereiro de 2020
Compatibilidade para exportação de dados de snapshots de banco de dados para o Amazon S3	O Amazon Aurora é compatível com a exportação de dados de snapshot de banco de dados para o Amazon S3 para MySQL e PostgreSQL. Para ter mais informações, consulte Exportar dados de snapshot de banco de dados para o Amazon S3.	9 de janeiro de 2020

Notas de lançamento do Aurora MySQL no histórico do documento	Esta seção agora inclui entradas do histórico de notas de release do Aurora edição compatível com MySQL para as versões lançadas após 31 de agosto de 2018. Para obter as notas de release completas de uma versão específica, escolha o link na primeira coluna da entrada do histórico.	13 de dezembro de 2019
Proxy do Amazon RDS	Você pode reduzir a sobrecarga do gerenciamento de conexões em seu cluster e reduzir a possibilidade de erros de “conexões em excesso” usando o proxy do Amazon RDS. Associe cada proxy a uma instância de banco de dados do RDS ou cluster de bancos de dados Aurora. Então, você usa o endpoint do proxy na string de conexão de seu aplicativo. O Proxy do Amazon RDS, atualmente, está em um estado de demonstração pública. Ele é compatível com o mecanismo de banco de dados do Aurora MySQL. Para ter mais informações, consulte Gerenciar conexões com o Amazon RDS Proxy (visualização) .	3 de dezembro de 2019

A API Data para Aurora Serverless v1 oferece suporte a dicas de mapeamento de tipos de dados	Agora, você pode usar uma dica para instruir a API de dados para Aurora Serverless v1 a enviar um valor `String` ao banco de dados como um tipo diferente. Para ter mais informações, consulte Chamar a API de dados.	26 de novembro de 2019
A API de dados para o Aurora Serverless v1 oferece suporte a uma biblioteca de cliente Java (prévia)	Você pode fazer download e usar uma biblioteca cliente Java para a API de dados. Ela permite mapear suas classes do lado do cliente para solicitações e respostas da API de dados. Para ter mais informações, consulte Usar a biblioteca cliente Java para a API de dados.	26 de novembro de 2019
O Aurora PostgreSQL é ALTAMENTE elegível para o FedRAMP	O Aurora PostgreSQL é ALTAMENTE elegível para o FedRAMP Para obter detalhes sobre a AWS e os esforços de conformidade, consulte os serviços AWS no escopo pelo programa de conformidade.	26 de novembro de 2019

Nível de isolamento READ COMMITTED habilitado para réplicas do Amazon Aurora MySQL	Agora, você pode habilitar o nível de isolamento `READ COMMITTED` em réplicas do Aurora MySQL. Isso requer que definição da configuração `aurora_read_replica_read_committed_isolation_enabled` seja habilitada no nível da sessão. Usar o nível de isolamento `READ COMMITTED` para consultas de longa execução em clusters OLTP pode ajudar a resolver problemas com comprimento da lista de histórico. Antes de habilitar essa configuração, certifique-se de compreender como o comportamento de isolamento em Réplicas do Aurora difere da implementação comum do MySQL de `READ COMMITTED`. Para ter mais informações, consulte Níveis de isolamento do Aurora MySQL.	25 de novembro de 2019
O Performance Insights oferece suporte para a análise de estatísticas de consultas Aurora PostgreSQL em execução	Agora, você pode analisar estatísticas de consultas em execução com o Performance Insights para instâncias de banco de dados Aurora PostgreSQL. Para ter mais informações, consulte Analisar estatísticas de consultas em execução.	25 de novembro de 2019

Mais clusters em um banco de dados global Aurora	Agora, você pode adicionar várias regiões secundárias a um banco de dados global Aurora. Você pode aproveitar as leituras globais de baixa latência e a recuperação de desastres em uma área geográfica mais ampla. Para ter mais informações sobre os bancos de dados globais do Aurora, consulte Trabalhar com bancos de dados globais do Amazon Aurora.	25 de novembro de 2019
Compatibilidade com machine learning do Aurora no Aurora MySQL	No Aurora MySQL 2.07 e versões posteriores, você pode chamar Amazon Comprehend para análise de sentimento e SageMaker AI para uma ampla variedade de algoritmos de machine learning. Os resultados são usados diretamente no seu aplicativo de banco de dados, por meio da incorporação de chamadas a funções armazenadas nas suas consultas. Para ter mais informações, consulte Usar recursos de machine learning (ML) com o Aurora.	25 de novembro de 2019

O banco de dados global do Aurora não requer mais a configuração do modo do mecanismo	Você não precisa mais especificar `--engine-mode=global` ao criar um cluster destinado a fazer parte de um banco de dados global Aurora. Todos os clusters Aurora que atenderem aos requisitos de compatibilidade serão elegíveis para fazer parte de um banco de dados global. Por exemplo, o cluster atualmente deve usar o Aurora MySQL versão 1 com compatibilidade com o MySQL 5.6. Para obter informações sobre os bancos de dados globais do Aurora, consulte Trabalhar com bancos de dados globais do Amazon Aurora.	25 de novembro de 2019

O banco de dados global do Aurora está disponível para o Aurora MySQL versão 2	A partir do Aurora MySQL 2.07, você pode criar um banco de dados global Aurora compatível com o MySQL 5.7. Você não precisa especific ar o modo do mecanismo `global` para os clusters primários ou secundários. É possível adicionar qualquer novo cluster provisionado com o Aurora MySQL 2.07 ou superior para um banco de dados global Aurora. Para obter informações sobre o Aurora Global Database, consulte Trabalhar com o Amazon Aurora Global Database.	25 de novembro de 2019
Otimização da disputa de linha dinâmica do Aurora MySQL disponível sem o modo de laboratório	A otimização da disputa de linha dinâmica agora está geralmente disponível para o Aurora MySQL e não requer a ativação da configuração do modo de laboratório do Aurora. Esse recurso melhora substancialmente as taxas de transferência para workloads com muitas transações disputando linhas na mesma página. Essa melhoria envolve a alteração do algoritmo de liberação de bloqueios usado pelo Aurora MySQL.	19 de novembro de 2019

Junções de hash do Aurora MySQL disponíveis sem o modo de laboratório	O atributo junção de hash já está geralmente disponível para o Aurora MySQL e não exige o modo de laboratório do Aurora definido como ativado. Esse atributo pode melhorar a performance da consulta quando você precisa unir uma grande quantidade de dados usando um equijoin. Para ter mais informações sobre o uso desse recurso, consulte Como trabalhar com junções de hash no Aurora MySQL.	19 de novembro de 2019
Compatibilidade com o Aurora MySQL2.* para mais classes de instâncias db.r	Os clusters do Aurora MySQL já oferecem suporte para os tipos de instância db.r5.8xlarge, db.r5.16xlarge e db.r5.24xlarge. Para ter mais informações sobre tipos de instância para clusters Aurora MySQL, consulte Escolher a classe da instância de banco de dados.	19 de novembro de 2019

Compatibilidade com o Aurora MySQL 2.* para retrocesso	As versões do Aurora MySQL 2.* agora oferecem uma maneira rápida de recuperação após erros de usuários, como descartar a tabela incorreta ou excluir a linha incorreta. O Backtrack permite que você mova seu banco de dados para um período anterior, sem precisar restaurar a partir de um backup, e realiza isso em questão de segundos, até mesmo para bancos de dados maiores. Para obter detalhes, consulte Retroceder um cluster de bancos de dados Aurora.	19 de novembro de 2019

Compatibilidade com etiquetas de faturamento do Aurora	Agora, você pode usar tags a fim de acompanhar a alocação de custos para recursos, como clusters do Aurora, instâncias de banco de dados dentro de clusters do Aurora, E/S, backups, snapshots e assim por diante. É possível ver os custos associados a cada tag usando o AWS Cost Explorer. Para ter mais informações sobre como usar tags com o Aurora, consulte Atribuição de tags aos recursos do Amazon RDS. Para obter informações gerais sobre tags e maneiras de usá-las para análise de custos, consulte Using cost allocation tags (Usar tags de alocação de custos) e User-defined cost allocation tags (Tags de alocação de custos definidas pelo usuário).	23 de outubro de 2019
API de dados para o Aurora PostgreSQL	Agora o Aurora PostgreSQL é compatível com o uso da API de dados com os clusters de banco de dados Amazon Aurora Serverless v1. Para ter mais informações, consulte Usar a API de dados do Aurora Serverless v1.	23 de setembro de 2019

O Aurora PostgreSQL comporta upload de logs de bancos de dados no CloudWatch Logs	É possível configurar seu cluster de bancos de dados Aurora PostgreSQL para publicar dados de log em um grupo no Amazon CloudWatch Logs. Com o CloudWatch Logs, você pode executar análise em tempo real de dados de log e usar o CloudWatch para criar alarmes e visualizar métricas. Você pode usar o CloudWatch Logs para armazenar seus registros de log em armazenamento resiliente. Para ter mais informações, consulte Publicar logs do Aurora PostgreSQL no Amazon CloudWatch Logs.	9 de agosto de 2019
Clusters de vários mestres para o Aurora MySQL	É possível configurar os clusters de vários mestres do Aurora MySQL. Nesses clusters, cada instância de banco de dados tem capacidade de leitura/gravação. Para ter mais informações, consulte Trabalhar com clusters de vários mestres do Aurora.	8 de agosto de 2019

O Aurora PostgreSQL é compatível com o Aurora Serverless v1	Agora, você pode usar o Amazon Aurora Serverless v1 com o Aurora PostgreSQL. Um cluster de bancos de dados Aurora Serverless inicia, encerra, amplia ou reduz automaticamente sua capacidade computacional de acordo com as necessidades do seu aplicativo. Para ter mais informações, consulte Usar o Amazon Aurora Serverless v1.	9 de julho de 2019
Clonagem entre contas para o Aurora MySQL	Agora, é possível clonar o volume do cluster para um cluster de bancos de dados Aurora MySQL entre contas da AWS. Você autoriza o compartilhamento por meio do AWS Resource Access Manager (AWS RAM). O volume de cluster clonado usa um mecanismo copy-on-write, que requer apenas armazenamento adicional para dados novos ou alterados. Para ter mais informações sobre clonagem para o Aurora, consulte Clonagem de bancos de dados em um cluster de bancos de dados Aurora.	2 de julho de 2019

O Aurora PostgreSQL é compatível com as classesde instância de banco de dados db.t	Agora, você pode criar clusters de bancos de dados do Aurora PostgreSQL que usam as classes de instâncias de banco de dados db.t3. Para ter mais informações, consulte Classe de instância de banco de dados.	20 de junho de 2019
Compatibilidade com importação de dados do Amazon S3 para o Aurora PostgreSQL	Agora, você pode importar dados de um arquivo do Amazon S3 para uma tabela em um cluster de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Importar dados do Amazon S3 para um cluster de banco de dados Aurora PostgreSQL.	19 de junho de 2019
O Aurora PostgreSQL agora oferece recuperaç ão de failover rápida com o gerenciamento de cache do cluster	O Aurora PostgreSQL agora oferece gerenciam ento de cache do cluster para garantir a recuperaç ão rápida da instância do banco de dados primário no caso de um failover. Para ter mais informações, consulte Recuperação rápida após failover com o gerenciamento de cache do cluster.	11 de junho de 2019

API Data para o Aurora Serverless v1 disponível para o público geral	É possível acessar clusters do Aurora Serverless v1 com aplicações baseadas em serviços Web usando a API de dados. Para ter mais informações, consulte Usar a API de dados do Aurora Serverless v1.	30 de maio de 2019
O Aurora PostgreSQL é compatível com o monitoramento de banco de dados com fluxos de atividades de banco de dados	O Aurora PostgreSQL agora inclui fluxos de atividades de banco de dados, que fornecem um fluxo de dados da atividade de banco de dados do seu banco de dados relacional quase em tempo real. Para ter mais informações, consulte Usar os streams de atividade de banco de dados.	30 de maio de 2019
Recomendações do Amazon Aurora	Agora, o Amazon Aurora fornece recomendações automatizadas para recursos do Aurora. Para ter mais informações, consulte Usar recomendações do Amazon Aurora.	22 de maio de 2019

O Performance Insights é compatível com o banco de dados global do Aurora	Agora você pode usar o Performance Insights com o banco de dados global do Aurora. Para obter informaçõ es sobre o Insights de Performance para o Aurora, consulte Usar o Amazon RDS Performance Insights. Para obter informações sobre os bancos de dados globais do Aurora, consulte Trabalhar com o Aurora Global Database.	13 de maio de 2019
O Performance Insights está disponível para o Aurora MySQL 5.7	O Amazon RDS Performance Insights está agora disponível para o Aurora MySQL versões 2.x, que são compatíveis com o MySQL 5.7. Para ter mais informações, consulte Uso de Insights de Performance do Amazon RDS.	3 de maio de 2019
Bancos de dados globais do Aurora disponíveis em mais Regiões da AWS	Agora é possível criar bancos de dados globais do Aurora na maioria das Regiões da AWS em que o Aurora está disponível. Para obter informações sobre os bancos de dados globais do Aurora, consulte Trabalhar com bancos de dados globais do Amazon Aurora.	30 de abril de 2019

Capacidade mínima de 1 para o Aurora Serverless v1	A configuração da capacidade mínima que você pode usar para um cluster do Aurora Serverless v1 é 1. Anteriormente, o mínimo era 2. Para obter informações sobre como especificar os valores de capacidade do Aurora Serverless, consulte Configurar a capacidade de um cluster de banco de dados do Aurora Serverless v1.	29 de abril de 2019
Aurora Serverless v1 Ação de tempo limite do	Agora você pode especificar a ação a ser tomada quando a alteração de uma capacidade do Aurora Serverless v1 atinge o tempo limite. Para ter mais informações, consulte Ação de tempo limite para alterações de capacidade.	29 de abril de 2019
Cobrança por segundo	O Amazon RDS agora é cobrado em incrementos de um segundo em todas as Regiões da AWS, exceto AWSGovCloud (EUA) para instâncias sob demanda. Para ter mais informações, consulte Cobrança da instância de banco de dados para o Aurora.	25 de abril de 2019

Compartilhar snapshots do Aurora Serverless v1 entre Regiões da AWS	Com o Aurora Serverless v1, os snapshots são sempre criptografados. Se você criptografar o snapshot com sua própria AWS KMS key, agora poderá copiá-lo ou compartilhá-lo entre Regiões da AWS. Para obter informações sobre snapshots de clusters de banco de dados de Aurora Serverless v1, consulte Aurora Serverless v1 e snapshots.	17 de abril de 2019
Restaurar backups do MySQL 5.7 usando o Amazon S3	Agora você pode criar um backup de seu banco de dados do MySQL versão 5.7, armazená-lo no Amazon S3 e depois restaurar o arquivo de backup em um novo cluster de bancos de dados Aurora MySQL. Para ter mais informações, consulte Migração de dados de um banco de dados MySQL externo para um cluster de bancos de dados Aurora MySQL.	17 de abril de 2019

Compartilhar snapshots do Aurora Serverless v1 entre regiões	Com o Aurora Serverless v1, os snapshots são sempre criptografados. Se você criptografar o snapshot com sua própria AWS KMS key, agora será possível copiar ou compartilhá-lo entre regiões. Para obter informações sobre snapshots de clusters de banco de dados do Aurora Serverless v1, consulte Aurora Serverless e snapshots.	16 de abril de 2019
Tutorial sobre prova de conceito do Aurora	Você pode aprender como realizar uma prova de conceito para experimentar seu aplicativo e sua workload com o Aurora. Para ver o tutorial completo, consulteRealizar uma prova de conceito do Aurora.	16 de abril de 2019
Aurora Serverless v1 O oferece suporte à restauração de um backup do Amazon S	Agora é possível importar backups do Amazon S3 para um cluster do Aurora Serverless. Para obter detalhes sobre esse procedimento, consulte Migrar dados do MySQL usando um bucket do Amazon S3.	16 de abril de 2019

Novos parâmetros modificáveis para o Aurora Serverless v1	Agora, você pode modificar os seguintes parâmetros de banco de dados para um cluster do Aurora Serverless v1: `innodb_file_format` , `innodb_file_per_table` , `innodb_large_prefix` , `innodb_lock_wait_timeout` , `innodb_monitor_disable` , `innodb_monitor_enable` , `innodb_monitor_reset` , `innodb_monitor_reset_all` , `innodb_print_all_deadlocks` , `log_warnings` , `net_read_timeout` , `net_retry_count` , `net_write_timeout` , `sql_mode` e `tx_isolation` . Para obter informações sobre os parâmetros de configuração para clusters do Aurora Serverless v1, consulte Aurora Serverless v1 e grupos de parâmetros.	4 de abril de 2019
O Aurora PostgreSQL é compatível com as classesde instância de banco de dados db.r	Agora você pode criar instâncias de bancos de dados do Aurora PostgreSQL que usam as classes de instâncias de banco de dados db.r5. Para ter mais informações, consulte Classe de instância de banco de dados.	4 de abril de 2019

Replicação lógica do Aurora PostgreSQL	Agora você pode usar a replicação lógica do PostgreSQL para replicar partes de um banco de dados para um cluster de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Usar replicação lógica do PostgreSQL.	28 de março de 2019
Compatibilidade com o GTID para o Aurora MySQL 2.04	Agora você pode usar a replicação com o recurso de ID da transação global (GTID) do MySQL 5.7. Esse recurso simplifica a replicação de log binário de performance (binlog) entre o Aurora MySQL e um banco de dados externo compatível com o MySQL 5.7. A replicação pode usar os cluster do Aurora MySQL com a origem ou o destino. Esse recurso está disponível para o Aurora MySQL 2.04 e posterior. Para ter mais informações sobre a replicação baseada em GTID e o Aurora MySQL, consulte Usar replicação baseada em GTID para o Aurora MySQL.	25 de março de 2019

Fazer o upload de logs do Aurora Serverless v1 no Amazon CloudWatch	Agora, você pode fazer com que o Aurora faça upload de logs de banco de dados no CloudWatch para um cluster do Aurora Serverless v1. Para ter mais informações, consulte Visualizar clusters de bancos de dados Aurora Serverless. Como parte desse aprimoramento, agora você pode definir valores para parâmetros no nível da instância em um parameter group do cluster de banco de dados, e esses valores se aplicam a todas as instâncias de banco de dados no cluster, a menos que elas sejam substituídas no parameter group do cluster de banco de dados. Para ter mais informações, consulte Trabalhar com grupos de parâmetros de banco de dados e grupos de parâmetros de cluster de banco de dados	25 de fevereiro de 2019
O Aurora MySQL é compatível com as classesde instância de banco de dados db.t	Agora, você pode criar clusters de bancos de dados do Aurora MySQL que usam as classes de instâncias de banco de dados db.t3. Para ter mais informações, consulte Classe de instância de banco de dados.	25 de fevereiro de 2019

O Aurora MySQL é compatível com as classesde instância de banco de dados db.r	Agora você pode criar instâncias de bancos de dados do Aurora MySQL que usam as classes de instâncias de banco de dados db.r5. Para ter mais informações, consulte Classe de instância de banco de dados.	25 de fevereiro de 2019
Contadores do Performance Insights para Aurora MySQL	Agora você pode adicionar contadores de performance aos gráficos do Performance Insights para instâncias de bancos de dados Aurora MySQL. Para ter mais informações, consulte Componentes do painel do Performance Insights.	19 de fevereiro de 2019
O Amazon RDS Performance Insights é compatível com a visualização de mais textos em SQL para o Aurora MySQL	O Amazon RDS Performance Insights agora é compatível com a visualização de mais textos em SQL no painel do Performance Insights para instâncias de bancos de dados Aurora MySQL. Para ter mais informações, consulte Visualizar mais textos em SQL no painel do Performance Insights.	6 de fevereiro de 2019

O Amazon RDS Performance Insights é compatível com a visualização de mais textos em SQL para o Aurora PostgreSQL	O Amazon RDS Performance Insights agora é compatível com a visualização de mais textos em SQL no painel do Performance Insights para instâncias de bancos de dados Aurora PostgreSQL. Para ter mais informações, consulte Visualizar mais textos em SQL no painel do Performance Insights.	24 de janeiro de 2019
Faturamento de backup do Aurora	Use as métricas do Amazon CloudWatch `TotalBackupStorageBilled` , `SnapshotStorageUsed` e `BackupRetentionPeriodStorageUsed` para monitorar o uso do espaço dos backups do Aurora. Para ter mais informações sobre como usar as métricas do CloudWatch, consulte Visão geral do monitoramento. Para ter mais informações sobre como gerenciar o armazenamento para dados de backup, consulte Noções básicas sobre o uso do armazenamento de backup do Aurora.	3 de janeiro de 2019

Contadores do Performance Insights	Agora você pode adicionar contadores de performance aos gráficos do Performance Insights. Para ter mais informações, consulte Componentes do painel do Performance Insights.	6 de dezembro de 2018
Banco de dados global Aurora	Agora crie bancos de dados globais do Aurora. Um banco de dados global do Aurora abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e recuperação de desastres decorrentes de interrupções de energia em toda a região. Para ter mais informações, consulte Como trabalhar com o Amazon Aurora Global Database.	28 de novembro de 2018
Gerenciamento de planos de consultas no Aurora PostgreSQL	O Aurora PostgreSQL agora oferece gerenciamento de planos de consultas que você pode usar para gerenciar planos de execução de consultas do PostgreSQL. Para ter mais informações, consulte Gerenciar planos de execução de consultas do Aurora PostgreSQL.	20 de novembro de 2018

Editor de consultas para Aurora Serverless v1 (beta)	Execute instruções SQL no console do Amazon RDS em clusters do Aurora Serverless v1. Para ter mais informações, consulte Usar o editor de consultas para Aurora Serverless v1.	20 de novembro de 2018
API Data para o Aurora Serverless v1 (beta)	É possível acessar clusters do Aurora Serverless v1 com aplicativos baseados em serviços Web usando a API Data. Para ter mais informações, consulte Usar a API de dados do Aurora Serverless.	20 de novembro de 2018
Compatibilidade com TLS para o Aurora Serverless v1	Aurora Serverless v1Os clusters do oferecem suporte à criptografia TLS/SSL. Para ter mais informações, consulte TLS/SSL para Aurora Serverless.	19 de novembro de 2018

Endpoints personalizados	Já é possível criar endpoints associados a um conjunto arbitrário de instâncias de banco de dados. Esse recurso ajuda no balanceamento de carga e na alta disponibilidade dos clusters do Aurora em que algumas instâncias de banco de dados têm capacidade ou configuração diferente s de outras. Use endpoints personalizados, em vez de se conectar a uma instância de banco de dados por meio do endpoint da instância. Para ter mais informações, consulte Endpoints do Amazon Aurora.	12 de novembro de 2018
Suporte à autenticação do IAM no Aurora PostgreSQL	O Aurora PostgreSQL agora é compatível com a autenticação do IAM. Para ter mais informações, consulte Autenticação do banco de dados IAM.	8 de novembro de 2018

Grupos de parâmetros personalizados para restauração e recuperação em um ponto anterior no tempo	Agora você pode especificar um parameter group personalizado ao restaurar um snapshot ou executar uma operação de recuperação em um ponto anterior no tempo. Para ter mais informações, consulte Restaurar a partir de um snapshot de cluster de banco de dados e Restaurar um cluster de banco de dados a um horário especificado.	15 de outubro de 2018
Proteção contra exclusão para clusters de bancos de dados Aurora	Quando você habilita a proteção contra exclusão para um cluster de banco de dados, o banco de dados não pode ser excluído por nenhum usuário. Para ter mais informações, consulte Excluir um cluster de banco de dados.	26 de setembro de 2018
Recurso Interromper/Iniciar do Aurora	Agora, você pode interromper ou iniciar um cluster inteiro do Aurora com uma única operação. Para ter mais informações, consulte Interromper e iniciar um cluster do Aurora.	24 de setembro de 2018

Recurso de consulta paralela do Aurora MySQL	O Aurora MySQL agora oferece uma opção para paralelizar as operações de E/S das consultas em toda a infraestrutura de armazenamento do Aurora. Esse recurso acelera as consultas analíticas com muitos dados, que geralmente são as operações mais demoradas em uma workload. Para ter mais informações, consulte Parallel query for Aurora MySQL.	20 de setembro de 2018
Novo guia	Esta é a primeira versão do Manual do usuário do Amazon Aurora.	31 de agosto de 2018

Glossário da AWS

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência do Glossário da AWS.